Paper 203-31

The "Handy-Dandy, Quick-n-Dirty" Automated Contrast Generator - A SAS/IML® Macro to Support the GLM, MIXED, and GENMOD Procedures

Paul A. Thompson, Ph.D.
Division of Biostatistics, School of Medicine, Washington University in St. Louis

ABSTRACT

Contrasts are an important component of the armamentarium of the statistician. In the SAS/STAT ® GLM, ANOVA, MIXED, and GENMOD procedures, the contrasts are used to answer specific additional questions. In many cases, it is difficult to define contrasts which are estimable, or correctly formed. A macro which converts a question about differences between cells (defined in several ways) into estimable contrasts is described. This macro uses SAS/IML ® to convert the comparison into the various components of the comparison into SAS ® system contrasts. Using several macro components unique to the SAS/IML system, the macro generates contrasts which are invariably estimable.

INTRODUCTION

The SAS ® System offers a very complete set of procedures for the applied experimental statistician. From the basic tecniques offered in PROC ANOVA to the workhorse PROC GLM to the sophisticated and contemporarily interesting PROC MIXED and PROC GENMOD, the applied statistician can find useful tools for many experimental situations (note that, although PROC CATMOD is often considered to be one of this set of experimental design tools, it is excluded due to technical differences in the contrast construction methods employed by the CATMOD developers). When using these procedures, the experiment designer generally defines factors or categorical independent variables, and then considers specific hypotheses about the components of the experiment. Continuous variables can also be included as well.

When the statistician considers the design, the factor-based differences between cells are usually of paramount importance. If we have an experiment involving two solutions (Factor S) and three concentrations of each (Factor C), we are usually interested in the differences between the levels of Factor S, the differences between the levels of Factor C, and the interaction between the two factors. These factors are tested using either a ratio of variance estimates or an estimated factor test (in PROC MIXED and PROC GENMOD).

After examining tests for factors and interactions, further investigations are often needed to clarify matters. In the experiment involving the two solutions and three concentrations, further investigations may be needed to understand a significant interaction. These further questions are answered using contrasts.

Here is an example of several contrasts. GA has 2 levels, GA has 3 levels, GA has 4 levels, so that the design has 24 cells (and all cells are represented).

```
PROC GLM;
     CLASSES GA GB GC;
     MODEL DV=GA GB GC GA*GB;
     CONTRAST "GA Overall"       GA  1 -1;
     CONTRAST "GB L1 V L2"       GB  1 -1  0;
     CONTRAST "GC L2 V L3"       GC  0 -1  1  0;
     CONTRAST "GB L1 V L3, GA:1" GC  1  0 -1  0  0  0;
RUN;
```

These contrasts are quite typical. In some cases, the contrasts are synonymous with other components of the design. In the example above, "GA Overall" is synonymous with the overall GA effect from the design as given. The other contrasts ask questions which cannot be addressed using standard design components. Thus, the applied statistician must understand contrasts to fully examine all relevant questions in most experimental design situations.

DEFINITION OF A CONTRAST

A contrast is considered to be an neutral question about the differences between parameters in an experimental design. Generally, this involves means of cells, but continuous variables may also be involved in contrasts. When the contrast is implemented as a coefficient vector $\boldsymbol{L}$, the contrast is used in conjunction with the matrix $\hat{\boldsymbol{\beta}}$. We form the expression:

$$\boldsymbol{L}'\hat{\boldsymbol{\beta}}\boldsymbol{u} = \boldsymbol{\Gamma}$$

where $\boldsymbol{L}$ is an vector comparing cells between groups, $\boldsymbol{u}$ is a vector comparing cells within groups, $\hat{\boldsymbol{\beta}}$ is the estimates of the coefficients for the experiment, and $\boldsymbol{\Gamma}$ is an *a priori*-specified statement of the results of the expression, often equal to 0.

Contrasts have several simple rules:

Rule 1: $\boldsymbol{L}'\boldsymbol{1} = 0$

This indicates that the sum of the contrast coefficients is 0, and may be interpreted as indicating that the contrast asks a neutral question about the differences between parameter estimates.

Rule 2: $\boldsymbol{L}'\boldsymbol{L} = 1$

This indicates that the coefficient sum of squares is 1. This rule is optional, as the coefficient sum of squares is involved in the computation of the coefficient test statistic Ψ.

Contrasts are often single *df* vectors, but need not be. When several contrasts are considered simultaeneously, the coefficients should be independent.

CONTRASTS WITHIN SAS

Within the SAS system, contrasts are used in a slightly different way than defined in Equation 1 above. Rather than a single vector/matrix $\boldsymbol{L}$, the SAS system requires that the contrasts be defined in terms of the experimental design being examined, and that the coefficients be identified with the effects and interactions of that design. In this discussion, these SAS contrasts will be refered to by the letter $\boldsymbol{c}$, and contrasts for a given factor will be defined by a subscript referencing that factor; if it is Factor A, we will use Contrast $\boldsymbol{c}_A$.

For example, consider a design in which we have Factors A (2 levels) and B (3 levels). If we use PROC GLM for this design, we would have:

```
PROC GLM DATA=CDS;
     CLASSES A B;
     MODEL DV=A B A*B;
     CONTRAST "B1 V B2: A=1"  A*B  1 -1  0  0  0  0;
RUN;
```

Conceptually, this is exactly what we are interested in; a comparison of the Levels 1 and 2 of Factor B, for the Level 1 of Factor A only.

If we were to actually run the code shown above, we would discover that the contrast `"B1 V B2: A=1"` as defined above would be `NON-EST`, or not estimable by the SAS system. That is because the contrast does not ask a proper question. The correct statement of the contrast required to compare the two cells is:

```
     CONTRAST "B1 V B2: A=1" B    1 -1  0  A*B  1 -1  0  0  0  0;
```

Estimability can be a difficult problem in working with SAS contrasts. With one factor, it is never a problem, as Vector $\boldsymbol{L}$ is synonymous with Vector $\boldsymbol{c}$. When there are only two factors, most users are able to define estimable contrasts, as they can often determine which effects to include, by trial-and-error if no other way. When designs get more complex, many users find that there are more problems in defining estimable contrasts.

It is sometimes stated that all problems with estimability of contrasts can be solved by working quite directly with Equation $\boldsymbol{L}'\hat{\boldsymbol{\beta}}\boldsymbol{u}$. In this view, a single classification variable can be substituted for all factors

in the design. The user would then perform all factor-oriented tests using appropriate contrasts. Additional questions could be addressed by adding other rows to $\boldsymbol{L}$. Although seemingly sensible, this idea is often incorrect. With multi-level analyses (tested in GLM or MIXED), the use of such an approach would both test overall hypotheses and contrasts incorrectly. Multi-level designs require the designation of Level-1, Level-2 and Level-3 effects in a separable and identifiable manner. Additionally, many users would probably find the process of correct construction of tests for factor comparisons and interactions correctly.

ESTIMABILITY

An estimable contrast may be thought of as a comparison or a question which makes sense in the context of the experiment at hand, and which is a proper linear combination of the parameter estimates for the experiment, including all relevant parameters. It is often quite easy to think of the question which needs to be asked in the experiment, but often the first, second or third attempt at the construction of the contrast within the SAS PROC results in the cryptic statement "NON-EST" in the output from the procedure.

Estimability is defined in the *SAS Help and Documentation System* in the section "The Four Types of Estimable Functions" (SAS Institute, 2002-2003), as follows:

> For linear models such as
>
> $$\boldsymbol{Y} = \boldsymbol{X}\beta + \epsilon$$
>
> with $E(\boldsymbol{Y}) = \boldsymbol{X}\beta$, a primary analytical goal is to estimate or test for the significance of certain linear combinations of the elements of β. This is accomplished by computing linear combinations of the observed $\boldsymbol{Y}$s. . . . (omit). . . "$\boldsymbol{L}\beta$ is estimable if and only if there is a linear combination of the rows of $\boldsymbol{X}$ that is equal to $\boldsymbol{L}$ . . . (omit). . . " Thus, the rows of $\boldsymbol{X}$ form a generating set from which any estimable $\boldsymbol{L}$ can be constructed. . . . (omit). . . Therefore, if $\boldsymbol{L}$ can be written as a linear combination of the rows of $\boldsymbol{X}$ . . . (omit). . . then $\boldsymbol{L}\beta$ is estimable.

The entire idea of the macro is contained in that quote. Contrasts which are guaranteed to be estimable can be constructed by the formation of a linear combination of the rows of $\boldsymbol{X}$. SAS contrasts $\boldsymbol{c}$ can be formed by:

1. determining the cells for comparison;
2. expressing this comparison in a matrix $\boldsymbol{L}$;
3. defining the design matrix $\boldsymbol{X}$;
4. computing $\boldsymbol{c}_{\text{Overall}} = \boldsymbol{L}'\boldsymbol{X}$; and
5. associating the components of $\boldsymbol{c}_{\text{Overall}}$ with their respective factors and interactions, so that the proper SAS contrast statement can be constructed.

Simple, eh?

A FEW NOTES TO START UP

Utility macro _itemcnt: When working with complex macros, it is useful to have macro functions defined to perform useful jobs. We need to count items in macro variables, and do so with Macro _itemcnt:

```
1 %macro _itemcnt(_items=a b,_delim=1,_n=_nit);   /* Count items in _items  */
2 %let _delx=%str( ,*\~/-);                       /* Delimeter list         */
3 %let &_n.=0;                                    /* Initialize counter     */
4 %let _ksep=%quote(%substr(&_delx.,&_delim.,1)); /* Select counter         */
5 %do %while(%length(%scan(%str(&_items.),%eval(&&&_n.+1),%str(&_ksep.))) > 0);
6 %let &_n.=%eval(&&&_n. + 1); %end;              /* Increment counter      */
7 %mend _itemcnt;
```

When called correctly, this macro returns the count as a value in the macro call. This is achieved by the use of the triple-ampersand construction "&&&_n" to recover the value of the count variable "&_n" For the macro to work properly, the macro variable "&_n" must be defined in the calling environment. Here is an example of the use of the macro:

```
/* Setting up and using the _itemcnt macro ***************************** */
%let _nv=;%let _itm=a b c;
%_itemcnt(_items=%str(&_itm),_n=_nv); %put _nv[&_nv];
/* Results of _itemcnt macro ******************************************* */
_nv[3]
```

Most SAS macro programs use a "call-by-value" approach, in which the macro parameters in the macro provide values, but are not returned if changed. This macro (and the macro for contrast construction) uses a "call-by-reference" approach, and returns the value computed by the macro as a variable supplied to the program.

Special SAS/IML macro characteristics: The SAS/IML procedure is used for the macro described in this paper. The SAS/IML environment has some special characteristics. In particular, since every statement is executed as encountered, macros behave in somewhat unexpected ways. Compare these two sets of statements:

```
/* Macros in the data step ********************************************* */
    %let _vl=1;%put Before:_vl[&_vl];
    data _null_;call symput("_vl","2");%put During:_vl[&_vl];run;
    %put After :_vl[&_vl];
/* Results from data step processing *********************************** */
    Before:_vl[1]
    During:_vl[1]
    After :_vl[2]
/* Macros in PROC IML ************************************************** */
    %let _vl=1;%put Before:_vl[&_vl];
    proc iml;vl={2};call symput("_vl",compress(char(vl)));
    %put During:_vl[&_vl];quit;
    %put After :_vl[&_vl];
/* Results from PROC IML processing ************************************ */
    Before:_vl[1]
    During:_vl[2]
    After :_vl[2]
```

The SAS/IML environment provides more flexibility in the use and applications of the macro toolkit than is available in other areas of SAS.

SAS/IML

The construction of SAS contrasts by matrix multiplication is relatively simple within the SAS/IML procedure. Many SAS users are not familiar with SAS/IML nor are they comfortable using this versatile and important tool. Thus, using a macro which invokes SAS/IML to generate contrasts can take advantage of the strengths of the SAS/IML system even for those uners unfamiliar with SAS/IML.

Within SAS, classificatory variables are parameterized such that each factor with k levels has k indicator-vector columns, each of which represents the values of one level. These indicator variables have values of 0 and 1 only. This design matrix is LTFR (less than full rank), and thus requires an estimation approach which does not involve simply inverting the $\boldsymbol{X}'\boldsymbol{X}$ matrix. The order of factors in the design matrix is given by the order that factors are listed on the CONTRAST statement.

The steps needed to build a contrast automatically in SAS are given here. For illustrative purposes, a design involving 3 factors will be used. The factors will be termed A, B, and C, with 2, 3, and 4 levels respectively. The design has 24 cells in total.

SPECIFY THE DESIGN

The design of the experiment must be determined first. This involves the construction of a matrix $\boldsymbol{X}$, which is constructed from the components from each factor and interaction. This matrix is constructed at the cell

level, rather than at the individual level, so that the final matrix will have 1 row for each cell, rather than 1 row for each individual in the experiment. In constructing this matrix, no attention need be paid to cases per cell. We assume that no empty cells are present in the design.

Main effects: The Kronecker product operator "@" is used to construct design matrix components. This operator produces a matrix by multiplying all items in each of the two matrices together. When one of the matrices is a 1 vector (all elements equal 1), the process duplicates the other matrix a given number of times:

```
  xa=j(2,1,1); xb=i(2); xc=j(3,1,1); xd=(xa@xb@xc)'; print xa xb xc xd;
```

results in (xd is transposed to save space):

```
 XA    XB     XC    XD
 1     1 0    1     1 1 1 0 0 0 1 1 1 0 0 0
 1     0 1    1     0 0 0 1 1 1 0 0 0 1 1 1
              1
```

When this is used to construct design matrices, the idea is used iteratively:

```
nrpre=1;       nrpst=1;
nrpst=nrpst*2; nrpst=nrpst*3;     nrpst=nrpst*4;
nrpst=nrpst/2; xa=j(nrpre,1,1)@i(2)@j(nrpst,1,1); print xa; nrpre=nrpre*2;
nrpst=nrpst/3; xb=j(nrpre,1,1)@i(3)@j(nrpst,1,1); print xb; nrpre=nrpre*3;
nrpst=nrpst/4; xc=j(nrpre,1,1)@i(4)@j(nrpst,1,1); print xc; nrpre=nrpre*4;
```

Macro _contrgen can be used to complete the same process:

```
/* Start-up definition of a contrast generator ********************** */
%macro _contrgen(_factnm=%str(a/b/c),_factlv=%str(2/3/4));
%let _fcnt=;%_itemcnt(_items=%str(&_factnm),_n=_fcnt,_delim=6);
%do _i=1 %to &_fcnt;
%let _fn&_i=%scan(%str(&_factnm),&_i,/);  /* Factor name                  */
%let _fl&_i=%scan(%str(&_factlv),&_i,/);  /* Number of levels in factor   */
%end;
PROC IML;
     NRPRE=1;NRPST=1;%do _i=1 %to &_fcnt;nrpst=nrpst*&&_fl&_i.;%end;
%do _i=1 %to &_fcnt;
     NRPST=NRPST/&&_fl&_i.;
     x&&_fn&_i.=j(nrpre,1,1)@i(&&_fl&_i.)@j(nrpst,1,1);PRINT &&_fn&_i.;
     NRPRE=NRPRE*&&_fl&_i.;
%end;
QUIT;
%mend  _ctrx;
```

This produces the following matrices (presented in transpose form, due to space limitations):

```
XA
 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
XB
 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1
XC
```

```
 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
```

Interaction effects: Once these components are in place, we can construct the interaction components, again using the Kronecker product operator "@" and the direct multiplication operator "#":

```
XAB=(XA@J(1,NCOL(XB),1))#(J(1,NCOL(XA),1)@XB);
```

This produces an interaction term by duplicating columns appropriately from each of the relevant design components, and then forming the direct product using the "#" operator (which multiplies corresponding elements in matrices on a cell-by-cell basis). Using the matrices above, we obtain the following matrices (presented in transpose form, due to space limitations):

```
XAB
 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

The macro _contrgen can be expanded and extended to perform this function.

A problem with interactions: When working with n-way interactions, the first and most obvious approach lies in defining the terms in the interaction using n do loops. With 3-way interactions, we need:

```
do i=1 to numfact-2;
   do j=i+1 to numfact-1;
      do k=j+1 to numfact;
         3-way interaction processes - effect i-j-k
      end;
   end;
end;
```

With a four-factor design, four levels of loops are needed and so forth. The number of loop structures expands as a hard, irreducible function of the number of factors.

There is a simple and straight-forward method for converting the requirement of loop construction to the number of factors. When we consider the number of factors and associated interactions, we obtain $2^f - 1$ terms. By running thru all numbers from 1 - $2^f - 1$, and using the bitmasking method, we can determine the full range of factors and interactions, and list these out in a linear manner. This converts a combinatorial explosing into a straight process linear in the number of factors. If we have 3 factors, the code is as follows:

```
/* Obtaining all combinations ****************************************** */
DATA _NULL_;
     DO I=1 TO 2**3-1;
        PUT I ":" @;
        IF (I = '1..'b) THEN PUT ", Factor A" @;
        IF (I = '.1.'b) THEN PUT ", Factor B" @;
        IF (I = '..1'b) THEN PUT ", Factor C" @;PUT ;
     END;
RUN;
/* Results from that code ********************************************** */
1 :, Factor C
```

```
2 :, Factor B
3 :, Factor B, Factor C
4 :, Factor A
5 :, Factor A, Factor C
6 :, Factor A, Factor B
7 :, Factor A, Factor B, Factor C
```

After generating the various combinations of factors, the ordering can be easily corrected.

Although PROC GLMMOD can be used to generate the design matrix, the output from this procedure is a series of columns identified by a neutral name, and the design matrix contains one row for each individual. Thus, post-processing for this output would be required. This approach does not simplify the requirements for the design matrix construction problem.

COMPUTING CONTRAST COMPONENTS

At this point, the contrast components can be easily computed as a matrix multiplication. With each component of the final design matrix $\boldsymbol{X}$, the matrix product:

```
\bm{c}_F = \bm{L}‘ \bm{X}
```

produces a component of the SAS coefficient set. If the elements of the vector $\boldsymbol{c}_F$ has non-zero elements, this component of the coefficient must be included in the contrast specification. Thus, the process involves performing the matrix multiplication, determining if the coefficients are non-zero, and including any such components in the final contrast statement.

ELIMINATING REDUNDANT TERMS

With the basic algorithm as described above, redundant coefficients are generated. For instance, using an L vector consistent with a main effect (a repeated comparison of the cells from one level with the cells from another level), several terms which are not needed for the final contrast are generated. For example, in the 3-factor situation shown above, an L vector for comparing 2 levels of the B effect is shown below. If this factor is used in the macro, the contrast which is produced is shown in Rows 2-3. This test can be performed by the reduced set shown in the Row 4.

```
1 L vector: 1 1 1 1 0 0 0 0 -1 -1 -1 -1 1 1 1 1 0 0 0 0 -1 -1 -1 -1
2 Contrast: B 8 0 -8 A*B 4 0 -4 4 0 -4 B*C 2 2 2 2 0 0 0 0 -2 -2 -2 -2
3           A*B*C 1 1 1 1 0 0 0 0 -1 -1 -1 -1 1 1 1 1 0 0 0 0 -1 -1 -1 -1
4 Reduced : B 8 0 -8
```

That is because the coefficients for the “B”, after appropriate duplication, have a correlation of 1 with the other terms in the “Contrast” statement, and each of the other terms in the CONTRAST statement involve the “B” term in some manner or another. To determine the correlation of a CONTRAST component, the two vector components are adjusted to match in length (duplication of the shorter component is performed using the Kronecker product operator, using a 1 vector with as many columns as the other components of the interaction). At that point, we compute:

$$\boldsymbol{VC} = \frac{\boldsymbol{V}_A * \boldsymbol{V}'_B}{\sqrt{\boldsymbol{V}_A * \boldsymbol{V}'_A * \boldsymbol{V}_B * \boldsymbol{V}'_B}}$$

This is a ratio or non-centered correlation. When $\boldsymbol{VC} = 1$, the information in two CONTRAST components are entirely redudant, and the longer component can be eliminated from the CONTRAST statement.

This is both a sensible idea, and one which generates contrasts which behave more properly. In PROC GLM and several other contrasts, the contrast degrees of freedom are determined from the effects listed in the CONTRAST statement. The last term in the CONTRAST statement determines the default degrees of freedom for the CONTRAST test.

CONVERTING CONTRASTS TO MACRO VARIABLES

In converting the final contrast components to macro variables, the following code is used:

```
1  call symput("_ncv",compress(char(ncol(resv&_i))));
2  %do _zi=1 %to &_ncv;
3  call symput("_vf",compress(char(resv&_i[,&_zi])));
4  %let _cx=&_cx &_vf;
5  %end;
```

In Row 1, the number of items in the vector is determined, and converted into a macro variable. In Row 2, the macro variable from Line 1 is used, to process all items in Vector `_resv&_i` in a macro loop. Next, in Row 3, each element is converted into a macro variable. This is done using the "`char`" function, which converts numeric matrices into character values within the SAS/IML procedure. In Row 4, the macro variable for the element is added to the macro variable for Contrast "`_cx`".

This demonstrates the use of macro variables in the SAS/IML environment. The real difference in this environment is the immediate execution of code, in which the SYMPUT statement is evaluated immediately, and the resulting macro variable can be used immediately. This type of code for macro use is different than code in other components of the SAS system.

SPECIFICATION OF CELLS FOR COMPARISON

In setting up the master contrast $\boldsymbol{L}$, the user may specify the contrast as an argument to the macro. This is often a simple approach, but is sometimes difficult to do, especially if there are a large number of factors. An alternative entry method involves passing in the cells for comparison in two sets, the positive set and the negative set. These two sets of cells are used to set up the master contrast $\boldsymbol{L}$. When defining the sets of cells, the wildcard character "`*`" may be used to select all values in a particular level.

MACRO _CONTRGEN

The full macro cannot be printed here, as it exceeds 300 lines (excluding diagnostic code). Thus, rather than presenting the macro, the basic method will be described carefully. The full macro is available from the author via email.

Macro _contrgen

1. Check if _contrgen to generate new contrast or add to old
2. Count number of items in factor list and level count list
3. Determine number of terms in full experiment and generate full list
4. Eliminate any that are to be excluded
   (a) Check for nesting of factors in others - eliminate inconsistent terms
   (b) Eliminate terms if above max interaction count
5. Generate each factor-oriented component of design matrix
6. Generate all requisite interaction design matrix components
7. If design matrix component $\boldsymbol{X}_z$ is included:
   (a) Compute $c_z = \boldsymbol{L}'\boldsymbol{X}_z$
   (b) Determine if c_z has non-zero coefficients
   (c) Determine if c_z is redundant with other components
   (d) If component is still to be included, add to contrast
8. Assign final contrast to output variable, and terminate.

EXAMPLE

Consider a design with two factors B (3 levels) and C (4 levels), set up in a design with an interaction B*C. If we wish to use a contrast comparing A1B2 to A2B1 (that is, with a L vector: 0 1 0 0 -1 0 0 0 0 0 0 0), we would end up with a contrast:

```
/* Macro Call 1: _contrgen ******************************************** */
%_contrgen(_factnm=%str(b\c),
           _factlv=%str(3\4),
           _contrnm=Demo 3,
           _contrast=_conc,
           _help=0,
           _lvector=0 1 0 0 -1 0 0 0 0 0 0 0);
/* Results of Macro Call 1 ******************************************** */
CONTRAST "Demo 3" b 1 -1 0 c -1 1 0 0 b*c 0 1 0 0 -1 0 0 0 0 0 0 0
/* Macro Call 2: _contrgen ******************************************** */
%_contrgen(_factnm=%str(a\b\c),
           _factlv=%str(2\3\4),
           _contrnm=Demo 4,
           _contrast=_cone,
           _help=0,
           _posset=1-2-2\2-2-3,
           _negset=2-2-2\2-2-1\1-2-1
);
/* Results of Macro Call 2 ******************************************** */
CONTRAST "Demo 4" a 1 -1 b -4 4 0 c 1 3 0 -4 a*b -2 3 0 -2 1 0
  a*c 3 0 0 -2 -2 3 0 -2 b*c 0 0 0 -4 1 3 0 0 0 0 0 0
  a*b*c 0 0 0 -2 3 0 0 0 0 0 0 0 0 0 0 -2 -2 3 0 0 0 0 0 0
```

SPECIAL TOPICS

Several special topics can be

Missing cells: Contrast generation for missing cells is a particularly important topic. Missing cells are encountered in a number of contexts, and the macro accounts for them by inserting a place-holder row of 0s in the design matrix, defining the contrast, and then ensuring that the resulting contrast component contain no vestige of the missing cell.

Continuous variables: Continuous variables can be easily included in CONTRAST statements. This is done using two additional macro parameters for the _contrgen statement.

ESTIMATE statements: Estimate statements within SAS have requirements which are virtually identical to those for CONTRAST statements. Thus, the macro allows either a CONTRAST statement, an ESTIMATE statement, or both to be generated.

Random components for PROC MIXED: At this point, only fixed component coefficients within PROC MIXED can be generated by the _contrgen macro. Inclusion of coefficients for random portions of contrasts is under consideration.

CONCLUSION

The capabilities of the linear models procedures within the SAS system are greatly enhanced by contrasts. These allow a large number of more sophisticated questions to be addressed by these procedures. The _contrgen macro generates contrasts from specifications about the comparison of groups of cells. These contrasts are always estimable, since the same method used for testing for estimability is used to generate the contrasts. The development and use of the macro is described, along with several interesting features of the SAS/IML system within the SAS system.

BIBLIOGRAPHY

SAS Institute Inc (2002-2003) SAS Help and Documentation for SAS for Windows, Version 9.1. Cary NC: SAS Institute Inc.

ABOUT THE AUTHOR

Paul A. Thompson was introduced to SAS while attending the University of North Carolina-Chapel Hill in 1975, making him a 30-year SAS user. As a long-term user, he has written many SUGI papers using many of the components of the SAS system. The macro system is featured prominently in many of his uses of the system. Currently, he is Research Associate Professor of Biostatistics, Washington University School of Medicine, St. Louis, MO. In that position, he supports multi-site clinical trials in kidney disease, stroke, rare childhood cancer, psychiatry and other areas using a combination of the `SAS/IntrNet` and `htmSQL` toolsets. He enjoys gardening, wine-making, TEX/LATEX and go.

This manuscript was prepared using LATEX and the `SUGconf.cls` class, obtained from Ron Fehd at the very last minute.

CONTACT INFORMATION

Your comments or suggestions are welcome.

Author Name:	Paul A. Thompson
Company:	Division of Biostatistics
	Washington University School of Medicine
Address:	660 S. Euclid, CB 8067
	St. Louis, MO 63110-1093
Telephone:	314-747-3793
E-mail:	paul@wubios.wustl.edu
Website:	www.biostat.wustl.edu/ paul

Copies of Macro `_contrgen` may be obtained by request to the author via email.