“Dualing” Arrays: Reducing DATA Step Passes with Array Look-Ups

Nancy Brucken, STATPROBE, Inc., Ann Arbor, MI
Lisa R. Reeves, STATPROBE, Inc., Ann Arbor, MI

BACKGROUND

As part of an analysis of the effect of treatment group and subject order in study entry within site (subject number) on change in certain laboratory values, we were asked to provide pairwise comparisons of treatment effects by subject number. With the Output Delivery System (ODS), we were able to retrieve the required p-values, and store them in a SAS® dataset. The problem then became one of extracting the p-values for the desired comparisons. Using the DATA step in a non-traditional way and building look-up tables with arrays allowed us to identify the LSMEAN numbers for the desired comparisons and then pull off the p-values calculated for those comparisons in a single DATA step, with only one pass through each of the associated datasets.

STATISTICAL METHODOLOGY

The primary objective of the study under consideration was to examine the change in a particular lab parameter from baseline to the last valid visit. This parameter was to be compared among four treatment arms using an analysis of covariance (ANCOVA) model, including as covariates the (centered) baseline value of the lab parameter and the subject order in study entry within site (first, second, third, fourth, fifth or later). Treatment-by-baseline and treatment-by-subject number interaction terms were also entered into the model. If the treatment-by-baseline interaction was not significant at the 0.10 level, it was removed from the model; the treatment-by-subject number interaction remained in the model as a nuisance parameter regardless of its significance.

Pairwise treatment comparisons were based on treatment least square means (ie, adjusted mean responses) from the final model. Least square means were estimated for each treatment group overall, and for each treatment group within subject number (ie, among patients enrolled first, among patients enrolled second, and so on).

The following code enabled us to fit the full model using PROC GLM, and store the desired output sections in SAS datasets:
The treatment least square means went into the LSMEANS dataset, while p-values corresponding to every possible pairwise comparison between treatment least square means went into the DIFFMAT dataset.

**NOW THE FUN BEGINS**

We had to provide p-values for pairwise comparisons of treatment groups separately for each subject number (1-5), and across all subject numbers. The all-subjects case was simpler, so we tackled that one first. The initial step was to determine the LSMEANS number for each treatment group. Once we had that, we could go into the p-values dataset, and extract the necessary p-values between pairs of treatment groups. When done manually, the process went something like this:

Desired table of p-values:

<table>
<thead>
<tr>
<th></th>
<th>Group 1 vs. Group 2</th>
<th>Group 1 vs. Group 3</th>
<th>Group 1 vs. Group 4</th>
<th>Group 2 vs. Group 3</th>
<th>Group 2 vs. Group 4</th>
<th>Group 3 vs. Group 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Subjects</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
</tr>
<tr>
<td>First Subject</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
</tr>
<tr>
<td>Second Subject</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
<td>XXXXX</td>
</tr>
</tbody>
</table>
Figure 1.

The GLM Procedure
Least Squares Means

| RXGRP | A1CCHG | Standard Error | Pr > |t| | LSMEAN | Number |
|-------|--------|----------------|------|---|--------|--------|
| 1     | -1.49209350 | 0.02975621 | <.0001 | 1 |
| 2     | -1.33895871 | 0.02973217 | <.0001 | 2 |
| 3     | -1.53049898 | 0.02874815 | <.0001 | 3 |
| 4     | -1.30197294 | 0.02834799 | <.0001 | 4 |

Figure 2.

Least Squares Means for effect RXGRP
Pr > |t| for H0: LSMean(i)=LSMean(j)

Dependent Variable: AICGHG

<table>
<thead>
<tr>
<th>i/j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0003</td>
<td>0.3533</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.0003</td>
<td>&lt;.0001</td>
<td>0.3680</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3533</td>
<td>&lt;.0001</td>
<td>&lt;.0001</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>&lt;.0001</td>
<td>0.3680</td>
<td>&lt;.0001</td>
<td></td>
</tr>
</tbody>
</table>

Resulting table of p-values:

<table>
<thead>
<tr>
<th></th>
<th>Group 1 vs. Group 2</th>
<th>Group 1 vs. Group 3</th>
<th>Group 1 vs. Group 4</th>
<th>Group 2 vs. Group 3</th>
<th>Group 2 vs. Group 4</th>
<th>Group 3 vs. Group 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Subjects</td>
<td><strong>0.0003</strong></td>
<td>0.3533</td>
<td>&lt;0.0001</td>
<td>&lt;0.0001</td>
<td>0.3680</td>
<td>&lt;0.0001</td>
</tr>
</tbody>
</table>

As you can see, a matrix of p-values was generated, and the challenge lay in identifying the desired p-values.

We decided to set up two arrays in a DATA step. The first array would hold the treatment groups and corresponding LSMEAN values. The second array would hold the matrix of p-values. The code would then read in a dataset containing the planned comparisons, and go through the arrays to pick off the corresponding p-values. To code this efficiently, we decided to use a powerful coding technique known as the Whitlock-Dorfman DO ("DOW") loop. This tool enabled us to code the entire process in a single DATA step, and required only a single pass through each of the two datasets. The code is as follows:
*** Create all subjects row of pairwise comparisons table;
data allsubs (keep=order raw_p);
array row(4) _1 _2 _3 _4;
array pvals(4,4) _temporary_;

*** Set up array of all LS means p-values;
do until (eof);
set diffmat end=eof;
do j=1 to dim(row);
pvals(rowname, j) = row(j);
end;
end;

*** Pull off p-value for each comparison and output each on a separate row;
do until (eofrx);
set rxcomp end=eofrx;
order = 0;
raw_p = pvals(rxgrp1,rxgrp2);
output;
end;
run;

The DOW-loop gives the programmer complete control of the implicit DATA step DO loop, and in particular, control over when the Program Data Vector (PDV) is re-initialized. In this DATA step, we first defined two arrays. ROW contained the current row of the p-value matrix read in from the DIFFMAT dataset created from PROC GLM. PVALS, after all of the rows in DIFFMAT had been read in, contained the complete matrix of p-values. So, at the end of the first DO-loop, the entire contents of the DIFFMAT dataset had been loaded into an array. Since arrays are stored in memory, retrieval of the desired p-values would be extremely fast.

The next DO-loop read in the RXCOMP dataset, which we had created to hold the desired pairs of treatment groups to be compared. These were stored in variables RXGRP1 and RXGRP2. The DATA step then went directly into the PVALS array to extract the element representing the p-value corresponding to the indicated pair of treatment groups, and then output the p-value and subject number (ORDER) to a dataset. Note that the entire DATA step code is executed only once, and only one pass is made through both the DIFFMAT and RXCOMP datasets.

BUT I HAVE TWO PARAMETERS TO WORRY ABOUT

The process described above yielded the values populating the first row of the desired table. However, we also needed to obtain p-values by subject number for each pairwise treatment group comparison. In the previous example, the treatment group and LSMEAN numbers were the same, effectively enabling us to skip that look-up step in the code. Now that we have two parameters, subject number and treatment group, that is no longer the case.
Since the output dataset containing the LSMEAN values for each combination of subject number and treatment group is one-dimensional, we decided the easiest way to convert it to a look-up table was to make it a format. It actually became an informat, so we could take the subject number/treatment group combination and convert it to a number. The following code builds an input control dataset into an informat, and then uses that informat to obtain the LSMEAN number for each desired comparison. RXCOMP is the dataset containing the list of desired comparisons.

``` Sas
*** Create look-up table from LSMEANS output dataset;
data lsmeanft;
  set lsmeans2;
  length start $ 3;
  retain fmtname 'lsmean' type 'j';
  start = trim(left(put(rxgrp, 1.))) || '_' || trim(left(put(order, 1.)));
  label = lsmeannu;
run;

proc format cntlin=lsmeanft;
run;

*** Determine appropriate LSMEANS pairs for each treatment group comparison
*** and subject number;
data rxordcml;
  set rxordcmp;
  length rxord1 rxord2 $ 3;
  rxord1 = trim(left(put(rxgrp1, 1.))) || '_' ||
           trim(left(put(order, 1.)));
  lsmean1 = input(rxord1, $lsmean.);
  rxord2 = trim(left(put(rxgrp2, 1.))) || '_' ||
           trim(left(put(order, 1.)));
  lsmean2 = input(rxord2, $lsmean.);
run;

proc sort data=rxordcml;
  by order sortord;
run;
```

The process then became identical to the previous simpler case. We simply loaded the entire p-value matrix into a 2-dimensional array, read in the dataset containing the desired comparisons and corresponding LSMEANS values, and then used those LSMEANS values as array indices for directly accessing the correct p-value. The code is as follows:
*** Create dataset of pvalues for remaining rows of pairwise comparisons
*** table;
data compbyord (keep=order raw_p);
  array row(20) _1-_20;
  array pvals(20,20) _temporary_;  
  array compare(6) comp1-comp6;

*** Set up array of all LS means p-values;
do until (eof);
  set diffmat2 end=eof;
  do j=1 to dim(row);
    pvals(rowname, j) = row(j);
  end;
end;

*** Pull off p-value for each comparison and output each on a separate
*** record;
do until (eofrx);
  set rxordcml end=eofrx;
  by order sortord;
  raw_p = pvals(lsmean1,lsmean2);
  output;
end;
run;

Again note that the entire DATA step executes only once, and that only one pass is made through each of the DIFFMAT2 and RXORDCML datasets.

**SO WHAT? THESE ARE SMALL DATASETS**

That is true. The p-value matrix in the second example is only a 20x20 array, and there are only 30 planned comparisons, so the volume of data is tiny. However, the same techniques can be applied to large datasets. SAS performs array look-ups extremely quickly, and with the memory capacity of modern computers, two-dimensional arrays with thousands of elements can be loaded and accessed in far less time than it would take to perform DATA step merges or PROC SQL joins between datasets of that size.

The key to taking advantage of that processing speed is to load the array from a dataset and perform the look-up using another dataset within the same DATA step, because arrays do not persist across DATA step boundaries. Minimizing the number of passes through the data also plays a big part in determining how long a program takes to execute. As we have just shown, the DOW-loop allows the loading and look-up to take place all within a single pass through each of two datasets.
REFERENCES


Ian Whitlock and Paul Dorfman, among others, have posted numerous times on the DOW-loop on the SAS-L listserv / Usenet newsgroup (comp.soft-sys.sas).

SAS is a registered trademark of SAS, Inc., Cary, NC.

CONTACT INFORMATION

Nancy Brucken
STATPROBE, Inc.
5430 Data Court, Suite 200
Ann Arbor, MI 48108
(734) 769-5000 x1231
Nancy.Brucken@statprobe.com

Lisa Reeves
STATPROBE, Inc.
5430 Data Court, Suite 200
Ann Arbor, MI 48108
(734) 769-5000 x1234
Lisa.Reeves@statprobe.com