Paper 106-26

Comparing Pairs: The Array's the Thing

Mary MacDougall, SPS Software Services, Inc., Canton, OH

ABSTRACT

As SAS® programmers, we are accustomed to working with data from top to bottom, one row at a time. However, when we need to investigate the relationships between multiple events, it may be preferable to have all the data about those events available at the same time. This paper describes an example of a manufacturer that needs to compare the defect rates of several different machines over time. The solution, which is applicable to all platforms and can be adapted to solve similar comparison problems, involves using base SAS tools to summarize and restructure the data. The array's the thing that enables us to move back and forth across the data, examining each different pairing of machines.

INTRODUCTION

A woolen mill weaves blankets on four different looms, and every blanket is tested against three different quality criteria. The percentage of a loom's output that does not meet the quality criteria is called its defect rate. Our task is to produce a data set containing a comparison of the defect rates for each loom to the defect rates of each of the other looms, by type of defect and week. The solution will involve using multi-dimensional arrays and nested DO loops.

The source data set contains records for four looms over twelve weeks. There is one row for each blanket, with a character identification code, numeric variables that specify the week and loom, and three numeric variables with values of 1 or 0, indicating whether the blanket had a particular type of defect or not. A sample of the source data is shown in Figure 1.

RESTRUCTURING THE DATA

In order to compare each loom to every other loom, it would be helpful to restructure the data so that all the information about a week is contained in a single row. The first step is to sort the data by Week and Loom.

```
PROC SORT data=Blankets;
  by Week Loom;
run;
```

Next, we create three arrays to help us refer to the data in a systematic way. Def is a one-dimensional array that refers to the three different types of defects. TotCount is a one-dimensional array that will be used to count the total number of blankets produced on each loom. DefSum is a two-dimensional array that will be used to sum the number of imperfect blankets by defect type and loom. DefSum refers to a set of new variables D1_L1 through D3_L4.

```
DATA WeekSum;
  SET Blankets;
  by Week Loom;
  array Def {3} DefType1 - DefType3;
  array TotCount {4} Tot1 - Tot4;
  array DefSum {3,4} D1_L1 - D1_L4
                     D2_L1 - D2_L4
                     D3_L1 - D3_L4;
```

A DO loop initializes the TotCount array and resets the values of Tot1 through Tot4 to zero with every new Week.

```
  if first.Week then do;
     DO i = 1 to 4;
        TotCount{i} = 0;
     end;
  end;
  TotCount{loom} + 1;
```

We use another DO loop to count the occurrences of each of the three types of defects and store the sums in the appropriate member of the DefSum array. For example, on the first iteration of the DO loop, the sum of type 1 defects on loom 1 is stored in variable D1_L1. On the next iteration, the sum of type 2 defects on loom 1 is stored in D2_L1. At the end of each week's data, one record is output that contains a summary of the total blankets by loom and defects by type and loom for that week.

```
  DO d = 1 to 3;
    if first.Loom then DefSum{d,Loom} = 0;
    DefSum{d,Loom} + Def{d};
  end;
  if last.Week then output;
  keep Week Tot1 - Tot4
       D1_L1 - D1_L4
       D2_L1 - D2_L4
       D3_L1 - D3_L4;
run;
```

At the end of the data step, WeekSum contains defect and total counts by week for each loom and defect type, as shown in Figure 2.

COMPARING EACH PAIR

In the next data step, we again use the DefSum and TotCount arrays to help us refer to the data.

```
DATA Pairs;
  set WeekSum;
  by Week;
  array TotCount{4} Tot1 - Tot4;
  array DefSum{3,4} D1_L1 - D1_L4
                    D2_L1 - D2_L4
                    D3_L1 - D3_L4;
```

We use three nested DO loops to move through the TotCount and DefSum arrays. The DefType loop cycles through each of the defect types. The Loom loop and the Compare loop calculate the difference in defect rates for each combination of looms. The variable Comp keeps track of which loom is being compared. It is the critical element that enables us to move across the DefSum array, examining each pairing of looms.

The first iteration through the DO loops creates a character variable called Pairing that identifies the two looms being compared. Dif is calculated as the difference between the percentage of type 1 defects on the first loom and the percentage of type 1 defects on the second loom. A new record is then output containing all the needed information about this pairing: defect type, week number, a code to identify the two looms being compared, and the difference between the two defect rates.

We increment Comp by one so that the second iteration of the Compare loop will compare the first loom to the third one. This process repeats until there are no more looms to be compared to the first loom.

On the second iteration of the Loom loop, the defect rate on loom 2 is compared to the rate on loom 3. After all the pairings of looms have been examined for defect type 1, the DefType loop iterates and repeats the process for the other two types of defects.

```
   do DefType = 1 to 3; /* Begin DefType loop */
     do Loom = 1 to 4;     /* Begin Loom loop */
       Comp = Loom + 1;  /* Begin Compare loop*/
       do while (Comp < 5);
         Pairing = put(Loom,1.)||'_'||
                   put(Comp,1.);
         if tot{Loom} > 0 and tot{Comp} > 0 then
           Dif = DefSum{DefType,Loom}/tot{Loom}
               - DefSum{DefType,Comp}/tot{Comp};
         else Dif = 0;
         output;
         Comp + 1;
       end;                /* End Compare loop */
     end;                     /* End Loom loop */
   end;                    /* End DefType loop  */
run;
```

The resulting data set contains all the data we need, but it is not in the order we want, so a SORT procedure completes the solution.

```
   PROC SORT data=pairs;
     by DefType Pairing Week;
   run;
```

Figure 3 shows a portion of the final data set, which contains the difference between the defect rates of each pair of looms, by type of defect and week.

CONCLUSION

This example illustrates how to transpose rows of data about events into columns, and then use arrays to compare each combination of events. Note that the size of the output data set depends on the number of distinct combinations to be compared, not on the size of the source data set. In this example, twelve rows of summarized data were expanded into 216 rows of output (3 types of defects x 12 weeks x 6 loom pairings). If your problem will produce a very large number of comparisons, you may want to structure your problem so that the output is more manageable.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mary MacDougall
SPS Software Services, Inc.
3142 21st St. N.W.
Canton, OH 44709
Work Phone: 330-454-4241
Fax: 330-454-4468
Email: marymacd@spssoft.com
Web: www.spssoft.com

APPENDIX

Obs	Week	Loom	ID	DefType1	DefType2	DefType3
1	1	1	L01-B0001	0	0	1
2	1	1	L01-B0002	0	1	0
3	1	1	L01-B0003	0	0	0
...						
4798	12	4	L04-B4798	0	0	1
4799	12	4	L04-B4799	0	0	0
4800	12	4	L04-B4800	0	0	0

Figure 1. Sample of source data

Obs	Week	D1_L1	D1_L2	D1_L3	D1_L4	D2_L1	D2_L2	D2_L3	D2_L4	D3_L1	D3_L2	D3_L3	D3_L4	TOT1	TOT2	TOT3	TOT4
1	1	0	0	2	0	3	0	0	0	1	0	1	0	100	100	100	100
2	2	1	0	3	1	6	0	3	1	2	0	0	1	100	100	100	100
3	3	1	0	1	1	0	1	1	1	1	0	1	1	100	100	100	100
4	4	3	0	1	3	0	0	1	1	0	0	0	4	100	100	100	100
5	5	0	0	2	1	0	0	0	2	0	0	0	0	100	100	100	100
6	6	1	1	0	1	2	0	0	1	0	1	1	0	100	100	100	100
7	7	1	1	0	0	1	0	0	1	2	1	0	0	100	100	100	100
8	8	1	2	1	0	0	1	0	0	2	0	0	0	100	100	100	100
9	9	0	0	0	0	0	1	0	0	0	1	1	0	100	100	100	100
10	10	2	0	0	0	0	1	0	0	0	2	0	1	100	100	100	100
11	11	4	0	0	0	2	1	0	0	1	0	1	0	100	100	100	100
12	12	0	1	1	0	1	1	0	0	0	2	0	1	100	100	100	100

Figure 2. Counts summed by week

Obs	DefType	Pairing	Week	Dif
1	1	1_2	1	0.00
2	1	1_2	2	0.01
3	1	1_2	3	0.01
4	1	1_2	4	0.03
5	1	1_2	5	0.00
6	1	1_2	6	0.00
...				
211	3	3_4	7	0.00
212	3	3_4	8	0.00
213	3	3_4	9	0.01
214	3	3_4	10	-0.01
215	3	3_4	11	0.01
216	3	3_4	12	-0.01

Figure 3. Comparison of defect rates for each pair of looms, by type of defect and week