Reading Unbalanced, Unstructured Data with SAS Software

Ronald P. Cody
University of Medicine and Dentistry of New Jersey
Rutgers Medical School

Abstract - A method of entering and reading data is proposed that is easier and less prone to error than traditional methods for certain classes of problems such as unbalanced designs. Program techniques are developed to process these data as stream input with imbedded "tags." Examples of SAS programs using traditional techniques and the proposed "stream" technique are developed.

Introduction

Data that are unbalanced and unstructured can cause problems for the SAS programmer. For example, when several types of data with variable number of observations or records are to be read, the traditional method is to supply the program (in the data step) with enough information to read and interpret the data correctly.

In some applications, it is easier to imbed information with the data itself in the form of "tags," that instruct the program about the nature of the data. A t-test example with unequal n's and an unbalanced ANOVA problem will serve to illustrate the use of tags and stream data input.

Example 1 - Unbalanced T-test.

The amount and complexity of the data have been reduced to make the examples short and easy to follow. The techniques discussed have their strength with larger, more complicated data sets.

Our first example is taken from an experiment where we had 5 control and 3 treatment subjects and we recorded a single variable per subject. The data are shown below:

<table>
<thead>
<tr>
<th>GROUP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>25</td>
<td>42</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>TREATMENT</td>
<td></td>
</tr>
</tbody>
</table>

The simplest, most straightforward method to read these data is shown below:

*DATA SET EX1A ON ACCOUNT LA00004
SAS EXAMPLES FOR T-TEST
EXAMPLE WITH UNBALANCED GROUPS
RON CODY - AUGUST 1984*

DATA EX1A;
INPUT GROUP $ X @@;
CARDS;
C 20 C 25 C 23 C 27 C 30
T 40 T 42 T 35
PROC PRINT;
TITLE METHOD A TEST;
Example 1-A

For large amounts of data, this program has some problems. It is tedious and time consuming to repeat the group identification before each variable to be read. This can be corrected in two ways: First, we can put the information concerning the number of observations per group in the program (Example 1-B) or we can put this information in the data itself (Example 1-C). For this very small, simple example, any of these methods would suffice. However, if the number of observations were large (several hundred), or there were more groups and variables, a single mistake in counting would have disastrous results.

DATA EX1B;
GROUP='C';
DO I=1 TO 5;
   INPUT X @@;
   OUTPUT;
END;
GROUP='T';
DO I=1 TO 3;
   INPUT X @@;
   OUTPUT;
END;
DROP I;
CARDS;
20 25 23 27 30
40 42 35
PROC PRINT;
TITLE METHOD B TEST;
Example 1-B
DATA EX1C;
DO GROUP = 'C','T';
   INPUT N;
   DO I=1 TO N;
      INPUT X @@;
      OUTPUT;
   END;
END;
DROP N I;
CARDS;
  5
  20 25 23 27 30
  3
  40 42 35
PROC PRINT;
TITLE METHOD C TEST;
Example 1-C

The approach were are suggesting is shown in example 1-D below:

DATA EX1D;
RETAIN GROUP;
INPUT TEST $ @@;
IF VERIFY (TEST, 'CT ') = 0 THEN DO;
   GROUP = TEST;
   RETURN;
END;
ELSE DO;
   X = INPUT (TEST,S.O);
   OUTPUT;
END;
DROP TEST;
CARDS;
  C 20 25 23 T 40 42
  C 30 T 35
PROC PRINT;
TITLE METHOD D TEST;
Example 1-D

Example 2 - Unbalanced Two-way ANOVA.
The next example will be an unbalanced design for which we want to perform an analysis of variance. Our design is as follows:

<table>
<thead>
<tr>
<th>GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>72</td>
</tr>
</tbody>
</table>

DATA FOR EXAMPLE 2

The straightforward method of entering these data would be:

DATA;
INPUT GROUP $ SEX $ SCORE;
CARDS;
A M 20
A M 30
A F 45
A F 30
etc.

This is a lengthy and wasteful data entry method. For small data set of this type, we could follow the example of the unbalanced t-test problem and enter the number of observations per cell, either in the program or imbedded in the data. A preferable method, especially for large number of observations per cell where counting would be inconvenient, is shown in the example below:

*DATA SET EX2A ON ACCOUNT LA00004;
DATA EX2A;
DO SEX='M','F';
   DO GROUP='A','B','C';
      INPUT TEST $ @@;
      DO WHILE (TESTNE '#');
         SCORE=INPUT(TEST,6.);
         OUTPUT;
      INPUT TEST $ @@;
   END;
END;
END;
DROP TEST;
CARDS;
  20 30 40 20 50 # 70 80 90
  30 # 90 80 90 # 25 30 45 30
  65 72 # 70 90 90 80 85 # 20 20 30 #
PROC PRINT;
etc.
This program reads and assigns observations to a cell until a "#" is read in the data stream. The program then finishes the innermost loop and the next cell is selected. We can read as many lines as necessary for the observations from a given cell.

An improved version of this program is shown next. With this program, we can read the cells in any order and do not have to supply the program with the cell identification since it is incorporated right in the tags. Let's look over the program first, and then we will discuss the salient points:

```
*DATA SET UNBAL3 ON ACCOUNT LA00004;
DATA EX2B;
RETAIN GROUP SEX;
INPUT TEST $ @@;
IF VERIFY (TEST,'ABCMF ') = 0 THEN DO;
  GROUP = SUBSTR (TEST,1,1);
  SEX = SUBSTR (TEST,2,1);
  DELETE;
  RETURN;
END;
SCORE = INPUT (TEST,6.);
DROP TEST;
CARDS;
AM 20 30 40 20 50
BM 70 80 90
CM 90 80 80 90
AF 25 30 45 30 65 72
BF 70 90 90 80 85
CF 20 20 30
PROC PRINT;
etc.
```

This program allows us to enter the cells in any order and even use as many lines as necessary for the observations from a cell. This form of data entry is also convenient when we will be adding more data at a later time. The analysis can be rerun without any changes to the program. Additional observations for a single cell can even be added at the end of the original data.

This program uses the VERIFY function which will return a 0 if all the characters of the variable TEST can be found as one of the character variables in the second argument of the function. Note that a blank is included in argument 2 since the length of TEST is, by default, equal to 8 bytes which means that it will contain one alphabetic letter (A, B, C, M, or F) and seven blanks, when a cell identifier is read. The SUBSTR function picks off the GROUP and SEX values from the TEST string and the INPUT function converts all character values which are really numeric, to numeric format.