Examples of data manipulation in SAS: Filling in "known" missing data

Chong Ho Yu, Ph.D.
Arizona State University, Tempe AZ 85287

ABSTRACT

On many occasions analysts encounter the problem of missing data. The objective of this presentation is not to address issues associated with "unknown" missing data; rather, the focus centers on "known" missing data that become a hindrance to data processing. Consider this example: In an exam, item responses are stored in the format that each response of each respondent occupies a row (tall structure). When some subjects skip items, these item responses should be considered wrong answers. However, due to a design flaw, these rows are omitted rather than being stored as a blank response. Consider another example: In a study when the same subject re-takes the same exam, constant values of the same subject should be inherited from the previous records. However, for some reason, this inheritance function was not turned on in the database. As a remedy, this presentation demonstrates two SAS programs for auto-filling in missing data based on known values.

INTRODUCTION

On many occasions analysts encounter ill-structured data, such as missing data. The objective of this presentation is not to address issues associated with "unknown" missing data; rather, the focus centers on "known" missing data that become a hindrance to data processing. One common source of error is use (or misuse) of the tall structure. Another problem is the omission of inheritance from the previous records.

TALL STRUCTURE VS. WIDE STRUCTURE

In most cases, a data set is structured as a R X C matrix, in which each row of data represents a subject's responses to all items, and each column represents the values of an item filled in by all subjects. This configuration of data is known as the wide structure. Table 1(a) is a typical example. In the wide structure, usually it is easy to spot missing data. In this example, subject 2 did not answer Item 2 and Item 4, even though no code for missing data (e.g. "." or "9") is used.

Table 1(a). R X C matrix in the wide structure (b) The wide structured data set in text format

(a)

ID	Item 1	Item 2	Item 3	Item 4	Item 5
Subject 1	1	2	3	3	2
Subject 2	1		1		2
Subject 3	1	2	1	3	2
Subject 4	2	2	3	3	2
Subject 5	1	2	3	3	2
Subject 6	1	1	3	3	2
Subject 7	1	2	3	3	2
Subject 8	2	2	3	4	4
Subject 9	1	2	3	4	4
Subject 10	1	2	3	3	2

(b)

```
12332
1 1 2
12132
22332
12332
11332
12332
22344
12344
12332
```

Table 1(b) is a typical example of a wide structured date set in text format. However, sometimes the item responses may be more than one digit or one character, such as responses to multiple-choice-multiple-answer items (e.g. check all that apply) or open-ended questions. If the tall structured data are stored in a spreadsheet, as what is seen in Table 1(a), then this is not a problem. But when the data are formatted as a text file as what you see in Table 1(b), then it is impossible for one column to store more than one digit or one character. When a long string appears in the item response, a tall structure is necessary. Table 2 is an example of a tall-structured data set:

Table 2.. Wide structured data set for storing long item responses

Subject 1	Item 1 (Check all that apply	124
Subject 1	Item 2 (Short answer)	SAS is the best!

Usually the tall structured data are raw responses, which may be useful for distracter analysis (to examine what type of subjects choose which options) and qualitative analysis (to code the answers of open-ended questions into different categories). For these two types of analysis, missing data are not detrimental. However, problems develop when the responses in the tall structured data set are scored and then transposed to a wide structured data set. If there are missing data, this transposition will yield inaccurate data.

Table 3.. A wide-structured data set in text format transformed from a tall structured data set

```
5084846 11111111111011101111
5019469 1110111111
5019448 11011111111011111101
5019498 1111111111101101
5080307 11101111111111011101
5019361 11111110101011011101
5080612 1111111010101101101
5019424 10011110011111111101
5019378 1111110011101101101
5080611 1111111010011101100
```

In Table 3, subject 2 did not answer half of the items in the test, and subject 4 omitted four items. If this had been a speed test, it is possible that the examinees ran out of time and thus were unable to finish the test. But if there was no time limit for the test, it is unlikely that both subjects only skipped the items near the end. When you compare the preceding matrix with the tall structure, you may find that the missing responses scatter all over the exam. However, the tall structure did not capture the non-response; thus, rows for missing data never existed. During the transposition, the blanks in the middle are “pushed” by other item responses. As a result, it appears that both test takers did not answer the items near the end. Fortunately, you can find out which data are missing by comparing the wide structure with the tall structure. However, this is a time-consuming process that requires fixing all of the records by hand. In the following, a SAS program will be introduced to remediate this problem of “known missing data” in an efficient fashion. For simplicity of the illustration, only three subjects and five items are included in the following example (Table 4).

Table 4(a). A tall structure with missing data; (b) A wide structure with coded missing data.

(a)

User ID	Item ID	Score
123	Q1	4
123	Q2	2
123	Q3	3
123	Q4	4
124	Q1	3
124	Q2	4
124	Q4	4
125	Q2	2
125	Q3	1
125	Q4	4

(b)

User ID	Q1	Q2	Q3	Q4	Q5
123	4	2	3	4	.
124	3	4	.	4	.
125	.	2	1	4	.

Table 4(a) depicts a data set consisting of five items. Question 5 is skipped by all examinees while the other items are skipped by only some of the examinees. It is easy to manually convert 5(a) to 5(b) when there are only five items and three subjects. However, this process will be extremely tedious when there are 50 items and 3,000 examinees. Therefore, an automated method is introduced.

In the tall structure, the ID of each person will be duplicated n times, where n is the number of items. To automate the clean-up process, first you need to create a file to retain unique IDs only:

```
data idfile; set rawdata;
    proc sort nodupkey; by userid; run;
```

Second, create another file to make each subject carry 5 rows (items). In this example I use five items only. Please replace the number with your own number of items. This array will assign a fake ID (FID), which is just the ascending order (1-5), to each row:

```
data idonly; set idfile;

    array XX(5) X1-X5;
        do fid=1 to 5;
        x=xx[fid];
        output;
    end;

    drop x1-x5 x itemid rawscore;
    proc sort; by fid; run;
```

Next, sort the raw data for the merging procedure in a later stage:

```
data score; set rawdata;
    proc sort; by userid itemid; run;
```

Then, import a file listing the FID and all item IDs, including those that no one answered. Merge the ID file and item file so that every subject carries five items regardless of whether they have missing data.

```
data item; set formitem;
    proc sort; by fid; run;

data iditem; merge idonly item; by fid;
    proc sort; by userid itemid; run;
```

After the raw data and the iditem file are merged, missing values should also show up. The FID can be used to check whether every subject has exactly five items.

```
data realdata; merge score iditem; by userid itemid; run;
    proc sort; by userid itemid; run;
```

Last, transpose the data to make each row represents a subject with answered and non-answered items:

```
data transpose; set realdata; by userid itemid;
   length i1-i5 $8;

   array scores s1-s5;
   array inames $ i1-i5;
      retain s1-s5 i1-i5 n;
      if first.userid then n = 1;
      scores(n) = rawscore;
      inames(n) = itemid;
      n = n+1;
      if last.userid then output;
run;
```

Table 5.. Repeated measures data without inheriting repeated values

username	age	attempt	score
Alex	29	1	100
Alex		2	98
Alex		3	97
Alex		4	100
Jody	27	1	99
Jody		2	78
Jody		3	100
Jody		4	97
Jody		5	96

OMISSION OF INHERITANCE

Now consider another case: You collected data yielded from a repeated measures design, in which the examinees took the same tests several times. However, the data entry person only entered the age of the subject during the first attempt into the database. You would like to fill in the missing cells of the field "age" according to the value of the first instance for each subject. Some subjects took four trials, some took five, and some took even more. This is another case of “known missing data”.

It would be easy to fix the problem by hand if there were only two subjects and nine rows. But in reality the dataset might have more than 1000 rows. Again, this situation necessitates automation. The following macro function is very simple yet it accomplishes this purpose. Let's name the preceding table as data "one" and all manipulation is performed in dataset "two." The lag function inherits the previous value in the field "age" and puts it into the next row and a new variable named "temp." In the next row, if "age" is empty, then its value will be replaced with the one in "temp," which is copied from the previous "age." Please note that the do loop is set to run five times even though you only have four attempts. Actually, the function still works even if you set the loop to run ten times. If you are not sure about the maximum number of attempts, enter a large number in the do loop. The output will look like Table 6:

```
data two ;set one; run;
%macro fillin;
%do i = 1 %to 5;
data two; set two;
temp =lag(age);
if temp NE " " and age = " " then age = temp;
run;
%end;
%mend fillin;
%fillin;
```

Table 6. Repeated measures data with inherited values

username	age	age	score	temp
Alex	29	29	100	
Alex	29		98	29
Alex	29		97	29
Alex	29		100	29
Jody	27	27	99	29
Jody	27		78	27
Jody	27		100	27
Jody	27		97	27
Jody	27		96	27

CONCLUSION

The preceding SAS codes can be customized to handle other problems of a similar nature. Nonetheless, being proactive is always better than reactive. The best defense strategy is an offensive one. Rather than waiting for a disaster to happen and then writing SAS codes to patch the hole, in the beginning, the data analyst should implement error-checking codes to ensure data integrity. For example, if a database storing tall structured data does not capture blank answers, the issue should be addressed before it becomes a problem. Similarly, in a repeated measures design, two tables should be used to separate user demographic information from the test scores. In this way, the subject demographic information can be entered only once and then a one-to-many relationship can be established by a primary key (e.g. subject ID) to join the two tables.

ACKNOWLEDGMENTS

Special thanks to Dr. Thompson Marilyn and Dr. Kristina Kuppanoff for their valuable input in developing the source code.

CONTACT INFORMATION

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

Chong Ho Yu, Ph.D.
Director of Testing, Measurement, Assessment, and Research
Applied Learning Technology Institute
Arizona State University
3S89 Computing Commons
Tempe, AZ 85287-0101
USA
Work Phone: 480-727-0670
Email: chonghoyu@gmail.com
Web: www.creative-wisdom.com