Paper CC-11

Selecting Earliest Occurrence: Watch Your Step

David H. Abbott, Veterans Affairs Health Services Research

ABSTRACT

In processing statistical data, it is often necessary to select the earliest occurrence of some event, e.g., the earliest diagnosis or treatment date or perhaps the earliest purchase or payment. This seems straight-forward enough logically, however, it is all too easy to implement it *not quite right* when using the SAS® data step. That would not be so bad if such errors generated errors or warnings, but for several important errors there is nothing issued to alert the analyst. Rather, the result looks pretty good but is off by a little and that, for the conscientious analyst, is a little too much. We examine successive attempts at a solution for this apparently simple problem and show the errors committed before arriving at an entirely correct solution.

Introduction

When writing SAS data preparation/manipulation programs, the SAS analyst fairly often needs to select the earliest occurrence of some event, e.g., the earliest diagnosis or treatment date or perhaps the earliest purchase or payment. This seems straight-forward enough logically, however, it is all too easy to implement it *not quite right* when using the SAS data step. For several important errors, moreover, there is nothing issued to alert the analyst. Rather, the result looks pretty good but mishandles a small percentage of observations and so the problem will likely not be detected quickly and may propagate to delivered work products (ouch!). We examine successive attempts at a solution for this apparently simple problem and show the errors encountered by our hypothetical SAS data analyst.

A Test Data Set

To illustrate the problems that can result when the solution to this problem is off the mark, here's a test data set:

Obs	patient	dateOf Event
1	101	2010-10-21
2	101	2010-11-08
3	102	.
4	102	A
5	102	2010-07-04
6	102	2010-10-02
7	103	B

Figure 1. Test Data Set for Selecting Earliest Occurrence

Patient 101 represents most basic test case. This patient has two non-missing event dates and 2010-10-21 is the desired result for this patient. Patient 102 has missing values of two types - a regular and a special missing value [Foley 2005]. This patient also has two non-missing values. The date 2010-07-04 is the desired result for patient 102. Patient 103 has only a single observation that is a the special missing value .B. The desired result for this patient is either .B or, if patients without an event date are not of interest, no observation for patient 103 in the resultant data set.

Attempt 1 - Mishandling Missing

The analyst is apt to start with code that simply selects the lowest sorting observation for each patient:

```
DATA GetEarliest;
  SET IncomingDs;  BY patient dateOfEvent;
  IF FIRST.patient THEN OUTPUT;
RUN;
```

This code runs without complaint and selects the proper values of dateOfEvent for patient 101 and 103. However, it selects the regular missing value (i.e., ".") for patient 102 rather than 2010-07-04 - not what is wanted. In a real data set with thousands of observations and a small percentage of missing values for dateOfEvent, this error can be both hard to detect and introduce significant inaccuracies in statistical results.

Attempt 2 – Faulty Fix

Assuming the analyst carefully examines the results, he/she will detect the fault and perhaps conclude that by adding a check for the dateOfEvent being missing the problem can be corrected:

```
DATA GetEarliest;
  SET IncomingDs;  BY patient dateOfEvent;
  IF FIRST.patient and dateOfEvent ne .  THEN OUTPUT;
RUN;
```

However, adding this condition has the effect of producing no output record for a patient with a missing date since the observation following the missing date has FIRST.patient=0. The observation with the missing was the FIRST even if not a useful FIRST! So, patients 101 and 103 are handled correctly but 102 doesn't even appear in the result data set.

Attempt 3 – Better for Most Data Sets

When the missing results are identified (preferably by the analyst ☺) the author may try to filter out the observations with missing dates using a WHERE instead of adding the condition to the IF statement:

```
DATA GetEarliest;
  SET IncomingDs;  BY patient dateOfEvent;
  WHERE dateOfEvent ne .;
  IF FIRST.patient THEN OUTPUT;
RUN;
```

With this code only patient 101 is handled as desired. Patient 103 simply doesn't occur in the result data set, perhaps OK. However, the date of event reported out for 102 is .A instead of 2010-07-04, definitely not OK. The special missing value .A does not test equal to the regular missing value and .A is represented by a hugely negative number and so it is taken as the earliest date.

This solution is actually fine if 1) no special missing values are in the data set and 2) patients with a missing value of dateOfEvent are not of interest. However, it is easy to fix the code to properly handle special missing values …

Attempt 4 – All types of missing handled, but …

Using the SAS MISSING function is the preferred way to deal with all flavors of missing values:

```
DATA GetEarliest;
  SET IncomingDs;  BY patient dateOfOccurrence;
  WHERE not MISSING(dateOfEvent);
```

```
  IF first.patient THEN OUTPUT;
RUN;
```

This solution provides the correct values of dateOfEvent for patients 101 and 102, but patient 103 is still missing from the result data set. To correct this possibly unacceptable behavior, quite a different approach is required.

Attempt 5 – Finally

The different approach is to forego the notion of processing the event dates for a patient from earliest to latest. Rather, simply process them in whatever order (no sorting by dateOfEvent required) and use a retained variable and the MIN() function to keep track of the earliest non-missing date for each patient:

```
DATA GetEarliest;
  RETAIN earliest;
  SET IncomingDs; BY patient;
  IF FIRST.patient THEN DO; earliest=.Z; END;
  IF missing(earliest) and missing(dateOfEvent) THEN DO;
    IF dateOfEvent lt earliest THEN earliest= dateOfEvent;
  END;
  ELSE earliest = min(earliest, dateOfEvent);
  IF last.patient THEN OUTPUT;
  KEEP patient earliest;
RUN;
```

This code follows the typical RETAIN, IF FIRST, IF LAST pattern so often used in reducing many observations in a BY group to a single observation. The four lines of the IF/ELSE could reduce to just the "earliest =" line (no ELSE) if the MIN function handled the flavors of missing better, but it always returns a regular missing when all arguments are some flavor of missing . So, reporting out the right flavor of missing value in the case of all values being missing needs to be handled as a special case. The smallest flavor of missing value is retained until a non-missing value is encountered. And **finally** the code to this "simple" problem is entirely correct for all reasonable input data sets.

The astute reader may notice that a KEEP statement entered the sequence of trial solutions for the first time above. Actually, it would be better if this statement appeared in all trial solutions. Without it, the result data set in all previous steps has values for whatever other variables are on IncomingDs. Might this be a good thing? A bonus? Typically, it is not unless IncomingDs has the property that the value of the patient and the earliest date uniquely determines the values of all the other fields. If two observations have the same patient ID and earliestDate but differing values in one or more of the other variables then which of those differing values gets reported out to the result data set becomes essentially arbitrary and that is never a good thing.

A Different Path

If it seems that solving this problem with a SAS data step is just way too tricky, the reader will be glad to know that SAS provides an appealing alternative – solve the problem with SQL code:

```
PROC SQL;
  CREATE TABLE result AS
  SELECT patient, MIN(dateOfEvent)AS earliest
  FROM work.IncomingDs
  GROUP BY patient;
QUIT;
```

PROC SQL handles missing values and special missing values entirely correctly using the code the analyst would naturally write on the first try. Kudos to the PROC SQL authors! PROC MEANS might also be tried, but it mishandles a case like patient 103 by reporting out a regular missing value.

Lessons Learned

- It is important to get very clear on the behavior required even for apparently “simple” operations.
- FIRST and LAST variables are powerful algorithmic tools, but using them opens several avenues for error that must be avoided.
- If it is possible for special missing values to be in the incoming data, it is essential to handle them correctly and the MISSING function is a big help in doing so.
- PROC SQL provides a very nice solution for this problem.

References

Foley, Malachy J. 2005, “MISSING VALUES: Everything You Ever Wanted to Know” Proceedings of SESUG2005, http://analytics.ncsu.edu/sesug/2005/TU06_05.PDF

ACKNOWLEDGMENTS

The views expressed in this paper are those of the author and do not necessarily reflect the position or policy of the Department of Veterans Affairs or the United States government.

Without the leadership and encouragement of Dr. Dawn Provenzale, director of the Durham Epidemiologic Research and Information Center at the Durham VA Medical Center, this work could not have occurred. She takes a strong interest in fostering many dimensions of excellence in her employees.

CONTACT INFORMATION

Name	David H. Abbott
Enterprise	Center for Health Services Research in Primary Care
Address	Durham Veterans Affairs Medical Center
	HSR&D Service (152)
	508 Fulton St.
City, State ZIP	Durham, NC 27705
Work Phone:	919-286-0411
E-mail:	david.abbott@va.gov