Longitudinal Data Techniques: Looking Across Observations
Ronald Cody, Ed.D., Robert Wood Johnson Medical School, Piscataway, NJ

Introduction

One of the most difficult tasks for a SAS® programmer is to perform operations across multiple observations. For example, you may have a data set of patient visits, with a variable number of visits per patient and the data for each visits stored in a separate observation. Some of the techniques for working with such data include the RETAIN statement, FIRST. and LAST. logical variables, the LAG function, and the use of multiple SET statements (each with a separate FIRSTOBS= option). Other techniques for summarizing data for each patient (such as the number of visits, the mean and median of some variables) involve procedures such as PROC FREQ and PROC MEANS to output summary data sets. We will demonstrate and discuss these techniques in this tutorial.

Using the RETAIN Statement and FIRST. Logical Variables

Let's start by describing a typical longitudinal data set. This data set, called LABS, has from one to four observations per patient, with each observation representing data from a visit to the clinic. Run the program below to create this data set:

```
***DATA STEP TO CREATE LABS;
DATA LABS;
   LENGTH PATNO $ 3;
   INFORMAT DATE DOB MMDDYY10.;
   INFILE DATALINES MISSOVER;
   INPUT PATNO DATE DOB HR SBP DBP;
   FORMAT DATE DOB MMDDYY10.;
DATALINES;
001 10/21/1997 10/21/1946 48 128 74
003 11/11/1998 09/08/1955 52 140 80
123 01/28/1998 01/01/1944 80 180 96
001 11/04/1998 . 52 130 76
001 11/07/1998 . 54 132 78
123 05/04/1998 . 80 178 90
007 04/04/1998
008 03/22/1998 02/08/1980 58 144 72
008 04/21/1998 . 66 144 74
001 02/01/1998 . 44 126 70
354 04/12/1998 07/07/1955 90 210 110
012 05/06/1998 . 80 120 80
```

```
013 11/11/1998 11/09/1930 100 180 108
013 11/18/1998 . 90 170 98
554 06/08/1998 09/12/1944 48 108 66
888 01/01/1998 03/14/1922 46 110 68
;
```

Below is a listing of this data set after we sorted by patient number (PATNO) and date of visit (DATE):

Listing of Data Set LABS

PATNO	DATE	DOB	HR	SBP	DBP
001	10/21/1997	10/21/1946	48	128	74
001	02/01/1998	.	44	126	70
001	11/04/1998	.	52	130	76
001	11/07/1998	.	54	132	78
003	11/11/1998	09/08/1955	52	140	80
007	04/04/1998	.	.	.	.
008	03/22/1998	02/08/1980	58	144	72
008	04/21/1998	.	66	144	74
012	05/06/1998	.	80	120	80
013	11/11/1998	11/09/1930	100	180	108
013	11/18/1998	.	90	170	98
123	01/28/1998	01/01/1944	80	180	96
123	05/04/1998	.	80	178	90
354	04/12/1998	07/07/1955	90	210	110
554	06/08/1998	09/12/1944	48	108	66
888	01/01/1998	03/14/1922	46	110	68

Notice several features of this data set. First, as we mentioned earlier, there are from one to four observations per patient. Next, the date of birth (DOB) is entered only on the first observation for each patient. The variables HR, SBP, and DBP represent the patient's heart rate, systolic blood pressure, and diastolic blood pressure, respectively.

Our first task with this data set is to add the date of birth to the observations corresponding to the second to the last, for each patient. Two very useful tools, the RETAIN statement and the FIRST.PATNO logical variable make this an easy task. Let's first look at the program and we will then explain how it works:

```
PROC SORT DATA=LABS;
   BY PATNO DATE;
RUN;
```

```
DATA LABS2;                              PROC SORT DATA=LABS2;
   SET LABS;                                BY PATNO DATE;
   RETAIN OLDDOB;                         RUN;
   BY PATNO;
   IF FIRST.PATNO THEN OLDDOB = DOB;     DATA FIRST;
   ELSE DOB = OLDDOB;                       SET LABS2;
   DROP OLDDOB;                             BY PATNO;
RUN;                                        IF FIRST.PATNO;
                                         RUN;
```

Remember, your data set must first be sorted before you can use a BY statement. Notice that we chose to sort by visit date (DATE) within patient number (PATNO), so that the visits are in the correct order. The BY statement creates two logical variables, FIRST.PATNO and LAST.PATNO. When we are processing the first observation within the BY group, FIRST.PATNO is true, otherwise it is false. Likewise, LAST.PATNO will be true when we are processing the last observation within the BY group. For those patients with only one visit, both FIRST.PATNO and LAST.PATNO will be true. Now that we know how to determine when we are reading the first observation for each patient, we can set a variable (OLDDOB) equal to the DOB in order to "remember" the value for subsequent observations. However, since the SAS supervisor sets all variables to missing at each iteration of the Data Step, we need a RETAIN statement to prevent this activity from occurring. So, each time we are processing the first visit for a patient, the retained variable OLDDOB is set equal to the date of birth. For all subsequent observations for the same patient, the variable DOB is set to OLDDOB.

Selecting the First (or last) Visit for Each Patient

Selecting the first or last observations for each subject is a frequently needed operation. Again, thanks to the built-in FIRST. and LAST. logical variables, this is easily accomplished. First, let's write a short data step to select the first visit for each patient. We will use the LABS2 data set, created above, which has the date of birth on every observation. Here is the program:

As you can see, this is a very simple program. The key is5
the BY statement following the SET statement, creating the temporary logical variables FIRST.PATNO and LAST.PATNO. Remember that these are logical variables (with values of true or false), so it is not necessary to write:

```
IF FIRST.PATNO = 1;
```

Although that would also work. The IF statement is a subsetting IF, which means that if it is true, an implied OUTPUT is performed and the only observations in data set FIRST will be the first visit for each patient.

Selecting the last visit for each patient is equally easy to do; just replace the variable FIRST.PATNO with LAST.PATNO (and change the name of the data set to something appropriate).

Computing Differences Between Observations

For our next trick, let's find the difference between the heart rate, systolic blood pressure, and diastolic blood pressure, from the first visit to the last visit, not counting patients with only one visit. As you can guess, the FIRST. And LAST. variables will come in handy. We can "remember" the value of the three variables (HR, SBP, and DBP) on the first visit by using RETAINED variables and, when we are processing the data for the last visit, we can subtract the two. Here is the program:

```
DATA DIFFERENCE; /*WHAT? NOT USING
                    VERSION 7!    */
  SET LABS2;
  BY PATNO;
  *REMOVE PATIENTS WITH ONE VISIT;
  IF FIRST.PATNO AND LAST.PATNO
    THEN DELETE;
  RETAIN R_HR R_SBP R_DBP;
  IF FIRST.PATNO THEN DO;
    R_HR = HR;
    R_SBP = SBP;
    R_DBP = DBP;
  END;
  IF LAST.PATNO THEN DO;
    DIFF_HR = HR - R_HR;
    DIFF_SBP = SBP - R_SBP;
    DIFF_DBP = DBP - R_DBP;
    OUTPUT;
  END;
  DROP R_: ;
RUN;
```

Before we explain anything else about this program, we had better explain the DROP statement before you all think this is a typo! A colon following R_, means to include all the variables starting with R_. This is a somewhat obscure, but useful SAS programming feature. There are several similarities between this program and the previous one--they both use FIRST. ,LAST., and retained variables. Remember that for patients with a single observation both FIRST.PATNO and LAST.PATNO are true. We use this fact to delete such observations (remember that following a DELETE statement, the logic returns to the top of the Data Step). Next, we choose three variables to hold the values of HR, SBP, and DBP corresponding to the first visit. Finally, when we are processing the last visit for each patient, we compute the difference scores and output the observation. The new data set will contain a single observation for each patient with the three difference variables.

Adding a Visit Number to Each Observation

It might be convenient to include a visit number in each of the observations in the LABS2 data set. Again, by using the FIRST. and LAST. variables, this is easily accomplished. At the first visit, we need to set the number of visits to one, and at each subsequent visit, we need to increment it. Here is the program:

```
DATA LABS3;
  SET LABS2;
  BY PATNO;
  IF FIRST.PATNO THEN VISIT = 1;
  ELSE VISIT + 1;
RUN;
```

Remember, when we use an expression of the form: VAR + INCREMENT; the variable is automatically retained.   When we run this program, each observation will include a visit number from one to four.

Computing Differences Using the LAG (or DIF) Function

The LAG function should come to mind, when searching for techniques that can be used with longitudinal data. The LAG function returns the value of its argument, the last time the function was executed. Suppose we want to compute the difference in HR, SBP, and DBP from each visit to the next. The LAG function is one straightforward way to accomplish this. Look at the program below:

```
DATA DIFF2;
  SET LABS2;
  BY PATNO;
  DIFF_HR = HR - LAG(HR);
 *ALTERNATIVE: DIFF_HR = DIF(HR);
  DIFF_SBP = SBP - LAG(SBP);
  DIFF_DBP = DBP - LAG(DBP);
  IF NOT FIRST.PATNO THEN OUTPUT;
RUN;
```

You need to take care when using the LAG (or DIF) function. In this program, we execute the LAG function for every observation in the LABS2 data set. When we are processing the first visit for a patient, the difference variables are the difference between the current value of these variables minus the value of these variables for the last visit of the previous patient! Not to worry. We are not outputting an observation corresponding to the first visit.   If you want to include the first visit in your data set

and to have a missing value for the difference variables, you can proceed as follows:

```
DATA DIFF3;
   SET LABS2;
   BY PATNO;
   RETAIN R_HR R_SBP R_DBP;
   R_HR = LAG(HR);
   R_SBP = LAG(SBP);
   R_DBP = LAG(DBP);
   IF NOT FIRST.PATNO THEN DO;
     DIFF_HR = HR - R_HR;
     DIFF_SBP = SBP - R_SBP;
     DIFF_DBP = DBP - R_DBP;
   END;
   DROP R_: ;
RUN;
```

Notice how this differs from the previous program. This time, we output an observation for every visit but only compute the difference variables for the second through last visit. Thus, the difference variables will be missing for the first visit.

Counting the Number of Observations in each BY Group

Suppose you would like to see a listing of all your patient numbers and the number of visits for each. We will solve this problem two ways: one, using a Data Step approach and the other, PROC FREQ.

All we need with a Data Step approach, is to modify the program we used to create a visit number variable. Here it is:

```
DATA COUNT_IT;
   SET LABS2(KEEP=PATNO);
   BY PATNO;
   IF FIRST.PATNO THEN N_VISITS = 1;
   ELSE N_VISITS + 1;
   IF LAST.PATNO THEN OUTPUT;
RUN;
```

Here we decided to use a KEEP= data set option to bring in only the patient number. The only other change to the previous program was to output an observation only when we were processing the last visit.

Using PROC FREQ to Output a Data Set Containing Counts

We will now show how to use PROC FREQ to create a data set containing the number of visits by each patient.

```
PROC FREQ DATA=LABS2 NOPRINT;
   TABLES PATNO /
     OUT=COUNTS(KEEP=PATNO COUNT
          RENAME=(COUNT=N_VISITS));
RUN;
```

The NOPRINT procedure option prevents the procedure from doing its usual outputting to the output window. The TABLES option OUT= creates an output data set of counts. Since we want the data set to look like the Data Step example, we rename the default variable COUNT to N_VISITS. As an alternative to PROC FREQ, we could have used PROC MEANS. We will discuss how PROC MEANS can be used to output a variety of summary statistics, as our next topic.

Creating Summary Data Sets Using PROC MEANS

If we use PROC MEANS, either with a CLASS or BY statement, we can produce an output data set summarizing statistics for each patient. Suppose we want to know the number of visits for each patient, the number of nonmissing values for HR, SBP, and DBP, as well as the patient means for each of these variables. The solution is easy. Here it is:

```
PROC MEANS DATA=LABS2 NWAY NOPRINT;
   CLASS PATNO;
   VAR HR SBP DBP;
   OUTPUT
     OUT  = SUMS(DROP=_TYPE_
            RENAME=(_FREQ_=N_VISITS))
     N    = N_HR N_SBP N_DBP
     MEAN = M_HR M_SBP M_DBP;
RUN;
```

Since we choose to use a CLASS statement instead of a BY statement, it is essential to include the NWAY option so that we only obtain statistics for each patient and not the grand

mean. The NOPRINT option serves the same purpose as it did with PROC FREQ. The output data set is called SUMS. Since _TYPE_ will have a value of one for every observation, it is not useful to us so it is dropped. While we are at it, we rename the _FREQ_ variable to N_VISITS. This variable represents the number of observations (missing or nonmissing) that were used to compute each of the patient statistics. Thus, it represents the number of visits for each patient. The values obtained from the N= statistics request, are the number of nonmissing values of HR, SBP, and DBP for each patient, respectively. The three M_ variables are, or course, the means for the three variables.

What if we wanted to include a median in the summary statistics? Before we had Version 7, it was necessary to use PROC UNIVARIATE. However, with Version 7, the median is one of the statistics that can be computed and added to the output data set. Here is the same program as above, with the three median values added:

```
PROC MEANS DATA=LABS2 NWAY NOPRINT;
  CLASS PATNO;
  VAR HR SBP DBP;
  OUTPUT
    OUT   = SUMS(DROP=_TYPE_
              RENAME=(_FREQ_=N_VISITS))
    N     = N_HR N_SBP N_DBP
    MED   = M_HR M_SBP M_DBP;
RUN;
```

If we use PROC UNIVARIATE to compute the median, we cannot use a CLASS statement and the automatic variable _FREQ_ is no longer available. To overcome this problem (in case you are still using Version 6.12), you need to output the number of missing observations (NMISS=) and, in a subsequent Data Step, add the number of non-missing and missing observations together to compute the total number of visits. Adding the median to PROC MEANS was a terrific upgrade.

Selecting all Patients with "n" Visits

Suppose you had a longitudinal data set and wanted to create a subset of all patients with "n" visits each. There is more than one approach to this problem and we will demonstrate two.

We already know how to create a data set containing the patient number and the number of visits. By merging this with the original data set, using an IN= data set option, we can select the patients we want. Here is the code to select all patients with exactly two visits:

```
PROC FREQ DATA=LABS2 NOPRINT;
  TABLES PATNO /
    OUT=COUNTS(KEEP=PATNO COUNT
         WHERE=(COUNT = 2));
RUN;

DATA TWO_VISITS;
  MERGE COUNTS(IN=IN_COUNT)
        LABS2;
  BY PATNO;
  IF IN_COUNT;
RUN;
```

OK, let's see how this works. First, we add a WHERE= data set option to subset the output data set from PROC FREQ. Since the original data set is sorted, we do not have to sort data set COUNTS. We merge the original data set (LABS2) with the data set containing the patient numbers for patients with two visits. We also use an IN= data set option to create the logical variable IN_COUNT. IN_COUNT is true whenever the data set COUNTS is making a contribution to the current observation being formed by the merge operation. Thus, adding the statement IF IN_COUNT; subsets the final data set to only those patients with two visits.

Instead of PROC FREQ, we can easily use a Data Step to create a data set containing the patient numbers of all patients with two visits. From there, we can merge the two data sets as before. Here is the Data Step approach:

```
DATA TWO;
   SET LABS(KEEP=PATNO);
   BY PATNO;
   IF FIRST.ID THEN N = 1;
   ELSE N + 1;
   IF LAST.ID AND N = 2 THEN OUTPUT;
RUN;
```

We initialize the counter (N) for each new patient and increment it at each visit. When we encounter the last visit for each patient and if the counter has a value of two, we output an observation.

Looking Ahead, Using Multiple SET Statements

While retained variables and LAG functions work great for looking backward, they don't help us if we need to look forward through observations.

Suppose that for each visit, we had the initials of the doctor that was treating the patient. If the same patient came back to the clinic within one month, we want to "credit" that particular doctor with a failure to cure the patient. In order to do this, we need to have access to the present observation for the doctor's identity and the next observation to determine if the same patient came back within the month. We could, alternatively, create a retained variable containing the doctor's ID, but we might want access to a large number of variables associated with the initial patient visit. What we need, is to be able to look ahead to see if the next visits fits the description for a treatment failure.

The "trick" here is to use two SET statements, with one of them employing a FIRSTOBS=2 data set option. This allows us to process the current observation and to "peek" ahead to see values from the next observation. To demonstrate this, we need another data set. Running the short data set below will create a data set called DOC:

```
DATA DOC;
   INPUT @1  PATNO $3.
         @5  VISIT MMDDYY10.
         @16 DOCTOR $3.;
```

```
   FORMAT VISIT MMDDYY10.;
DATALINES;
001 10/21/1998 ABC
001 10/29/1998 XYZ
001 12/12/1998 QED
002 01/01/1998 ABC
003 02/13/1998 QED
003 04/15/1998 MAD
005 05/06/1998 XYZ
005 05/08/1998 QED
;
```

Next, we will use a Data Step with two SET statements to create a data step of treatment failures (any patient that returned with 30 days). Here it is:

```
PROC SORT DATA=DOC;
   BY PATNO VISIT;
RUN;

DATA FAILURES;
   SET DOC;
   BY PATNO;
   SET DOC (FIRSTOBS=2
         KEEP=VISIT
         RENAME=(VISIT=NEXT_VISIT));
   IF NOT LAST.PATNO AND
     (NEXT_VISIT - VISIT) LT 30 THEN
      OUTPUT;
   KEEP PATNO VISIT NEXT_VISIT
         DOCTOR;
RUN;
```

Conclusion

We have demonstrated a few of the very powerful SAS programming tools that allow us to process longitudinal data sets. This is by no means an exhaustive collection of techniques, but it should give you a good start on working with longitudinal data.

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries, ® indicated USA registration.

Ronald P. Cody, Ed.D.
Robert Wood Johnson Medical School
Department of Environmental and Community Medicine
675 Hoes Lane
Piscataway, NJ 08822
(732)235-4490
cody@umdnj.edu