An Introduction to SAS-Lecture 4

Konstantinos Fokianos
University of Cyprus

SAS Arrays

Substituting one value for another in a group of variables.

Suppose that we have 43 variables (X1-X40, A, B and C) in a SAS data and a value of 999 is used to represent missing values.Suppose that you want yo substitute a SAS system missing value (.) for the value 999. Here is one way to do it:

```
DATA LONG;
   SET OLD;    * An old data set
   IF X1   = 999 THEN X1 = .;
   IF X2   = 999 THEN X2 = .;
   .
   .
   IF X40 = 999 THEN X100 = .;
   IF A    = 999 THEN A = .;
   IF B    = 999 THEN B = .;
   IF C    = 999 THEN C =. ;
RUN;
```

SAS Arrays

However, there is a better way to carry out this calculation.

```
DATA EASY;
   SET OLD;  * The old data set
   ARRAY TEST[105] X1-X40 A B C;
   DO I = 1 TO 43;
      IF TEST[I] = 999 THEN TEST[I] = .;
   END;
   DROP I;
RUN;
```

SAS Arrays

- In the previous program you create an array with the ARRAY statement, called TEST and consists of 43 variables.
- By placing an IF statement in a DO loop statement, and operating on the array TEST, you have the same result as in the previous programm. At the end of the loop you drop the index I that you do not need anymore.
- You can always refer to the variables contained in the array. For instance, TEST[1] refers to X1 and TEST[41] refers to A.
- Some other related commands

```
ARRAY TEST[*] X1-X40 A B C;    * SAS counts the number of variables
DO I=1 TO DIM(TEST);           * DIM counts the number of variables i
```

SAS Arrays

Substituting one value for another in all numeric variables

```
DATA NEW;
   SET OLD;
   ARRAY XXX[*] _NUMERIC_;
   DO I = 1 TO DIM (XXX);
      IF XXX[I] = 999 THEN XXX[I] = .;
   END;
   DROP I;
RUN;
```

SAS Arrays

- The internal variable `_NUMERIC_` is used to refer to all the numeric variables in a SAS data set.
- The terms `_CHARACTER_` and `_ALL_` refer to character and all variables, respectively.
- The `DIM` function is useful because it counts the number of numeric variables.
- Using `ARRAY $ XXX[*] _CHARACTER_;` and `IF YYY[I]=’NA’ THEN YYY[I]=’ ’;` we set the values `NA` to blanks for all character variables.

SAS Arrays

Creating multiple observations from a single observation

Suppose that you collect multiple measurements on a subject at different times. Then the `ARRAY` statement gives you a way to restructure your data set.
For instance,

```
Data Set Old
------------------
SUBJECT     X1     X2     X3
1           2       3      4
2           5       6      7
```

SAS Arrays

We want to create the a SAS data set `NEW`, with three observations per subject (one for each measurement), and a variable `TIME` which denotes the measurements (1,2 and 3).

```
Data Set NEW
-----------------------
SUBJECT     TIME        X
1             1         2
1             2         3
1             3         4
2             1         5
2             2         6
2             3         7
```

SAS Arrays

```
DATA NEW;
   SET OLD;
   ARRAY XX[3] X1-X3;
   DO TIME = 1 TO 3;
      X = XX[TIME];
      OUTPUT;
   END;
   DROP X1-X3;
RUN;
```

SAS Arrays

Suppose now that you have another data set but this time you also record the method that was used in addition to time. Suppose further that each record in the data contains the variables X1-X6. Variables X1-X3 represent values taken by method 1 at three different time points. Suppose further that X4–X6 denote the values taken by method 2 at the same time points as before. Our goal is to create a new data set with six observations per subject, one for each time–method combination.

```
Data Set Old
------------------
SUBJECT     X1     X2     X3    X4    X5      X6
1           2       3      4     5     6       7
2           8       9     10    11    12      13
```

SAS Arrays

In other words we want to create the following data set

```
Data Set NEW
-----------------------------
SUBJECT     METHOD TIME        X
1              1      1        2
1              1      2        3
1              1      3        4
1              2      1        5
1              2      2        6
1              2      3        7
2              1      1        8
2              1      2        9
2              1      3        10
2              2      1        11
2              2      2        12
2              2      3        13
```

SAS Arrays

```
DATA NEW;
   SET OLD;
   ARRAY XX[2,3] X1-X6;
   DO METHOD = 1 TO 2;
      DO TIME = 1 TO 3;
         X = XX[METHOD,TIME];
         OUTPUT;
      END;
   END;
   KEEP SUBJECT METHOD TIME SCORE;
RUN;
```

SAS Arrays

- This program has been written analogously to the previous one, except that it is in two dimensions.
- You use XX for for the name of the array and X for the name of the new variable in the data set NEW.
- The nested loops read through the data sequentially from X1 through X6 and assign the proper value to variables METHOD and TIME.
- The outer DO METHOD loop sets the variable METHOD to 1 and 2. The inner DO TIME loop cycles through the three times for each method.
- Each element of the array is is therefore selected, identified as to method and time and output as X to the new data set.

The RETAIN statement

The `RETAIN` statement works in the following manner. Before SAS reads a new record of data in the `DATA` step, it initializes each variable to a `MISSING` value. A `RETAIN` statement can be used to tell the system not to assign a missing value but rather to remember its value from past observations.

Suppose that you have some data which consists of one record per subject and two variables X1 and X2 for each subject. You want to print out the records of each subject while these are not identified in the data set.

The RETAIN statement

```
DATA PROBLEM;
   SUBJECT = SUBJECT + 1;
   INPUT X1 X2;
DATALINES;
 3  4
 5  6
 7  8
;
PROC PRINT DATA=PROBLEM;
   TITLE ’Incorrect Program’;
RUN;

DATA NOPROBLEM;
   RETAIN SUBJECT 0;
   SUBJECT = SUBJECT + 1;
   INPUT X1 X2;
DATALINES;
 3  4
 5  6
 7  8
;
PROC PRINT DATA=NOPROBLEM;
   TITLE ’Correct Program’;
RUN;
```

The RETAIN statement

- The first program is simply not right because for each iteration of the `DATA` step, all variables are initialized as missing (.). Since no value for `SUBJECT` was read, the result is missing value.
- The second program uses the `RETAIN` statement and initializes the value of `SUBJECT` to 0. As the data set is built, subject increases by 1 for each observation.

Printing your Data

For this part of the notes, we will create a data set, called MEDICAL, which will contain the following variables:

Variable Name	Description
SUB_ID	Subject ID
DIAGCODE	Diagnosis Code
ADMIT_DT	Admission Date
DISCH_DT	Discharge Date
HOSPCODE	Hospital Code
LOS	Length of stay
COST	Total cost of treatment

First, we will generate a simple report showing all the data.

Printing your Data

```
DATA MEDICAL;
INFORMAT ADMIT_DT DISCH_DT MMDDYY8. COST COMMA8.2;
INPUT SUB_ID DIAGCODE ADMIT_DT DISCH_DT HOSPCODE LOS COST;
FORMAT ADMIT_DT DISCH_DT MMDDYY8.;
DATALINES;
03916  291  04/13/92  04/14/92  19    1   325.00
09243  291  01/21/92  02/15/92  14   25  6000.00
71543  480  03/06/92  03/07/92  18    1   621.00
96298  480  01/06/92  01/18/92  17   12  7050.99
75986  493  01/13/92  01/27/92  18   14  5521.85
96913  493  03/02/92  03/02/92  15    0   200.00
;
```

Printing your Data

Here the `INFORMAT` statement gives the following information about the patterns in which some of the raw data elements are found:

- The data for `ADMIT_DT` and `DISCH_DT` are found in MM/DD/YY format.
- `commaW.D` informat: as an example of comma8.2 will allocate a total of 8 spaces for the output. 1 space is allocated for the decimal, 2 spaces for the number of decimals and 1 space for comma as a separator in every 3 digits.

Printing your Data

Generating a simple report:

```
PROC PRINT DATA=MEDICAL;
   VAR SUB_ID
       DIAGCODE
       ADMIT_DT
       DISCH_DT
       HOSPCODE
       LOS
       COST;
RUN;
```

Printing your Data

Dropping observation numbers and increasing readability:

```
PROC PRINT DATA=MEDICAL LABEL;
   TITLE  'Hospital Data Base Report';
   TITLE2 '-------------------------';
   ID   SUB_ID;
   VAR DIAGCODE
       ADMIT_DT
       DISCH_DT
       HOSPCODE
       LOS
       COST;
   LABEL DIAGCODE = 'Diagnosis Code'
         ADMIT_DT = 'Admission Date'
         DISCH_DT = 'Discharge Date'
         HOSPCODE = 'Hospital Code'
         LOS      = 'Length of Stay'
         COST     = 'Cost of Treatment';
   FORMAT COST DOLLAR7.
          SUB_ID SSN11.
          ADMIT_DT DISCH_DT MMDDYY8.;
RUN;
```

Printing your Data

- Use a LABEL statement in PROC PRINT and a separate LABEL statement. The second statement defines a set of variable labels which can be used instead of variable names as the column headings. The first statement tells SAS to use the the labels that are created in the first statement.
- Labels can be created in the DATA statement as well. But these will be kept global throughout the program.
- An ID variable replaces the OBS column and prints on the left side of the page.
- The format statement tells SAS to format the corresponding variables accordingly. The SSN11. format adds the leading zeroes as well as the dashes to the data when displaying them as Social Security Numbers.

Printing your Data

```
OPTIONS NOCENTER NODATE NONUMBER;

PROC SORT DATA=MEDICAL;
   BY DIAGCODE;
RUN;

PROC PRINT DATA=MEDICAL N LABEL;
   BY DIAGCODE;
   TITLE 'Hospital Data Base Report';
   TITLE2 '-------------------------';
   FOOTNOTE 'This is a footnote';
   SUM LOS COST;
   SUMBY DIAGCODE;
   ID SUB_ID;
   VAR DIAGCODE
       ADMIT_DT
       DISCH_DT
       HOSPCODE
       LOS
       COST;
   LABEL DIAGCODE = 'Diagnosis Code'
         ADMIT_DT = 'Admission Date'
         DISCH_DT = 'Discharge Date'
         HOSPCODE = 'Hospital Code'
         LOS      = 'Length of Stay'
         COST     = 'Cost of Treatment';
   FORMAT COST DOLLAR7.
          SUB_ID SSN11.
          ADMIT_DT DISCH_DT MMDDYY8.;
RUN;
```

Printing your Data

The previous examples illustrates how we can print out various summaries, footnotes and titles, and some overall system formatting options. The data are grouped by DIAGCODE.

- Use the PROC SORT option to sort the data by DIAGCODE, assuming that they have not been sorted before.
- The BY statement in the other procedure tells the system to use the BY groups. Then, the output is shown for each BY group.
- The N option gives the number of observations in the data set. When a BY statement is used, the N option shows the number of observations by group.
- The FOOTNOTE statement gives a short sentence at the bottom of each output page.
- The SUM statement prints the sums for the listed variables. The SUMBY statement is used only when there exists a BY statement and gives the sums of the variables.
- In the OPTIONS statement your titles are left aligned and you omit date, time and page numbering.