

Transforming Data in SAS[®]1: Restructuring Data Sets, Creating Look-Up Tables, and Forming Person-Year Records for Event History Analysis in SAS

Lawrence C. Marsh and Karin L. Wells

Department of Economics
University of Notre Dame
Notre Dame, Indiana 46556

introduction

Although SAS[®] is highly regarded for its multitude of statistical procedures, new users may not be fully aware of its extensive programming capabilities. For example, while a DATA section in SAS serves as a DO loop over observations, SAS's DO OVER loop allows at the same time for a "side-ways" do loop over variables. The focus of our paper is on using such SAS programming features to carry out data transformations needed for subsequent analysis by SAS procedures.

First we will discuss and provide some SAS code that may be useful for creating dummy (binary, indicator) variables from a single, multi-valued variable. Next we will demonstrate one way of creating a look-up table for adding one or more variables to an observation. Finally we show a method of creating person-year records from person records in preparation for an event history analysis.

creating dummy variables from a single variable

Often nominal data or even ordinal data are coded as character data such as F for female and M for male, or B for Buddhist, C for Christian, J for Jewish, and M for Muslim, or L for lower, M for middle, and U for upper. In order to perform statistical analysis it is frequently necessary to recode these character values into numeric values. Moreover, it may not be appropriate for some statistical analyses to simply recode nominal variables such as religion just as a single variable with integer values. For regression analysis religion might be better represented as a set of dummy variables, one for each religion. For example a dummy variable for Buddhist would be created that would take on the value 1 (one) if the person was of the Buddhist faith and a 0 (zero) if the person was not Buddhist. Thus, each religion may be given its own dummy variable.

Moreover, even an ordinal variable may be better represented with a set of dummy variables. For example, if the price of a residential property is being explained in part by the number of bedrooms, using a single variable BEDROOMS coded 1, 2, 3, et cetera, corresponding to one-bedroom, two-bedroom, three-bedroom homes, et cetera, forces the change in price as each additional bedroom is added to be held constant by the analysis. A more flexible alternative approach would be to create separate dummy variables for each bedroom type. This would allow the increase in price as an additional bedroom is added to be different depending on how many bedrooms are already present in the home. This would allow for the possibility of diminishing returns in the addition of bedrooms to a home but would not impose this as a restriction.

¹SAS is a registered trademark of SAS Institute, Inc., Cary, NC.

These examples motivate the need for SAS programming to translate a single nominal or ordinal variable (whether coded as character or numeric) into a corresponding set of dummy variables. If the variable RELIGION takes on four possible values then the program must create four dummy variables. If the dummy variable STATE takes on fifty possible values then fifty dummy variables must be created. Of course the use of multiple sets of dummy variables in a single regression analysis may bring about a perfect multicollinearity problem under some regression model setups. However, due to space limitations here, we must leave a discussion of such problems to some future paper.

The following SAS code takes the variable STATE which has for values the standard two-letter state codes, and creates a set of fifty dummy variables, one for each state.

```
PROC SORT; BY STATE;
DATA; SET; BY STATE;
IF FIRST.STATE THEN J+1;
ARRAY D D1-D50;
DO OVER D; I+1; D=0;
IF I=J THEN D=1; END; I=0;
PROC PRINT; VAR STATE D1-D50;
```

This SAS code makes use of the ability of SAS to identify the first occurrence of the value of a sorted variable using the FIRST.vble statement. It also uses SAS summation statements as in J+1; and I+1; where variables taking on the set of positive integers are being created as the value from the previous observation is augmented by 1 to create the value for the next observation. SAS ARRAY statements are used to create a list of variables for the DO OVER statement to operate on one variable at a time.

The SAS code above is useful when the number of values the original variable takes on are known in advance such as fifty for the STATE variable. However, often we may not know or want to bother to determine in advance the number of different values a variable such as OCCUPATION, DISTRICT, or INDUSTRY may take on. We need SAS code that will automatically determine the number of different possible outcomes for a variable and create a dummy variable for each one.

```
PROC SORT OUT=ONE; BY DISTRICT;
DATA REDUCED; SET ONE; BY DISTRICT;
IF LAST.DISTRICT THEN OUTPUT;
DISTSET=DISTRICT; KEEP DISTSET;
PROC TRANSPOSE DATA=REDUCED PREFIX=D OUT=TWO;
DATA THREE; SET TWO; DEND=1;
DATA MATCH; IF _N_ THEN SET THREE; SET ONE;
PROC SORT OUT=FOUR; BY DISTRICT;
DATA FIVE; SET FOUR; BY DISTRICT;
IF FIRST.DISTRICT THEN J+1;
ARRAY D D1--DEND;
DO OVER D; I+1; D=0;
IF I=J THEN D=1; END; I=0;
PROC PRINT; VAR DISTRICT D1--DEND;
```

This SAS code first replaces the original DISTRICT variable with one that retains only a unique set of the possible outcome values of the original DISTRICT variable. Then it makes use of PROC TRANSPOSE which takes the single variable DISTSET containing only a unique set of the possible values and transposes that variable to form a single observation with a dummy variable for each possible outcome value. Since we don't know how many such dummy variables have been created, we simply create one additional one called DEND so that we may refer to the full set as D1--DEND without knowing how many there are. This works because SAS positions variables in the order in which they are created and the double-dash (--) picks up variables by position including all the variables between the one listed before the double-dash up to and including the variable listed right after the double-dash. This is entirely different from the use of a single dash such as in D1-D50 which is incrementing the integer value following the prefix of the first variable by 1 until the integer value following the prefix in the variable following the single dash is reached.

creating and reading a look-up table in SAS

Next we want to consider the problem of creating a look-up table. Such a table is useful in assisting us in adding appropriate variables to each observation based on the values of the original set of variables in that observation. The following program creates a look-up table on unemployment rates for each of the 51 states (including Washington, DC) for each of seven years (1971-1977). Then the program reads from the primary family data set being analyzed to determine the family's state of residence in each of the seven years and creates an unemployment rate variable for that family for that year corresponding to the unemployment rate in their state of residence that year. Thus the program adds seven new unemployment variables to each observation corresponding to the appropriate rate for that family in that year.

DATA ZERO;

INPUT STATE71-STATE77; CARDS;

01 03 01 05 15 15 34

51 51 49 49 49 49 51

30 07 07 07 07 07 07

12 15 30 12 12 13 13

05 05 05 02 18 18 14

Each observation represents a person or family and the state codes of their residence from 1971 through 1977. Some may never change their state of residence while others may do so frequently.

. THOUSANDS OF RECORDS

. WITH STATE OF RESIDENCE EACH YEAR 71-77

;

DATA ONE;

INPUT A1-A408 @@; CARDS;

01 5.5 4.5 4.5 5.5 7.7 6.8 7.4

02 4.7 4.2 4.1 5.6 12.1 9.8 8.2

03 5.4 4.6 4.1 4.8 9.5 7.1 6.6

.

. 51 LINES: STATE CODE FOLLOWED

. BY UNEMPLOYMENT RATES

.

50 8.8 7.6 7.0 7.3 9.9 9.2 8.2

51 4.0 3.6 3.4 3.9 6.9 5.9 6.2

;

Each observation represents a state with the state code given first followed by seven numbers that indicate the unemployment rate for that state for the years 1971 through 1977.

```

DATA ONE; SET ONE;
ARRAY A{408} A1-A408;
ARRAY STCODE{51} STCODE1-STCODE51;
ARRAY UNRATE71{51} U71U1-U71U51;
ARRAY UNRATE72{51} U72U1-U72U51;
ARRAY UNRATE73{51} U73U1-U73U51;
ARRAY UNRATE74{51} U74U1-U74U51;
ARRAY UNRATE75{51} U75U1-U75U51;
ARRAY UNRATE76{51} U76U1-U76U51;
ARRAY UNRATE77{51} U77U1-U77U51;
J = -7;
DO I = 1 TO 51;
J = J + 8;
STCODE{I} = A{J};
UNRATE71{I} = A{J+1};
UNRATE72{I} = A{J+2};
UNRATE73{I} = A{J+3};
UNRATE74{I} = A{J+4};
UNRATE75{I} = A{J+5};
UNRATE76{I} = A{J+6};
UNRATE77{I} = A{J+7};
END;
DATA ALL; SET ZERO;
IF _N_ = 1 THEN SET ONE;
ARRAY STATE (J) STATE71-STATE77;
ARRAY UNEMPLOY (J) UNEMP71-UNEMP77;
ARRAY STCODE (I) STCODE71-STCODE77;
ARRAY UNRATE71 (I) U71U1-U71U51;
ARRAY UNRATE72 (I) U72U1-U72U51;
ARRAY UNRATE73 (I) U73U1-U73U51;
ARRAY UNRATE74 (I) U74U1-U74U51;
ARRAY UNRATE75 (I) U75U1-U75U51;
ARRAY UNRATE76 (I) U76U1-U76U51;
ARRAY UNRATE77 (I) U77U1-U77U51;
ARRAY UNRATE (J) UNRATE71-UNRATE77;
DO J = 1 TO 7;
DO I = 1 TO 51;
IF STATE = STCODE THEN UNEMPLOY = UNRATE;
END;
END;
PROC PRINT; VAR STATE71-STATE77 UNEMP71-UNEMP77;

```

The strategy here is simply to first attach the look-up table to each observations and then find the unemployment rates in the table that correspond to the state of residence for that person or family in that year.

A set of seven new variables representing unemployment levels for the state of residence for that family for the seven years from 1971 through 1977 are created.

If the state of residence equals the state code then an unemployment rate variable is created for that observation for that particular state in that particular year. Thus, by knowing the family's state of residence for each year in a seven year period we are able to create and attach seven new variables with the unemployment rates in those states for each of those years. In this

example only seven years of unemployment rates are created and attached to each observation but this code may easily be expanded to accommodate any number of years.

creating person-year records for event history analysis²

Each of the original observations provides information on the employment history of the head of household for seventeen years.

*CREATION OF PERSON-YEARS FOR HEADS;

```
DATA FIRST; SET ALL;
TIME = 0; HEMP89 = 0;
ARRAY GOV1{18} HEMP71-HEMP88;
ARRAY GOV2{18} HEMP72-HEMP89;
DO Z = 1 TO 18;
    IF GOV1{Z} = 1 THEN DO;
        TIME + 1;
        IF GOV2{Z} = 0 THEN Z = 18;
    END;
END;
```

The variable TIME provides a count of the number of consecutive years of government employment by the head of household where head of household is as defined by the Panel Study of Income Dynamics.

When the next year indicates an end of employment with the government the loop terminates.

```
DATA HEADS; SET FIRST;
```

```
DO A = 1 TO 18;
```

```
    ARRAY ONE{17} HEMP71-HEMP87;
    DO B = 1 TO 17;
        IF ONE{B} = 1 THEN DO;

            ARRAY TWO{17} AGE71-AGE87;
            DO C = 1 TO 17;
                IF C = B THEN DO;
                    AGE = TWO{C};
                END;
                IF C = B THEN C = 17;
            END;
        .
        .
        .
        .
```

These transformations are restricted to years when heads were working for the government.

The variables stored include age, race, gender, occupation, industry and event. Event is the way in which the head's employment with the government ended (if it ended at all).

²Adapted from the work of Jay Teachman, University of Washington, as presented at the Event History Analysis Workshop at the University of Michigan, July 1993. Professor Teachman is not responsible for any errors in this paper.

```

ARRAY TWENTY{17} EVENT71-EVENT87;
DO Q = 1 TO 17;
    IF Q = B THEN DO;
        EVENT = TWENTY{Q};
    END;
    IF Q = B THEN Q = 17;
END;

ONE{B} = 0; B = 17;
IF EVENT = 0 THEN CENSOR = 1;
ELSE IF EVENT = 10 THEN CENSOR = 2;
ELSE IF EVENT = 11 THEN CENSOR = 3;
ELSE IF EVENT = 12 THEN CENSOR = 4;
ELSE CENSOR = 0;
IF TIME LE A AND CENSOR = 0 THEN OCCUR = 0;
ELSE OCCUR = 1;
IF OCCUR = 0 THEN A = 18;
OUTPUT;
END;
END;
END;

```

The variable CENSOR is created on the basis of the type of event such that CENSOR has values 0, 1, 2, 3 or 4.

Right censored.

Non-response censored.

Retirement at age 62.

Retirement at age 65.

Not censored (i.e. the event of interest occurred).

Dummy variable OCCUR is created.

The OUPUT command creates a person-year observation for every year an individual is at risk of the event up to and including the year the event occurred. Thus the newly transformed data set contains a set of unique records representing each year each person worked for the government, including the year in which the person left government employment or the year in which the person was censored. In this example person-year observations are created only for heads of households as defined by the Michigan Survey Research Center for their Panel Study of Income Dynamics (PSID) data set.

summary and conclusion

In this paper we have attempted to demonstrate once again the power of SAS in carrying out moderately complicated data transformations. In particular we have shown some SAS code for automatically creating a set of dummy variables from a single multi-valued variable. This was shown both for the case where the number of possible unique outcome values was known in advance and for when one wishes to have the program automatically determine the number of possible outcomes and, therefore, the number of dummy variables that were needed. Note that other SAS programmers may have alternative strategies for carrying out this task. We do not claim to have the most efficient possible algorithm but merely one of possibly many that will do the job.

A second application of SAS programming methods involved the creation of a look-up table of unemployment rates for each state for each year from 1971 through 1977. The number of years was deliberately restricted to make the demonstration manageable but could easily be expanded to accommodate additional years. The ability to attach a look-up table to each observation was needed because more than one variable had to be created for each observation. Again, this SAS algorithm may have many variations and competitors. We have presented but one way to accomplish this.

Finally, we discussed an approach to creating event-person records from panel data. The increasing popularity of event history analysis necessitates this transformation and creation of appropriate observations for this sort of analysis. We would be interested to learn of other approaches to preparing a data set for event history analysis with multiple spells of an event.

references

Hill, Martha S., *The Panel Study of Income Dynamics: A User's Guide*, Sage Publications: Newbury Park, California, 1992.

SAS Institute Inc., *SAS Procedures Guide*, SAS Institute Inc.: Cary, North Carolina, 1990.