Macros & Matrices: Using the IML Procedure to Access Tabular Information

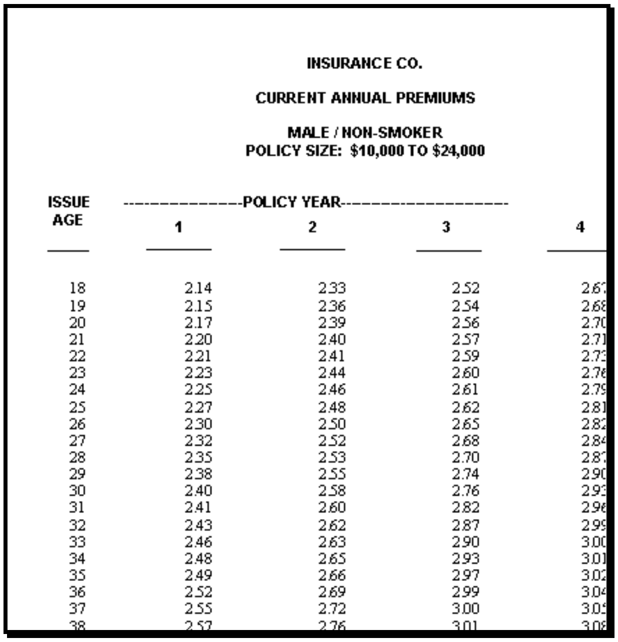
Jamie Perrett, Educators Mutual, Murray, UT

INTRODUCTION

Sometimes it is necessary to use data in worksheet form and access this data one cell at a time. For example, when calculating insurance rates, you may use pre-determined rates on a rate sheet rather than formulas. It is then necessary to have some type of process that will use given information (i.e. Age, Sex, Smoker/Non-Smoker, etc.) and look up a table value needed to calculate the rate. One way to organize this information is the combined use of macros and matrices through base SAS and SAS/IML®.

RATING EXAMPLE

With a table or rate sheet in the following format (Figure1), five variables of interest are needed to determine the appropriate table rate: sex, smoker/non-smoker, policy size, issue age and policy year. There could easily be more variables depending on the complexity of the rating system.



INSURANCE CO.

CURRENT ANNUAL PREMIUMS

MALE / NON-SMOKER
POLICY SIZE: $10,000 TO $24,000

ISSUE AGE	---------------POLICY YEAR---------------------			
	1	2	3	4
18	2.14	2.33	2.52	2.67
19	2.15	2.36	2.54	2.68
20	2.17	2.39	2.56	2.70
21	2.20	2.40	2.57	2.71
22	2.21	2.41	2.59	2.73
23	2.23	2.44	2.60	2.76
24	2.25	2.46	2.61	2.79
25	2.27	2.48	2.62	2.81
26	2.30	2.50	2.65	2.82
27	2.32	2.52	2.68	2.84
28	2.35	2.53	2.70	2.87
29	2.38	2.55	2.74	2.90
30	2.40	2.58	2.76	2.93
31	2.41	2.60	2.82	2.96
32	2.43	2.62	2.87	2.99
33	2.46	2.63	2.90	3.00
34	2.48	2.65	2.93	3.01
35	2.49	2.66	2.97	3.02
36	2.52	2.69	2.99	3.04
37	2.55	2.72	3.00	3.05
38	2.57	2.76	3.01	3.08

Figure 1

POSSIBLE CONCERNS

When accessing tabular data, several concerns must be addressed. Here are four.

1.Speed
Depending on the computer's capabilities and your company's time tables, a program's speed in looking up values can be very important.

2. Disk Space
Also depending on the computer's capabilities and company needs, the disk space allotted to a task may or may not be limited.

3. Code Efficiency
Sloppy, long code can mean headaches to the programmer who has to decipher it for future adjustments. It can also mean unnecessary steps for SAS during run time.

4. Software Constraints
SAS/IML is not a part of base SAS and is sometimes not available to the programmer.

POSSIBLE SOLUTIONS

Several solutions exist for these problems. Here are two feasible solutions that warrant consideration for their versatility. You may know another way of handling this problem that better suits your specific situation.

A. SQL Procedure

One way to access the table rates in Figure 1 is to put the rates into one or more data sets with data set name(s) and additional variables to enable a program to search through rows and columns (via "IF-THEN" or "WHERE" statements) to find the desired values.

An SQL procedure could be written as follows:

```
PROC SQL;
CREATE TABLE w_rates AS
SELECT DISTINCT a.*, b.rates
FROM in_data as a, r_table as b
WHERE a.sex=b.sex and a.smoker=b.smoker and
a.pol_size=b.pol_size and a.issu_age=b.issu_age and
a.pol_year=b.pol_year;
```

There are specific advantages and disadvantages with this method.

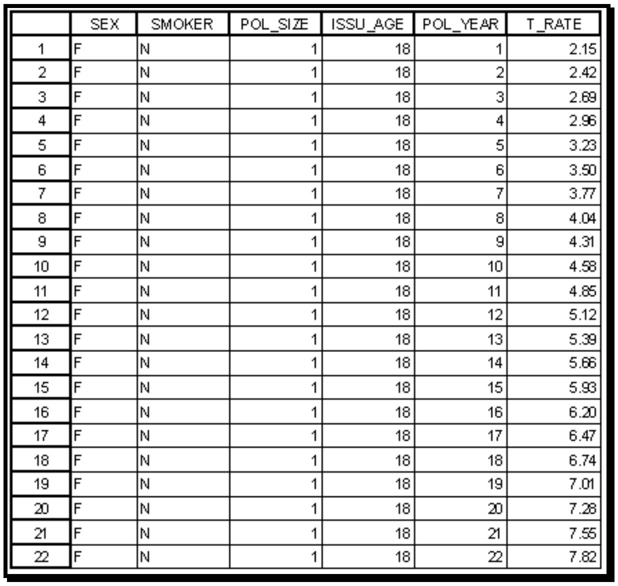
Advantages

1. Since the SQL procedure is a part of Base SAS, no additional products are needed to use this method.

2. The code for the actual program (the SQL procedure) is short and easy to understand.

Disadvantages

1. The data set(s) used for this method would have to be similar to the following (Figure 2):



	SEX	SMOKER	POL_SIZE	ISSU_AGE	POL_YEAR	T_RATE
1	F	N	1	18	1	2.15
2	F	N	1	18	2	2.42
3	F	N	1	18	3	2.69
4	F	N	1	18	4	2.96
5	F	N	1	18	5	3.23
6	F	N	1	18	6	3.50
7	F	N	1	18	7	3.77
8	F	N	1	18	8	4.04
9	F	N	1	18	9	4.31
10	F	N	1	18	10	4.58
11	F	N	1	18	11	4.85
12	F	N	1	18	12	5.12
13	F	N	1	18	13	5.39
14	F	N	1	18	14	5.66
15	F	N	1	18	15	5.93
16	F	N	1	18	16	6.20
17	F	N	1	18	17	6.47
18	F	N	1	18	18	6.74
19	F	N	1	18	19	7.01
20	F	N	1	18	20	7.28
21	F	N	1	18	21	7.55
22	F	N	1	18	22	7.82

Figure 2

As you can see, five variables are added to the data set. In the portion of the data set visible in figure 2, there are 110 values and 22 rates. These 132 values of the data set only provide rates for a non-smoking female of one specific policy size who was issued the policy at age 18 and has held the policy for less than 23 years. When both genders, smoking and non-smoking, all policy sizes, issue ages and policy years are added, the size of the data set can be enormous. Time spent entering this data is one specific disadvantage.

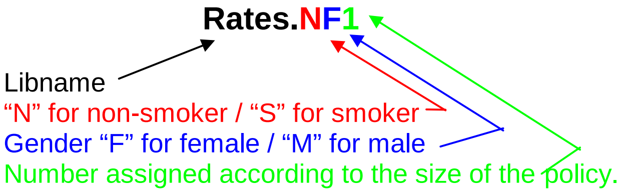
2. The additional variables in the rates data set(s) can take up valuable disk space.

3. SAS will have to check every record to find the one that matches the criteria of the SQL procedure. Depending on the size of the rates data set(s), this search may take a lot of CPU time.

B. Macro/Matrix Method

With the second method, each rate sheet is made into a data set. The name of the data sets would be an acronym describing the specific rates in each data set.

Example of data set name:



Rates.NF1

Libname

"N" for non-smoker / "S" for smoker

Gender "F" for female / "M" for male

Number assigned according to the size of the policy.

The following code will access the desired rate from the desired table:

The following is a macro that uses policy information to find the corresponding table rate.

```
%Macro Get_Prem(Dset);

Data _Null_;
      Set &Dset;
      Data=”Rates.”||Smoker||Sex||Pol_Size;
      call symput(‘data’,data);
      call symput(‘issu_age’,issu_age);
      call symput(‘Pol_year’,Pol_year);
Run;

Proc IML;
      Use &data;
/* Create matrix X using data set &data. */
      Read All Var {one, two, three, four, five}
      Into x;
      Rate = x[&issu_age - 17,&Pol_year];
      Edit &Dset;
/* Calculate the premium. */
      Premium = (&Face_Val*Rate/1000)
/* Replace the old value of Premium with its new
value. */
      Replace;
Quit;

%Mend;
/* end of macro Get_Prem */
```

The following is a macro that can be used in conjunction with macro %Get_Prem finding table rates for more than one record.

```
%Mult_Prem(In_Dat,Out_Dat);

/* Initialize the output data set. */
Data &Out_Dat;
     Length Policy $12 Premium 8;
     Policy=” “;
Run;

/* Find the number of records in the input data set. */
Data _Null_;
     Set &In_Dat END=LAST;
     If last Then Call Symput(‘Last’,_N_);
Run;

%Let Current=1;

/* Repeat the following steps for every record in the
input data set. */
%Do While (&Last GE &Current);
     Data One_Pol;
           Set &In_Dat (Obs=&Current) End=Last;
            If Last Then Output;
     Run;

     %Get_Prem(One_Pol);

     Data &Out_Dat;
            Set &Out_Dat One_Pol;
            If Policy=” “ Then Delete;
     Run;

     %Let Current=%Eval(&Current+1);
%End;

%Mend;
/* end of macro %Mult_Prem */
```

There are advantages and disadvantages with the Macro/Matrix method.

Advantages

1. The actual rates would be the only values contained in the rates data sets. (The extra variables would not be needed.) This would reduce the time needed to create and update the data sets and the amount of required storage space.

2. Depending on the number of policies and the size and number of rates sheets, the time required to locate the rates would be significantly less than the time needed for the same task using the SQL method. This is because the program knows the exact row and column of the data set where the desired rate is located. With the Macro/Matrix method you don’t need to check each record of the data set to find the desired rate.

CONCLUSION

Both methods are valid for accessing tabular values.

Under the following conditions one should use the [*]Macro/Matrix method:

- Very large table(s) (i.e. 1000+ values).
- Many variables required to locate the table value.
- Limited disk space.
- Speed is important.

Conditions under which an alternative method should be used:

- No access to SAS/IML.
- Small tables (i.e. less than 1000 values).
- Few variables required to locate table values.
- Unlimited disk space.
- Speed is not of great importance.

Author:
JAMIE PERRETT
EDUCATORS MUTUAL INSURANCE ASSOC.
852 E. ARROWHEAD LN
MURRAY, UT 84107
(801) 262-7476 ext. 410
jp72@email.byu.edu

[*] The Macro/Matrix method generates a very large log file. To eliminate problems associated with this, use the NONOTES option and turn off the MPRINT and MLOGIC options when calling these macros.