

Simplifying the Use of Multidimensional Array in SAS®

Alec Zhixiao Lin, Loan Depot, Foothill Ranch, CA

ABSTRACT

Multidimensional array is a common and effective way to code a look-up table in SAS. However, many users find this practice to be confusing and daunting. This paper introduces a process that greatly simplifies the use of multidimensional array. Users only need to define the dimensions of a look-up table and paste it to SAS with no need to understand the logic for programming a multidimensional array. The macro will automatically assign given values to each record.

INTRODUCTION

Business strategies use look-up tables very often. In the following example of credit card, analysts use FICO and a risk score (usually a company's propriety score) to assign APR (Annual Percentage Rate) to new customers.

FICO Band	Proprietary Risk Score									
	1	2	3	4	5	6	7	8	9	10
660-680	24.95%	23.95%	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%
681-700	23.95%	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%
701-720	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%
721-740	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%
741-760	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%
761-780	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%
781-800	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%	9.95%
>800	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%	9.95%	8.95%

Table 1 – Assigning Interest Rate for Credit card

SAS provides various ways to inputting the above look-up table to the system.

CODING FOR MULTIDIMENSIONAL ARRAY

We will illustrate two commonly used methods before introducing a new one.

Method 1 – IF-THEN statement

Most users use a series of IF-THEN statements as follows:

```
if fico_band='660-680' then do;
  if riskscore_tier=1 then rate=0.2495;
  .....
  else if riskscore_tier=9 then rate=0.1695;
  else if riskscore_tier=10 then rate=0.1595;
end;
else if fico_band='681-700' then do;
  if riskscore_tier=1 then rate=0.2395;
  .....
  else if riskscore_tier=9 then rate=0.1595;
  else if riskscore_tier=10 then rate=0.1495;
end;
else if fico_band='701-720' then do;
  .....
  .....
```

The above code is most straightforward to understand, but it will encounter the following challenges in actual use:

- It is prone to errors in copying rates from the look-up table to SAS.

- Analysts usually need to experiment with multiple rounds till a satisfying look-up table is obtained. Every time the SAS code has to be rewritten in order to start a new round of simulation.
- SAS code has to be rewritten if one or both dimensions of the look-up table change to different attributes, e.g., if the proprietary risk score is changed to a revenue score.
- A change in the structure of the look-up table, e.g. from 8x10 to 6x5, requires a substantial rewriting of the SAS code.

Method 2 – Common Way of Use Multidimensional Array

Multidimensional array is an effective solution to conquer the problems above, as illustrated by the following:

```
data &outfile;
set &infile;

ARRAY DOUBLE{8,10}
  R1C1-R1C10
  R2C1-R2C10
  R3C1-R3C10
  R4C1-R4C10
  R5C1-R5C10
  R6C1-R6C10
  R7C1-R7C10
  R8C1-R8C10
(
0.2495 0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595
0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495
0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395
0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295
0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195
0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095
0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995
0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995 0.0895
)
;

if fico_band='660-680' then do;
  if riskscore_tier=1 then interest_rate=R1C1;
  else if riskscore_tier=2 then interest_rate=R1C2;
  else if riskscore_tier=3 then interest_rate=R1C3;
.....
  else if riskscore_tier=9 then interest_rate=R1C9;
  else if riskscore_tier=10 then interest_rate=R1C10;
end;

.....

if fico_band='>800' then do;
  if riskscore_tier=1 then interest_rate=R8C1;
  else if riskscore_tier=2 then interest_rate=R8C2;
  else if riskscore_tier=3 then interest_rate=R8C3;
.....
  else if riskscore_tier=9 then interest_rate=R8C9;
  else if riskscore_tier=10 then interest_rate=R8C10;
end;
;
run;
```

Not only is the above code long, but many users find it to be quite confusing. Also, if one wants to change the array to a different structure (such as from 8X10 to 6X5), one needs to substantially rewrite the above codes.

Method 3 – A More Flexible Use of Multidimensional Array

The only prerequisite is that the all variables used in multiple dimensions should be expressed in ordinal numbers from 1 on. In our example, the FICO band should be changed to an expression of 1 to 8.

FICO Tier	FICO Band	Proprietary Risk Score									
		1	2	3	4	5	6	7	8	9	10
1	660-680	24.95%	23.95%	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%
2	681-700	23.95%	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%
3	701-720	22.95%	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%
4	721-740	21.95%	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%
5	741-760	20.95%	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%
6	761-780	19.95%	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%
7	781-800	18.95%	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%	9.95%
8	>800	17.95%	16.95%	15.95%	14.95%	13.95%	12.95%	11.95%	10.95%	9.95%	8.95%

Table 2 – add a field of FICO Tier denoting ranks from 1 to 8

The macro introduced in the paper might look very confusing, but it only requires users to define the macro values at the beginning of a data step:

```
%let infile=data_for_paper;          ** input file;
%let outfile= badrate_newcg;         ** output file;
%let vertivar=fico_tier;             ** vertical attribute of the array;
%let horivar=riskscore_tier;         ** horizontal attribute of the array;
%let rowcnt=8;                      ** # vertical tiers;
%let colcnt=10;                     ** # horizontal tiers;
%let valueasg=interest_rate;        ** attribute for assigning values to;
%let arraymcr=
0.2495 0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595
0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495
0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395
0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295
0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195
0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095
0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995
0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995 0.0895
;
```

(See Appendix for the complete code.)

Do not change any the names of the macro variables such as *infile*, *outfile*, etc. These macro variables will be used by the subsequent program that will automatically assign rates to the corresponding records in the file.

If one would like to change the structure of the look-up table from 8x10 to 6x5 or use a different attribute, he/she only needs to change the related macro values *rowcnt* and *colcnt* without rewriting a lot of codes. Users do not need to digest or understand the logic for programming multidimensional array in SAS. For quality check, the process uses PROC TABULATE at the end and generates a summary that looks completely consistently with the look-up table.

CONCLUSION

This paper introduces a SAS process that simplifies the use of multidimensional array in SAS. We hope it provides some good help for coding look-up tables in SAS.

ACKNOWLEDGEMENTS

I would like to thank the modeling team in Loan Depot led by Shawn Benner for their continuous support for this endeavor.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at

Alec Zhixiao Lin
 VP of Modeling
 Loan Depot
 26642 Towne Center Drive
 Foothill Ranch, CA 92610
 Email: alecindc@gmail.com
 Web: www.linkedin.com/pub/alec-zhixiao-lin/25/708/261/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

APPENDIX

The macro values defined above will be used by a few macro programs that will automatically assign rates to the corresponding records in the file.

```

** only change the macro values;
** do not change the names of macro variables;
%let infile=data_for_paper;          ** input file;
%let outfile= badrate_newcg;         ** output file;
%let vertivar=fico_tier;              ** vertical attribute of the array;
%let horivar=riskscore_tier;          ** horizontal attribute of the array;
%let rowcnt=8;                       ** # vertical tiers;
%let colcnt=10;                      ** # horizontal tiers;
%let valueasg=interest_rate;         ** attribute for assigning values to;
%let arraymcr=
0.2495 0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595
0.2395 0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495
0.2295 0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395
0.2195 0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295
0.2095 0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195
0.1995 0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095
0.1895 0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995
0.1795 0.1695 0.1595 0.1495 0.1395 0.1295 0.1195 0.1095 0.0995 0.0895
;

** do not change anything below;
** macro for multidimensional array starts;
%macro sharray;
array arraymcr{&rowcnt, &colcnt}
%macro colcnt;
%do j=1 %to &rowcnt;
    verticalbin&j._horizontalbin1-verticalbin&j._horizontalbin&colcnt
%end;
%mend;
%colcnt
(
&arraymcr
);
%mend;

%macro asgcell;
variablefill=1; if variablefill=. then delete;
%macro stkarray;
%do j=1 %to &rowcnt;
    %do i=1 %to &colcnt;
        else if verticalbin=&j and horizontalbin=&i then
&valueasg=verticalbin&j._horizontalbin&i;
    %end;

```

```

%end;
%mend;
%stkarray;
%mend;

%macro dropvar;
%do j=1 %to &rowcnt;
  %do i=1 %to &colcnt;
    verticalbin&j._horizontalbin&i
  %end;
%end;
%mend;

%macro forarray;
verticalbin=&vertivar;
horizontalbin=&horivar;

%sharray;
%asgcell;

drop variablefill
  %dropvar
;
%mend;
** macro for multidimensional array ends;
** do-not-change ends;

data &outfile;
set &infile;

** you can add SAS code here as you like;

%forarray;  ** invoke the macro for multidimensional array here;
run;

** the following table should be completely consistent with the table;
proc tabulate data=&outfile format=10.0;
class &vertivar &horivar;
table (&vertivar all), (&horivar all)*n; run;

proc tabulate data=&outfile format=6.5;
class &vertivar &horivar;
table (&vertivar all), (&horivar all)*pctn; run;

proc tabulate data=&outfile format=6.5;
class &vertivar &horivar;
var &valueasg;
table &vertivar, &horivar*&valueasg*mean; run;

```