ABSTRACT

Many industries are keenly interested in upselling their customers additional products. In the credit card industry, the active cardholder is generally deemed initially eligible for a wide variety of additional offers. However, it is often not feasible to present to an account all the sales offers they might qualify for. This is quite relevant if an account holder calls into the customer service call center, where realistically one or maybe two offers can be presented. Another application for this is direct mail, which usually focuses on one or two offers to highlight in a given mailing.

Which offer is the best one for the customer service rep to present? SAS® ARRAY processing provides an efficient way to determine which specific offer is the best for the rep to sell. Assuming prepped data, the array can prioritize a group of offers to determine the all-important “best offer”. The net result is that the customer has the Right Offer at the Right Time. Additional steps, such as arithmetic functions and PROC Transpose, can take the variables created by the array and convert them into a format suitable for loading into a customer service database system or direct mail datafile.

INTRODUCTION

Like many industries, credit card companies are seeking to maximize the revenue from each current customer. One way to do this is to offer them additional products and services. These would include items such as balance transfers, various insurances tied to the card, additional or upgraded cards, flexible payment terms, various rewards of cash-back offers, magazine subscriptions, or any other product similar to the ads included in a credit card statement. In any given month or statement cycle, the marketing teams can easily have 15-20 different offers to choose from to implement in various channels.

WHAT IS AN ARRAY?

While 15-20 offers may not seem like a huge number of offer variables, writing repetitious code to handle the 15-20 variables can become quite tedious very quickly. Each of these 15-20 offers has an associated profitability value. So the 15-20 programs have rapidly mushroomed into 30-40 variables to deal with. Clearly, hand coding this many variables is very tedious, as well as ripe with the possibility of mistakes. Another method is in order.

The SAS system provides a statement, the ARRAY statement, to handle this type of situation. The ARRAY statement is one of the statements used within a specific DATA step. At a high level, an array is a means to dynamically refer to a group of variables. The ARRAY statement defines variables to be processed as a group. The variables referenced by the array are called elements. The array then implements a process (or
processes) on these variables as a group. It is also possible to include a DO loop within the ARRAY statement to indicate how many times the ARRAY statement should process the elements within the defined group of variables. This is what enables the programmer to avoid the repetitive coding that can be involved in working with many variables. This processing can also lead to the creation of new variables.

**Data Preparation**

Before we do our array, we have to prepare the data so that the array can do its processing of the variables. In this case, we will be also be using the array to create new variables. The preparation actually begins before the analyst uses SAS at all. The marketing group has to provide criteria as to who qualifies for an offer, and rank which programs are the most valuable. In other words, which offer makes the company more money? Once the analyst has these relative offer values, s/he can then qualify the accounts for the offers and then assign the appropriate offer values to offer.

A useful convention is to use a flag value of ‘1’ to signify that an account qualifies for a particular offer. For figure 1 below, consider “Offer1” to be a Balance Transfer, “Offer2” to be a Card Upgrade, “Offer3” to be an additional card on the account, with “Offer4” being a reading light. The “Ovalue” is a measure of relative profitability to the company, with an Ovalue of 10 being more profitable than an Ovalue of 6.

**Figure 1.**
Portion of dataset showing the accounts that qualify for specific offers, and their respective Ovalues.

<table>
<thead>
<tr>
<th>Account</th>
<th>Offer1</th>
<th>OValue1</th>
<th>Offer2</th>
<th>OValue2</th>
<th>Offer3</th>
<th>OValue3</th>
<th>Offer4</th>
<th>OValue4</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67852</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>28374</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10394</td>
<td></td>
<td></td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18473</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98765</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>88364</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

In a real marketing campaign context, there can easily be hundreds of thousands or millions of accounts, and 15-20 sales offers a given cardholder or customer could potentially qualify for.

**Determining the Best Offer**

We will use the ARRAY statement below to create a series of new variables. These variables will in turn be manipulated by arithmetic functions and Proc Transpose to create a format suitable for upload into a customer service/DM database environment.
Figure 2
Offers Ranked By Accounts in a Format Suitable for Customer Service Upload or Direct Mail Campaign.

<table>
<thead>
<tr>
<th>Account</th>
<th>Offer</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>12345</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>67852</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>67852</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>67852</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>28374</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10394</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10394</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

This table has the top 4 accounts from Figure 1, but the format is different. The accounts now have their specific offers ranked in order of profitability. The array is the first step in the transformation of the data from Figure 1 into the required format of Figure 2.

ARRAY STATEMENT:

Rsubmit;
    Data sellme2;
    Set sellme1;
    Array anchor (1:4) Ovalue 1-Ovalue4;
    Array Z order1-order4;
    Array t[4] testvar1-testvar4;
    Do j=1 to 4;
        If anchor(j)= .; then anchor(j) = 0;
    End;
    Do k =1 to 4;
        T[k] max (of ovalue1-ovalue4);
        Do i = 1 to 4;
            If base (i) = t[k] then do;
                z(i) = k;
                Leave;
            End;
        End;
    Base(i) = -1;
End;
Run;
Endrsubmit;
Explanation of this Array

The data line names the output dataset; the set line is the dataset from which the array statement is reading its input data.

The first array is called “anchor”. It has 4 elements, which are the variables Ovalue1, Ovalue2, Ovalue3, and Ovalue4. (Arrays also work with many more than 4 variables). Array Z consists of a set of new variables to be created, which will be the order variables, corresponding to each (in this case, 4) of the Ovalues in the “anchor” array. Array T is a group of test variables, which will be used in the offer ranking process. The arrays to be used are now defined, which enables the rest of the processing to take place.

The first Do loop calls for 4 iterations of the further processing, one for each of the Ovalues by using the test variable from the array defined as T. It also makes sure that an Ovalue exists in this variable. From figure 1, not all accounts qualified for all of the offers. We want to exclude from processing the offers that an account did NOT qualify for by setting their Ovalues to 0.

The Do K =1 to 4 loop, with the max function, identifies which of the Ovalues is the highest, meaning that this is the most profitable offer remaining and should be ranked as such. Since this loop is from 1 to 4, it runs 4 times, looking for the maximum value.

The Do I loop takes the appropriate maximum value and links the order variables defined in the Z array to the specific Ovalue which has the top, second, third, and fourth rankings.

The base(i) statement simply ensures that once utilized, a given max Ovalue, and the program associated with it, do not get processed again by the array.

RESULT (Or, What Happens Now)

After using the array statements, we now have a set of variables (order1-order4) whose values range from 1 to x, with x in this case being 4. X is the number of variables the array processed, and can become quite large. In further data steps, we will generate the specific ranking variable for a specific program by multiplying it by the offer flag value (in effect, combining it) to get a program specific ranking variable suitable for processing by PROC Transpose. PROC Transpose then produces the format seen in Figure 2 above.

CONCLUSION

The ARRAY statement provides a method to avoid repetitive processing of multiple variables. A valuable application of the ARRAY statement is in a marketing context, where the SAS programmer needs to figure out which offer is the best offer for a given account. The sales offers are not necessarily numbered in order of profitability. An account may not have even qualify for all the potential offers, and varying numbers of accounts qualify for varying offers. Arrays (with DO Loops) are a tool to help line up a given account’s offers in order of their profitability.
REFERENCES

2. “Arrays Made Easy: An Introduction to Arrays and Array Processing”, presented at SUGI 30 by Steve First and Teresa Schudrowitz, both of Systems Seminar Consultants, Inc., Madison, WI.

CONTACT INFORMATION:

Len Zangwill
Zangwill Associates
lzang@dvol.com
215-517-5541

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.