Arrays: The How and the Why of it All

Jane Stroupe

When you need to join data sets, do you even ask yourself “Why don’t I just use an array?” Perhaps because arrays are a mystery or perhaps because you have never looked into how efficient arrays can be, you have not put them into your lookup toolbox. This presentation addresses both problems, How to utilize an array and why you should try it. In addition, you will discover how to use a multidimensional array to solve a tricky data combination problem.

Using One Dimensional Arrays

Here’s the problem. You have a SAS data set that looks like this:

ficticious_watch_orders

Date	ID	Price
01NOV2012	10003	1150
01NOV2012	12022	3150
01NOV2012	10004	1565
02NOV2012	11011	1655
02NOV2012	11012	2325
03NOV2012	12021	2590
03NOV2012	12022	3150
03NOV2012	12023	4300
03NOV2012	12024	3250
.	.	.
.	.	.
.	.	.

In this data set, the variable **ID** represents a character variable where the first two digits indicate the manufacturer of watches and the last two digits represents the style number. You need to combine the data with the following two SAS data sets. In these two SAS data sets, the variable **ID** is numeric.

ficticious_brands

ID	Name
10	Tagline
11	Roladex
12	Brett

ficticious_styles

ID	Name
1	Man About Town
2	Big Gold Face
3	Moons Abound
4	Aqua Proof
11	Prince Someone
12	Princess
13	Not Your Dads Watch
21	One Time Zone
22	Two Time Zones
23	Three Time Zones
24	Four Time Zones
25	Five Time Zones

There are several ways that you could proceed, but since the **ID** variable in the **ficticious_watch_orders** SAS data set is character and contains both the manufacturer id and the style id, you are going to have to process **ficticious_watch_orders** to split the **ID** variable. You could have combined the **ficticious_brands** SAS data and the **ficticious_styles** SAS data to create a variable which represents the combination of brand ID and style ID. Regardless of how you decide to create the appropriate ID, you still will have to sort the **ficticious_watch_orders** SAS data in order to merge the three data sets together.

So let's try to do this all in one DATA step and avoid a sort.

Example 1

First, a simple example will demonstrate how an array could be used to combine the **ficticious_brands** SAS data with the **ficticious_watch_orders** SAS data.

```
data watch_sales;
   set ficticious_watch_orders;
   array man{10:12} $20 _temporary_
                ('Tagline','Roladex','Brett'); ①
   M_ID=input(substr(ID,1,2),2.); ②
   Manufacturer=man{M_ID};
run;
```

① Create a temporary array that stores the character constants 'Tagline', 'Roladex', and 'Brett' in a length of 20. The array name is **man** and the index values range from 10 to 12.

② Use the SUBSTR function to extract the first two characters of the **ID** variable and the INPUT function to convert the value to numeric. ***Array indexes must be numeric.***

③ The variable **Manufacturer** is found by looking for the element of the array man that corresponds to the variable **M_ID**.

This example required that you type the values for the brand into the array.

Since you have the data in a SAS data set already, you can use IF/THEN logic and DO loops to load the array with the values of the brand.

Example 2

```
data watch_sales;
   set ficticious_watch_orders;
   array man{10:12} $20 _temporary_; ①
   if _N_=1 then do i=1 to NumObs; ②
      set ficticious_brands(rename=(ID=B_ID)) nobs=NumObs; ③
      man{B_ID}=Name; ④
   end;
   M_ID=input(substr(ID,1,2),2.);
   Manufacturer=man{M_ID};
run;
```

① Create a temporary array that stores the character constants in a length of 20. The array name is MAN and the index values range from 10 to 12. But no initial values are specified.

② The first time through the DATA step the DO loop is going to read all of the data from the **ficticious_brands** SAS data set. The **NumObs** variable is set during compile time when the NOBS= option on the SET statement names the variable. It is important to specify "**`if _n_=1`**" to execute the DO loop once. Executing the DO loop a second time would stop the DATA step because the SET statement would encounter the end of the SAS data set **ficticious_brands**, resulting in one observation in the **watch_sales** SAS data set.

③ In addition to creating the value of the variable **NumObs** during compile time, the RENAME= option changes the name of the **ID** variable in the **ficticious_brands** SAS data set so it is not confused with the ID variable in the **ficticious_watch_orders** SAS data set.

④ Load each element in the **man** array with the value of Name.

Now continue the program to include the style information from the **ficticious_styles** SAS data set.

Example 3

```
data watch_sales;
   set ficticious_watch_orders;
   array man{10:12} $20 _temporary_;
   array style{25} $35 _temporary_; ①
   if _N_=1 then do i=1 to NumObs;
      set ficticious_brands(rename=(ID=B_ID)) nobs=NumObs;
      man{B_ID}=Name;
   end;
   if _N_=1 then do i=1 to Num;
       set ficticious_styles(rename=(ID=S_ID)) nobs=Num; ②
       style{S_ID}=Name;
   end;
   M_ID=input(substr(ID,1,2),2.);
   St_ID=input(substr(ID,length(ID)-1),2.);
   Manufacturer=man{M_ID};
   Style_Name=style{St_ID};
run;
```

① Just like the previous ARRAY statement, this ARRAY statement creates an array named style to hold the constant names of the styles. Even though there are not 25 styles in the SAS data set **`ficticious_styles`**, the array refers to 25 values. The index variable for an array must be consecutive integers.

② Reads from the SAS data set **`ficticious_styles`** and names the NOBS= variable **NUM** (note, it is different from the previous NOBS= variable.)

After that, the program is almost identical to the previous one.

Example 4

The SAS data set sales contains the sales for five styles of watches:

Sales

Date	Style1	Style2	Style3	Style4	Style5
31JAN2012	129800	149800	171336	152024	1042
29FEB2012	282400	31400	372768	354912	4089
31MAR2012	68700	75570	90840	86156	9387
30APR2012	127200	13200	167900	151136	1333
31MAY2012	544000	584000	718800	62720	7754
30JUN2012	96000	105300	127160	117044	1728
31JUL2012	37000	38700	45840	41226	4943
31AUG2012	43980	43780	58536	52248	6267
30SEP2012	77600	85600	102430	92178	1166
31OCT2012	151400	166400	199980	179972	2928
30NOV2012	39000	42000	51480	46330	5554
31DEC2012	90000	99400	11980	10739	8742

The SAS data set **target** contains the target figures, in millions of dollars, for each month in the years 2009 to 2012.

Target

Year	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
2009	2	3	2	4	3	2	1	1	1	2	2	1
2010	3	5	3	6	5	4	3	3	3	3	4	3
2011	3	6	4	7	2	5	2	4	4	4	5	4
2012	5	5	3	6	4	6	4	2	5	6	6	5

To calculate the difference between the total sales amount for each date and the target for that month in 2012, you use PROC TRANSPOSE, and several DATA steps.

```
proc transpose data=target(where=(year=2012))
               out=target_T(rename=(col1=Target)) name=Mon;
run;
data target_T_mon;
   set target_T;
   where Target ne 2012;
   Month=input(substr(Mon,2),2.);
run;
data sales_sum(sortedby=Month);
   set sales;
   Month=month(Date);
   Total=sum(of Style:);
run;
data compare_;
   keep Date Style1-Style5 Total Target Difference;
   merge sales_sum target_T_mon;
   by Month;
   Difference=Total-(Target*1000000);
run;
```

.

Instead, you could have combined the two data sets in one DATA step, using an array.

```
data compare;
   keep Date Style1-Style5 Total Target Difference;
   array mon{*} Month1-Month12;
   if _N_=1 then set target
         (where=(Year=2012));
   set sales;
   Total=sum(of Style:);
   Month=month(Date);
   Target=mon{Month}*1000000;
   Difference=Total-Target;
run;
```

Using a Multidimensional Array to combine data

Suppose you have a data set that contains high and low temperatures, along with the wind speed:

temps

Date	High	Low	Wind
29AUG2012	8	0	14
30AUG2012	7	-2	22
31AUG2012	9	-1	20
01SEP2012	8	0	23
02SEP2012	9	1	19
03SEP2012	10	2	14
04SEP2012	12	3	16
05SEP2012	10	2	23
06SEP2012	12	0	12

In addition, you have a table that provides the temperatures based on the wind chill factors;

wchill

WSpeed	Neg10	Neg5	Tmp0	Tmp5	Tmp10	Tmp15	Tmp20	Tmp25	Tmp30
5	-22	-16	-11	-5	1	7	13	19	25
10	-28	-22	-16	-10	-4	3	9	15	21
15	-32	-26	-19	-13	-7	0	6	13	19
20	-35	-29	-22	-15	-9	-2	4	11	17
25	-37	-31	-24	-17	-11	-4	3	9	16
30	-39	-33	-26	-19	-12	-5	1	8	15
35	-41	-34	-27	-21	-14	-7	0	7	14
40	-43	-36	-29	-22	-15	-8	-1	6	13

Note that the wind speed is rounded to the nearest five, as is the temperature that ranges from -10 to 30.

To combine these two data sets, we will load the wchill data into a two dimensional array and be able to access the appropriate row and column for the high temperature and low temperature based on the wind speed based on the data in the temps SAS data set.

Here is the program:

```
data wndchll(keep=Date Wind High Low HighChill LowChill);
   array WC{8,9} _Temporary_; ①
   array farenheit{*} High Low HighChill LowChill; ②
   if _n_=1 then do I=1 to 8; ③
      set wchill;
      array Tmp{9} Neg10 -- Tmp30;
      do J=1 to 9; ④
         WC{I,J}= Tmp{J};
     end;
   end;
   set temps;
   Row=round(Wind,5)/5;⑤
   Column1=(round(High,5)/5)+3;
   Column2=(round(Low,5)/5)+3;
   HighChill=round(WC{Row,Column1});⑥
   LowChill=round(WC{Row,Column2});
   do i=1 to dim(Farenheit); ⑦
      Farenheit{i}=9/5*Farenheit{i}+32;
   end;
run;
```

① Creates a temporary array named **W** that refers to 8 rows and 9 columns. The rows and columns correspond to the observations and variables in the **wchill** SAS data set.

② Creates an array to refer to the **High** and **Low** temperatures in the **temps** SAS data set and create the new variables **HighChill** and **LowChill** temperatures.

③ The first time through the DATA step, the DO loop causes the SET statement to execute 8 times, thus reading through the **wchill** SAS data set. The **Tmp** array refers to the nine temperature variables in the **wchill** SAS data set.

④ The DO loop is going to fill the array **W** with all of the values in the data set **wchill**.

⑤ The three assignment statements round off the temperatures and wind speed that are being read from the **temp** SAS data set.

⑥ Using the **W** array, retrieve the value that is associated with the low temperature and wind speed, high temperature and wind speed.

⑦ The final DO loop is going to convert the variables containing the temperatures from Celcius to Fahrenheit.

Conclusion

There are many SAS techniques that can be used to combine data horizontally. If you have an appropriate index variable (remember, it must be a consecutive integer), an array could be the most efficient technique. And if you do not have a consecutive integer, try a DATA step hash object.

Contact Information

Jane Stroupe
SAS Contract Instructor
jgsstroupe@gmail.com