Advance Array Processing Techniques: Ingenious uses of Array for SAS

Gladys Vanessa Espinas, PPD
Eduard Joseph Siquioco, PPD

HELPING DELIVER LIFE-CHANGING THERAPIES

PPD

Main Author's Bio

+ Programmer Analyst in PPD
+ Previously worked in a local bank as a SAS programmer for Marketing Analytics
+ Attended a 3-month SAS Bootcamp Training Program facilitated by SAS Philippines in 2013
+ A hobbyist photographer residing in the Philippines

SECTION I: Basic Concepts

Defining Arrays

ARRAY statement

```
array array-name {n} <$><length> <array-elements> <(initial-value-list)>;
```

```
array charsamp {5} $ 10 varc1 varc2 varc3 varc4 varc5;
array numsamp (*) var1 var2 var3;
array initsamp [2] $ init1 init2 ('A', 'B');
```

Use an array reference statement to refer to an array defined within the same SAS DATA step:

```
charsamp {2}
```

Using Arrays

```
array numsamp (*) var1 var2 var3;

 do i=1 to 3;
  if numsamp[i] =. then numsamp[i] = 0;
 end;
```

DIM function

```
 dim (array-name)

 do i=1 to dim(numsamp);
  if numsamp[i] =. then numsamp[i] = 0;
 end;
```

SECTION II: Ingenious Uses of Arrays

FORMATTING MULTIPLE VARIABLES

VIEWTABLE: Work.Sample_sae1

	SUBJECT	_AESDTH	_AESLIFE	_AESHOSP	_AESDISAB	_AESCONG	_AESMIE
1	1001	0	1	1	0	0	0
2	1002	1	1	1	0	0	0
3	1003		0	1	0	0	

Decode 1 to ‘Y’

0 to ‘N’

VIEWTABLE: Work.Sample_sae2

	SUBJECT	AESDTH	AESLIFE	AESHOSP	AESDISAB	AESCONG	AESMIE
1	1001	N	Y	Y	N	N	N
2	1002	Y	Y	Y	N	N	N
3	1003		N	Y	N	N	

```
*Create array of input numeric variables;
array saen {*} _AESDTH _AESLIFE _AESHOSP _AESDISAB _AESCONG _AESMIE;
*Create array of output/formatted variables;
array saec {*} $1 AESDTH AESLIFE AESHOSP AESDISAB AESCONG AESMIE;

*Create loop to assign numeric value in formatted char value;
do i=1 to dim(saec);
  if saen{i} = 1 then saec{i} = 'Y’;
  else if saen{i} = 0 then saec{i} = 'N';
end;
```

FORMATTING MULTIPLE VARIABLES

Custom formats

```
proc format;
 value YN
   1 = 'Y'
   0 = 'N';
run;
```

```
do i=1 to dim(saec);
 if ~missing(saen{i}) then saec{i} = put(saen{i},yn.);
end;
```

SEARCHING MULTIPLE VARIABLES

VIEWTABLE: Work.Sample

	SUBJECT	FLAG1	FLAG2	FLAG3
1	AAA	Y	Y	Y
2	BBB	Y	N	
3	CCC	N	N	N

VIEWTABLE: Work.Sampleproc1

	SUBJECT	FLAG1	FLAG2	FLAG3	MISS_FLAG	N_FLAG
1	AAA	Y	Y	Y		
2	BBB	Y	N		Y	Y
3	CCC	N	N	N		Y

MISS_FLAG: when any Flag1 to Flag3 is missing
N_FLAG: when any Flag1 to Flag3 is “N”

```
array flag {3} $ flag1 flag2 flag3;

if ("") in flag then MISS_FLAG="Y";
if ("N") in flag then N_FLAG="Y";
```

Short Array Statements

```
array flag {3} $ flag1 flag2 flag3;

array flag {3} $ flag1-flag3;
array flag {*} $ flag:;
array flag {3} $;
```

Dynamic Element List using Macro Variables

```
proc sql noprint;
 select distinct NAME into :NEWFLAGS separated by " "
 from dictionary.columns
 where libname = "WORK" and memname = "SAMPLEPROC1" and
       upcase(NAME) contains "_FLAG";
quit;


array newfl {*} &newflags.;
```

CONCATENATION OF VALUES

VIEWTABLE: Work.Sample_recon1

	Subject	Date mismatch	Time mismatch	Visit mismatch
1	1001	Y	Y	Y
2	1002	Y	N	
3	1003	Y	N	Y

VIEWTABLE: Work.Sample_recon2

	Subject	Date mismatch	Time mismatch	Visit mismatch	MISS_N_FLAG
1	1001	Y	Y	Y	
2	1002	Y	N		Time mismatch, Visit mismatch
3	1003	Y	N	Y	Time mismatch

MISS_N_FLAG: will contain the variable labels of each flag variable with missing or 'N' values

```
length MISS_N_FLAG $200;

array flag{3} $;
array flagd{3} $100 _temporary_ ('Date mismatch' 'Time mismatch' 'Visit mismatch');

do i=1 to dim(flag);
  if flag{i} in ("" "N") then do;
    if missing(MISS_N_FLAG) then MISS_N_FLAG = flagd{i};
    else MISS_N_FLAG = strip(MISS_N_FLAG)||', '||flagd{i};
  end;
end;
```

Temporary array:
useful for holding constant values

APPLYING FUNCTIONS

VIEWTABLE: Work.Sample_macro1

	SUBJECT	WINDOWST	WINDOWEN	DATE1	DATE2	DATE3
1	1001	17MAR2017	05SEP2017	05SEP2017	01AUG2017	10FEB2017
2	1002	01APR2018	23MAY2018	02APR2018	17MAR2018	14AUG2018
3	1003	30DEC2016	12DEC2017	06SEP2017	26AUG2017	04NOV2017

VIEWTABLE: Work.Sample_macro2

	SUBJECT	WINDOWST	WINDOWEN	DATE1	DATE2	DATE3	MAXDT
1	1001	17MAR2017	05SEP2017	05SEP2017	01AUG2017		05SEP2017
2	1002	01APR2018	23MAY2018	02APR2018			02APR2018
3	1003	30DEC2016	12DEC2017	06SEP2017	26AUG2017	04NOV2017	04NOV2017

```
%macro within_window (date=);
  if not ((.<WINDOWST<=&date<=WINDOWEN) or
          (.<WINDOWST<=&date and missing(WINDOWEN))) then &date = .;
%mend within_window;

array date{3};
do i=1 to dim(date);
  %within_window(date=date{i});
end;
MAXDT=max(of date{*});
```

MATCH CORRESPONDING VALUES

VIEWTABLE: Work.Sample_os2

	SUBJECT	Withdrew consent date	Lost to follow-up date	Last contact date	MAXDT
1	1001	05SEP2017	01AUG2017		05SEP2017
2	1002	02APR2018			02APR2018
3	1003	06SEP2017	26AUG2017	04NOV2017	04NOV2017

VIEWTABLE: Work.Sample_os2

	SUBJECT	Withdrew consent date	Lost to follow-up date	Last contact date	MAXDT	CNSR	EVNTDESC
1	1001	05SEP2017	01AUG2017		05SEP2017	1	Withdrew consent
2	1002	02APR2018			02APR2018	1	Withdrew consent
3	1003	06SEP2017	26AUG2017	04NOV2017	04NOV2017	3	Ongoing

```
array date{3} date1 date2 date3;
array _cnsr{3} _temporary_ (1 2 3);
array _desc{3} $100 _temporary_
        ('Withdrew consent' 'Lost to follow-up' 'Ongoing');

  do j=1 to dim(date);
   if MAXDT = date{j} then do;
      CNSR=_cnsr{j};
      EVNTDESC=_desc{j};
   end;
  end;
```

CARRY OVER VALUES

VIEWTABLE: Work.Sample_score1

	SUBJECT	VISIT_SCR1	VISIT_SCR2	VISIT_SCR3	VISIT_SCR4
1	1001	3	3	4	5
2	1002	4	3	.	.
3	1003	5	.	.	.
4	1004	2	2	3	.
5	1005	.	1	2	.

VIEWTABLE: Work.Sample_score2

	SUBJECT	VISIT_SCR1	VISIT_SCR2	VISIT_SCR3	VISIT_SCR4
1	1001	3	3	4	5
2	1002	4	3	3	3
3	1003	5	5	5	5
4	1004	2	2	3	3
5	1005	.	1	2	2

```
array visit{*} visit_:;

do i=2 to dim(visit);
 if missing(visit) then visit{i}=visit{i-1};
end;
```

COMPLEX DATE DERIVATIONS

In clinical data, there are derivations that may require to check certain concomitant medications durations/doses.

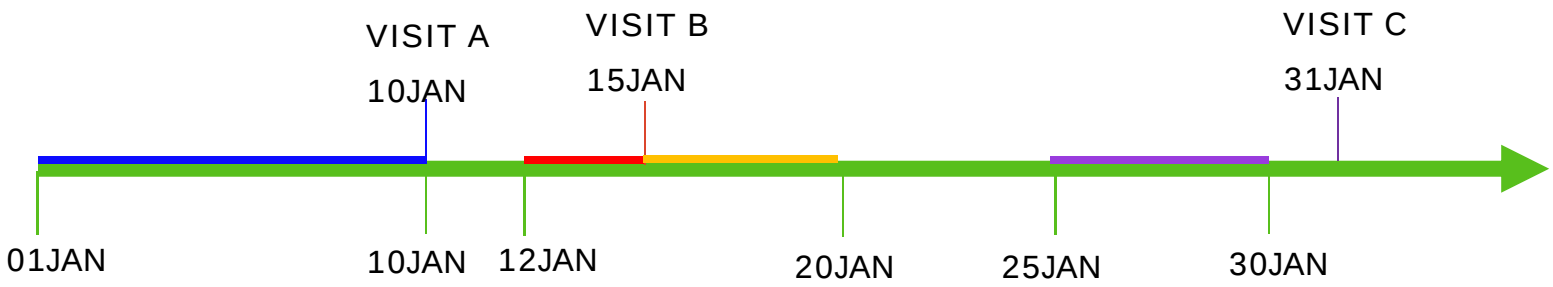
VIEWTABLE: Work.Sample

	SUBJECT	CMSTDT	CMENDT	SEQ
1	AAA	01JAN2018	10JAN2018	1
2	AAA	12JAN2018	20JAN2018	2
3	AAA	25JAN2018	30JAN2018	3

VIEWTABLE: Work.Visits

	SUBJECT	AVISIT	VSTDT
1	AAA	VISIT A	10JAN2018
2	AAA	VISIT B	15JAN2018
3	AAA	VISIT C	31JAN2018

The intent with these data is to calculate the cumulative dosing duration at VISIT X for Subject AAA.



VISIT A Cumulative Dosing duration = Blue Line

VISIT B Cumulative Dosing duration = Blue + Red Line

VISIT C Cumulative Dosing duration = Blue + Red + Yellow + Purple Line

COMPLEX DATE DERIVATIONS

VIEWTABLE: Work.Sample

	SUBJECT	CMSTDT	CMENDT	SEQ
1	AAA	01JAN2018	10JAN2018	1
2	AAA	12JAN2018	20JAN2018	2
3	AAA	25JAN2018	30JAN2018	3

We need to restructure the dataset to be compatible with using arrays

Transpose the sample dataset into a wide structure using SEQ as ID variable

VIEWTABLE: Work.Sample_wide

	SUBJECT	START_1	START_2	START_3	END_1	END_2	END_3
1	AAA	01JAN2018	12JAN2018	25JAN2018	10JAN2018	20JAN2018	30JAN2018

We merge the *sample_wide* dataset to *visits* dataset to get all the information in one dataset. (We only merge by subject variable)

VIEWTABLE: Work.Sample_array

	SUBJECT	START_1	START_2	START_3	END_1	END_2	END_3	AVISIT	VSTDT
1	AAA	01JAN2018	12JAN2018	25JAN2018	10JAN2018	20JAN2018	30JAN2018	VISIT A	10JAN2018
2	AAA	01JAN2018	12JAN2018	25JAN2018	10JAN2018	20JAN2018	30JAN2018	VISIT B	15JAN2018
3	AAA	01JAN2018	12JAN2018	25JAN2018	10JAN2018	20JAN2018	30JAN2018	VISIT C	31JAN2018

COMPLEX DATE DERIVATIONS

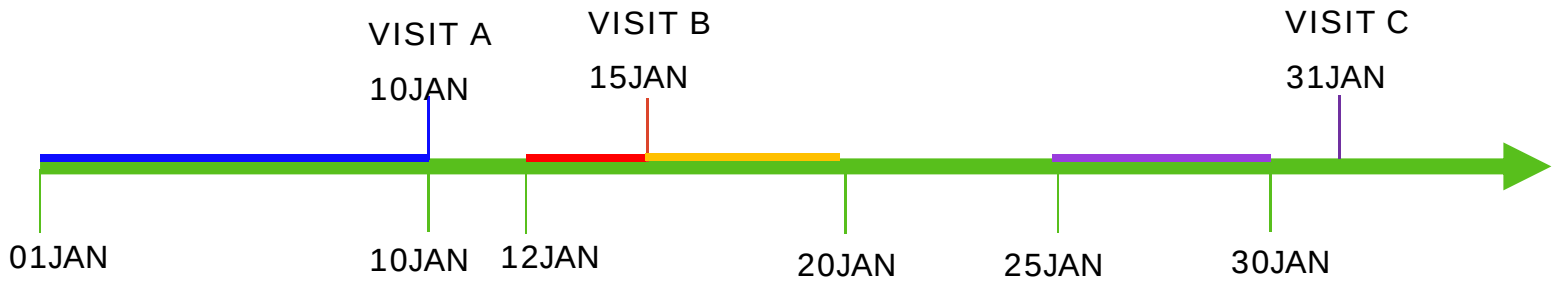
```
retain value 0; *Use a retained variable to sum up correctly.
if first.vstdt then value = 0; *Reset retained variable in next record

array start{*} start:;
array end{*} end:;

do i =1 to dim(start);
*Add duration as is when VSTDT >= ENDDATE
  if end[i] <= vstdt then value = value + (end[i]- start[i]+1);
*Change end date to VSTDT when in between doses
  else if end[i] > vstdt > start[i] then value = value + (vstdt - start[i]+1);
end;
```

VIEWTABLE: Work.Sample_process

	VALUE	SUBJECT	AVISIT	VSTDT
1	10	AAA	VISIT A	10JAN2018
2	14	AAA	VISIT B	15JAN2018
3	25	AAA	VISIT C	31JAN2018



SECTION III: Conclusion

CONCLUSION

- The convenient use of arrays in DATA step programming
  - can provide a more concise and efficient program codes
  - can enhance data manipulation techniques
- There is definitely more to know about ingenious applications of SAS arrays that programmers can benefit from

THANK YOU!

+ Name: Gladys Vanessa Espinas
  Organization: PPD INC.
  Address: 22F Net Park Building, Bonifacio Global City
  City, State ZIP: Taguig City, Philippines, 1634
  Work Phone: +63 2 689 6522
  E-mail: GladysVanessa.Espinas@ppdi.com
  Web: www.ppdi.com

+ Name: Eduard Joseph Siquioco
  Organization: PPD INC.
  Address: 22F Net Park Building, Bonifacio Global City
  City, State ZIP: Taguig City, Philippines, 1634
  Work Phone: +63 2 464 7675
  E-mail: EduardJoseph.Siquioco@ppdi.com
  Web: www.ppdi.com