**About the presenters**

**Andrew T. Kuligowski** is currently the Manager of CRM Data Infrastructure at HSN in St. Petersburg, FL. He has augmented his professional coding experiences in the retail, media, insurance, and petrochemical fields by speaking at various SAS conferences and user group meetings.

**Lisa Mendez, Ph.D.** currently works at IMS Government Solutions as a Sr. Consultant working with Military Health Systems data. She started using SAS in 1999 and is currently pursuing a Master’s Degree in Data Science at Southern Methodist University.
An Introduction to SAS® Arrays

Andrew T. Kuligowski, HSN
Lisa Mendez, Ph.D., IMSGS
Agenda

- Introductions & Overview
- The ARRAY statement
  - syntax
  - reference and bounds
  - compile time
- When / How to Use
  - Assorted scenarios demonstrating use
- Conclusion
SAS Arrays

- What IS an ARRAY / What ARE ARRAYS?
  - A temporary collection of SAS variables gathered and processed in a pre-arranged order
### Arrays

- Other software languages treat an array as an independent structure that is defined and holds values on its own.
- In SAS, an array is simply a collection of variables - those variables hold the values.

<table>
<thead>
<tr>
<th>State</th>
<th>Day 1</th>
<th>Day 2</th>
<th>...</th>
<th>Day 30</th>
<th>Day 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>1</td>
<td>37</td>
<td>17</td>
<td>...</td>
<td>8</td>
</tr>
<tr>
<td>Alaska</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>Arizona</td>
<td>3</td>
<td>29</td>
<td>15</td>
<td>...</td>
<td>30</td>
</tr>
<tr>
<td>Arkansas</td>
<td>4</td>
<td>18</td>
<td>7</td>
<td>...</td>
<td>9</td>
</tr>
<tr>
<td>Washington</td>
<td>47</td>
<td>52</td>
<td>13</td>
<td>...</td>
<td>21</td>
</tr>
<tr>
<td>West Virginia</td>
<td>48</td>
<td>8</td>
<td>6</td>
<td>...</td>
<td>4</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>49</td>
<td>48</td>
<td>52</td>
<td>...</td>
<td>38</td>
</tr>
<tr>
<td>Wyoming</td>
<td>50</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>...</th>
<th>Day 30</th>
<th>Day 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>State01Day01</td>
<td>State01Day02</td>
<td>...</td>
<td>State01Day30</td>
<td>State01Day31</td>
<td></td>
</tr>
<tr>
<td>Alaska</td>
<td>State02Day01</td>
<td>State02Day02</td>
<td>...</td>
<td>State02Day30</td>
<td>State02Day31</td>
<td></td>
</tr>
<tr>
<td>Arizona</td>
<td>State03Day01</td>
<td>State03Day02</td>
<td>...</td>
<td>State03Day30</td>
<td>State03Day31</td>
<td></td>
</tr>
<tr>
<td>Arkansas</td>
<td>State04Day01</td>
<td>State04Day02</td>
<td>...</td>
<td>State04Day30</td>
<td>State04Day31</td>
<td></td>
</tr>
<tr>
<td>Washington</td>
<td>State47Day01</td>
<td>State47Day02</td>
<td>...</td>
<td>State47Day30</td>
<td>State47Day31</td>
<td></td>
</tr>
<tr>
<td>West Virginia</td>
<td>State48Day01</td>
<td>State48Day02</td>
<td>...</td>
<td>State48Day30</td>
<td>State48Day31</td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td>State49Day01</td>
<td>State49Day02</td>
<td>...</td>
<td>State49Day30</td>
<td>State49Day31</td>
<td></td>
</tr>
<tr>
<td>Wyoming</td>
<td>State50Day01</td>
<td>State50Day02</td>
<td>...</td>
<td>State50Day30</td>
<td>State50Day31</td>
<td></td>
</tr>
</tbody>
</table>
The Array Statement

ARRAY array-name {n} <$> <length> array-elements[,] (initial values) ;

- **array** – SAS Keyword defining the statements
- **array-name** – Rules for naming arrays are the same as those for naming variables.
  - Arrays cannot have the same name as any variable defined to the active DATA step
- **Best Practice**
  - Array name similar to its elements
  - Do not use numbers in Array Name
**The Array Statement**

```
ARRAY array-name {n} <$$> <length> 
array-elements[,])(initial values) ;
```

- `{n}` - number of elements in array
  - Can use {brackets}, (parentheses), or [square brackets]
- `{n}` - can be a valid number (ex: 12)
- `{n}` - can be a range of numbers (ex: 1:12)
- `{n}` - can be an asterisk; SAS will count the elements
- Numbers separated by commas indicate a multidimensional array
The Array Statement

ARRAY array-name {n} <$> <length>
array-elements[,] (initial values) ;

- $ – Specifies that elements are character values
  - Optional, if components already specified as character; DO NOT USE if elements are numeric
- length – Specifies a length for new elements
  - optional
The Array Statement

ARRAY array-name {n} <$> <length>
array-elements[,] (initial values) ;

- **array-elements** – list of variables (in order) that make up array
  - `CHARACTER` will use all defined character variables
  - `NUMERIC` will use all defined numeric variables
  - `ALL` will use all defined variables (and they better all be of the same type)
  - `TEMPORARY` will create the necessary temporary variables to populate the array
The Array Statement

ARRAY array-name {n} <$> <length> array-elements[,,](initial values) ;

- array-elements is a list of variables (in order) that make up the array

- \_CHARACTER\_ will use all defined character variables

- \_NUMERIC\_ will use all defined numeric variables

- \_ALL\_ will use all defined variables and they better all be of the same type

- \_TEMPORARY\_ will create the necessary temporary variables to populate the array

If \( n \) is defined and variable list not provided, SAS will create variables named array-name1, array-name2, ... array-nameN
The Array Statement

ARRAY array-name {n} <$> <length>
array-elements[,,](initial values) ;

- **initial values** – values initially assigned to newly-created variables, in the order they are specified
The Array Statement at Compile Time

- The ARRAY statement is a “compile-time only” statement.
  - It cannot be conditionally executed (except via macro)
- Information is passed to the PDV (Program Data Vector), but then has nothing to actually execute
- Statement order is important!
Scenarios

- So, that’s what a SAS Array is. So What????
- We are now going to show some practical examples using arrays, to demonstrate their usefulness, as well as to show real-life syntax
**Do Loops**

- Wait, we need to discuss a quick side topic that goes well with arrays (like peanut butter and jelly)

- Basic syntax

```plaintext
DO <index-variable> = <start> TO <stop> BY <increment>;
    ... statement(s) ...
END;
```
**Do Loops**

```sas
DO <index-variable> = <start> TO <stop> BY <increment>;
   ... statement(s) ...
END;
```

- **Do** – SAS Keyword defining the statement, indicating begin point of loop
- **index-variable** - Valid SAS numeric variable that will contain incremented value
  - It **WILL** be written to the output dataset unless specified otherwise
- **END** – SAS keyword indicating termination of loop
Do Loops

DO <index-variable> = <start> TO <stop> BY <increment>;  
... statement(s) ...  
END;

- **start** - Numeric constant or valid SAS variable that denotes initial value of the index variable
- **stop** – Numeric constant or valid SAS variable that denotes final value of the index variable
- **increment** - Numeric constant or valid SAS variable that denotes the value that the index variable will be adjusted upon each iteration (default = 1)
Scenario 1: Convert Temperatures

/*------------------------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*------------------------------------------------------------*/

1. Data Patient_Temps_Converted (Drop = i);
2.   Set Patient_Temps;
   /* Declare arrays here */
3.   array temps {24} temp1-temp24;
4.   array temps_C {24} temp_C1-temp_C24;
   /* Do Loop - where the work is done */
5.   do i = 1 to 24;
6.      temps_C(i) = 5/9*(temps(i) - 32) ;
7.   end;
8. run;
Scenario 1: Convert Temperatures

/* Convert Fahrenheit Temps to Celsius Temps */

Data Patient_Temps_Converted (Drop = i);
Set Patient_Temps;
/* Declare arrays here */
array temps {24} temp1-temp24;
array temps_C {24} temp_C1-temp_C24;
/* Do Loop – where the work is done */
do i = 1 to 24;
    temps_C(i) = 5/9*(temps(i) - 32);
end;
run;

1. Create a data set, drop variable i
2. Read in records from SAS dataset
Scenario 1: Convert Temperatures

`/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*--------------------------------------------*/`

1. Data Patient_Temps_Converted (Drop = i);
2. Set Patient_Temps;
3. /* Declare arrays here */
4. array temps {24} temp1-temp24;
5. array temps_C {24} temp_C1-temp_C24;
6. /* Do Loop – where the work is done */
7. do i = 1 to 24;
8. temps_C(i) = 5/9*(temps(i) - 32) ;
9. end;
10. run;

- Declare arrays
- Each array contains 24 numeric elements
- “temps” holds existing variables, while “temps_C” contains new ones
Scenario 1: Convert Temperatures

/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*--------------------------------------------*/

1 Data Patient_Temps_Converted (Drop = i);
2 Set PatientTemps;
3 /* Declare arrays here */
4 array temps {24} temp1-temp24;
5 array temps_C {24} temp_C1-temp_C24;
6 /* Do Loop – where the work is done */
7 do i = 1 to 24;
8     temps_C(i) = 5/9*(temps(i) - 32) ;
9 end;
10 run;
11
5 Start of DO Loop
7 End of DO Loop
8 Execute DATA Step
Scenario 1: Convert Temperatures

/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps  */
/*--------------------------------------------*/

1. Data Patient_Temps_Converted (Drop = i);
2. Set Patient_Temps;
3. /* Declare arrays here */
4. array temps {24} temp1-temp24;
5. array temps_C {24} temp_C1-temp_C24;
6. /* Do Loop – where the work is done */
7. do i = 1 to 24;
8.   temps_C(i) = 5/9*(temps(i) - 32) ;
9. end;
10. run;

**Calculate Celsius temperature**

Obtain the \(i^{th}\) element of temps array. Perform requested calculation. Store result in \(i^{th}\) element of temps array.
Scenario 2: Convert Temperatures Daytime

/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*--------------------------------------------*/

1. Data Patient_Temps_Converted (Drop = i);
2. Set Patient_Temps;
3. /* Declare arrays here */
4. array temps {24} temp1-temp24;
5. array temps_C {24} temp_C1-temp_C24;
6. /* Do Loop - where the work is done */
7. do i = 1 to 24;
8.   temps_C(i) = 5/9*(temps(i) - 32) ;
9. end;
10. run;
Scenario 2: Convert Temperatures - Daytime

/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*--------------------------------------------*/

1. Data Patient_Temps_Converted (Drop = i);
2. Set Patient_Temps;
   /* Declare arrays here */
3. array temps {24} temp1-temp24;
4. array temps_C {24} temp_C1-temp_C24;
   /* Do Loop - where the work is done */
5. **do i = 1 to 24; do i = 6 to 18;**
6.     temps_C(i) = 5/9*(temps(i) - 32);
7. end;
8. run;

**Change bounds on DO Loop**
Scenario 2: Convert Temperatures Daytime

/*--------------------------------------------*/
/* Convert Fahrenheit Temps to Celsius Temps */
/*--------------------------------------------*/

Data Patient_Temps_Converted (Drop = i);
Set Patient_Temps;
/* Declare arrays here */
array temps {6:18} temp1-temp24;
array temps_C {6:18} temp_C1-temp_C24;
/* Do Loop – where the work is done */
do i = 6 to 18;

Each array contains 13 numeric elements. “temps” holds existing variables, while “temps_C” contains new ones.
Scenario 2: Convert Temperatures Daytime
/*/---------------------------------------------------------------*/
/*  Convert Fahrenheit Temps to Celsius Temps */
/*---------------------------------------------------------------*/
1 Data Patient_Temps_Converted (Drop = i);
2 Set Patient_Temps;
/* Declare arrays here */
3 array temps {13} temp6-temp18;
4 array temps_C {13} temp_C6-temp_C18;
/* Do Loop – where the work is done */
5 do i = 6 to 18;
6   temps_C(i) = 5/9*(temps(i) - 32) ;
7 end;
8 run;
This approach will work, but may make future maintenance unnecessarily challenging.
Scenario 2: Convert Temperatures Daytime

/* -------------------------------------------- */
/* Convert Fahrenheit Temps to Celsius Temps */
/* -------------------------------------------- */

1. Data Patient_Temps_Converted (Drop = i);
2. Set Patient_Temps;
3. /* Declare arrays here */
4. array temps {13} temp6-temp18;
5. array temps_C {13} temp_C6-temp_C18;
6. /* Do Loop – where the work is done */
7. do i = 6 to 18;
8.   temps_C(i) = 5/9*(temps(i) - 32);
9. end;
10. run;

Still must change Bounds on DO Loop.

This approach will work, but may make future maintenance unnecessarily challenging.

NOT RECOMMENDED!
Scenario 3: Increase Salaries by 20%

/*--------------------------------------------*/
/*     INCREASE SALARY VALUES BY 20 PERCENT  */
/*--------------------------------------------*/

DATA SALARY_INCREASED;
SET SALARY;
/* DECLARE ARRAY(S) HERE */
ARRAY SALARIES {5} ADMIN CLINICAL DIRECT_CARE_PROF DIRECT_CARE_PARAPROF RN ;
DO I = 1 TO 5;
IF SALARIES(I) GT 0 THEN
   SALARIES(I) = SALARIES(I) + (SALARIES(I)*.2);
END;
RUN;
Scenario 3: Increase Salaries by 20%

/*--------------------------------------------*/
/*   INCREASE SALARY VALUES BY 20 PERCENT     */
/*--------------------------------------------*/

DATA SALARY_INCREASED;
SET SALARY;
/* DECLARE ARRAY(S) HERE */
ARRAY SALARIES {5}  ADMIN  CLINICAL
                    DIRECT_CARE_PROF DIRECT_CARE_PARAPROF RN ;
DO I = 1 TO 5;
IF SALARIES(I) GT 0 THEN
  SALARIES(I) = SALARIES(I) + (SALARIES(I)*.2);
END;
RUN;

Declare array
Each array contains 5 numeric elements; variables named for job classifications.
Scenario 3: Increase Salaries by 20%

/* -------------------------------------------- */
/* INCREASE SALARY VALUES BY 20 PERCENT */
/* -------------------------------------------- */

1. DATA SALARY_INCREASED;
2. SET SALARY;
   /* DECLARE ARRAY(S) HERE */
3. ARRAY SALARIES {5} ADMIN CLINICAL DIRECT_CARE_PROF DIRECT_CARE_PARAPROF RN ;
4. DO I = 1 TO 5;
5.   IF SALARIES(I) GT 0 THEN
       SALARIES(I) = SALARIES(I) + (SALARIES(I)*.2);
6. END;
7. RUN;

5. Each element in the array is altered to contain a larger number
Scenario 4: Transpose a Record Structure

<table>
<thead>
<tr>
<th>Fiscal_Year</th>
<th>Fiscal_Month</th>
<th>Parent_DMIS_ID</th>
<th>Total_Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>1</td>
<td>117</td>
<td>11154</td>
</tr>
<tr>
<td>2011</td>
<td>2</td>
<td>117</td>
<td>37</td>
</tr>
<tr>
<td>2011</td>
<td>3</td>
<td>117</td>
<td>56620</td>
</tr>
<tr>
<td>2011</td>
<td>4</td>
<td>117</td>
<td>56124</td>
</tr>
<tr>
<td>2011</td>
<td>5</td>
<td>117</td>
<td>9378</td>
</tr>
<tr>
<td>2011</td>
<td>6</td>
<td>117</td>
<td>56865</td>
</tr>
<tr>
<td>2011</td>
<td>7</td>
<td>117</td>
<td>60563</td>
</tr>
<tr>
<td>2011</td>
<td>8</td>
<td>117</td>
<td>57490</td>
</tr>
<tr>
<td>2011</td>
<td>9</td>
<td>117</td>
<td>47023</td>
</tr>
<tr>
<td>2011</td>
<td>10</td>
<td>117</td>
<td>37</td>
</tr>
<tr>
<td>2011</td>
<td>11</td>
<td>117</td>
<td>56605</td>
</tr>
<tr>
<td>2011</td>
<td>12</td>
<td>117</td>
<td>63437</td>
</tr>
<tr>
<td>2011</td>
<td>13</td>
<td>117</td>
<td>56233</td>
</tr>
<tr>
<td>2011</td>
<td>14</td>
<td>117</td>
<td>50233</td>
</tr>
<tr>
<td>2011</td>
<td>15</td>
<td>117</td>
<td>50213</td>
</tr>
<tr>
<td>2011</td>
<td>16</td>
<td>117</td>
<td>11253</td>
</tr>
<tr>
<td>2011</td>
<td>17</td>
<td>117</td>
<td>52196</td>
</tr>
<tr>
<td>2011</td>
<td>18</td>
<td>117</td>
<td>50616</td>
</tr>
<tr>
<td>2011</td>
<td>19</td>
<td>117</td>
<td>57813</td>
</tr>
<tr>
<td>2011</td>
<td>20</td>
<td>117</td>
<td>45041</td>
</tr>
<tr>
<td>2011</td>
<td>21</td>
<td>117</td>
<td>11644</td>
</tr>
<tr>
<td>2011</td>
<td>22</td>
<td>117</td>
<td>61118</td>
</tr>
<tr>
<td>2011</td>
<td>23</td>
<td>117</td>
<td>53643</td>
</tr>
<tr>
<td>2011</td>
<td>24</td>
<td>117</td>
<td>53643</td>
</tr>
<tr>
<td>2011</td>
<td>25</td>
<td>117</td>
<td>53643</td>
</tr>
<tr>
<td>2011</td>
<td>26</td>
<td>117</td>
<td>9410</td>
</tr>
<tr>
<td>2011</td>
<td>27</td>
<td>117</td>
<td>9410</td>
</tr>
<tr>
<td>2011</td>
<td>28</td>
<td>117</td>
<td>9410</td>
</tr>
</tbody>
</table>

FROM

<table>
<thead>
<tr>
<th>Fiscal_Year</th>
<th>Parent_DMIS_ID</th>
<th>fm1</th>
<th>fm2</th>
<th>fm3</th>
<th>fm4</th>
<th>fm5</th>
<th>fm6</th>
<th>fm7</th>
<th>fm8</th>
<th>fm9</th>
<th>fm10</th>
<th>fm11</th>
<th>fm12</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>27</td>
<td>58200</td>
<td>50616</td>
<td>63231</td>
<td>65875</td>
<td>55843</td>
<td>69648</td>
<td>58124</td>
<td>57813</td>
<td>57490</td>
<td>47023</td>
<td>63985</td>
<td>46941</td>
</tr>
</tbody>
</table>

TO
Scenario 4: Transpose a Record Structure

```plaintext
/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */
DATA VISITS_TRANSPOSE (DROP = I);
  SET TOTAL_VISITS;
ARRAY FMONTHS {12} FM1-FM12;
DO I = 1 TO 12;
  IF FISCAL_MONTH = I THEN
    FMONTHS(I) = TOTAL_VISITS;
END;
RUN;

/* CONDENSE THE OBSERVATIONS */
PROC MEANS DATA = VISITS_TRANSPOSE NWAY NOPRINT;
CLASS FISCAL_YEAR PARENT_DMIS_ID;
OUTPUT OUT=VISITS_TRANSPOSE_2(DROP=_TYPE_ _FREQ_ FISCAL_MONTH TOTAL_VISITS)
SUM=;
RUN;
```
Scenario 4: Transpose a Record Structure

/* COUNT VISITS FOR EACH MONTH FOR EACH
CLINIC */

1. DATA VISITS_TRANSPOSE (DROP = I);
2. SET TOTAL_VISITS;
3. ARRAY FMONTHS {12} FM1-FM12;
4. DO I = 1 TO 12;
5. IF FISCAL_MONTH = I THEN
   FMONTHS(I) = TOTAL_VISITS;
6. END;

BEST PRACTICE: Be very judicious about naming variables after their datasets
Scenario 4: Transpose a Record Structure

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */

1. DATA VISITS_TRANSPOSE (DROP = I);
2. SET TOTAL_VISITS;
3. ARRAY FMONTHS {12} FM1-FM12;
4. DO I = 1 TO 12;
5.   IF FISCAL_MONTH = I THEN
       FMONTHS(I) = TOTAL_VISITS;
6. END;
7. TOT_VISITS;
Scenario 4: Transpose a Record Structure

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */

1. DATA VISITS_TRANSPOSE (DROP = I);
2. SET TOTAL_VISITS;
3. ARRAY FMONTHS {12} FM1-FM12;
4. DO I = 1 TO 12;
5.   IF FISCAL_MONTH = I THEN
       FMONTHS(I) = TOT_VISITS;
6. END;

The array contains 12 numeric elements corresponding to the 12 months

Declare array
Scenario 4: Transpose a Record Structure

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */

1. DATA VISITS_TRANSPOSE (DROP = I);
2. SET TOTAL_VISITS;
3. ARRAY FMONTHS {12} FM1-FM12;
4. DO I = 1 TO 12;
5. IF FISCAL_MONTH = I THEN
   FMONTHS(I) = TOT_VISITS;
6. END;

5. The array element corresponding to the MONTH variable is updated to store the VISITS value.
Scenario 4: Transpose a Record Structure

This is how your data will look after you update the VISITS value.

|   | fm | pdmisid | fcc4 | svc | skill | result | FM01 | FM02 | FM03 | FM04 | FM05 | FM06 | FM07 | FM08 | FM09 | FM10 | FM11 | FM12 | i  |
|---|----|---------|------|-----|-------|--------|------|------|------|------|------|------|------|------|------|------|------|-----|
| 1 | 01 | 0117    | BAAA | F   | 1     | aslte  | 69   | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 2 | 02 | 0117    | BAAA | F   | 1     | aslte  | .    | 68.2 | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 3 | 03 | 0117    | BAAA | F   | 1     | aslte  | .    | 68.05| .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 4 | 04 | 0117    | BAAA | F   | 1     | aslte  | .    | 68   | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 5 | 05 | 0117    | BAAA | F   | 1     | aslte  | .    | 64.7 | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 6 | 06 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | .    | 13  |
| 7 | 07 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | .    | 66   | .    | .    | .    | .    | 13  |
| 8 | 08 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | 65.5 | .    | .    | .    | .    | .    | 13  |
| 9 | 09 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | 91.5 | .    | .    | .    | .    | .    | 13  |
|10 | 10 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | 83.3 | .    | .    | .    | .    | .    | 13  |
|11 | 11 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | 84   | .    | .    | .    | .    | .    | 13  |
|12 | 12 | 0117    | BAAA | F   | 1     | aslte  | .    | .    | .    | .    | .    | .    | 80   | .    | .    | .    | .    | .    | 13  |
Scenario 4: Transpose a Record Structure

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */
DATA VISITS_TRANSPOSE (DROP = I);
SET TOTAL_VISITS;
ARRAY FMONTHS {12} FM1-FM12;
DO I = 1 TO 12;
   IF FISCAL_MONTH = I THEN
      FMONTHS(I) = TOT_VISITS;
END;
RUN;

/* CONDENSE THE OBSERVATIONS */
PROC MEANS DATA = VISITS_TRANSPOSE NWAY NOPRINT;
CLASS FISCAL_YEAR PARENT_DMIS_ID;
OUTPUT OUT=VISITS_TRANSPOSE_2(DROP=_TYPE_ _FREQ_ _FISCAL_MONTH TOTAL_VISITS) SUM=;
RUN;
Scenario 4: Transpose a Record Structure

/* CONDENSE THE OBSERVATIONS */

8 PROC MEANS DATA = VISITS_TRANSPOSE NWAY NOPRINT;
9   CLASS FISCAL_YEAR PARENT_DMIS_ID;
10 OUTPUT OUT=VISITS_TRANSPOSE_2
     (DROP=_TYPE_ _FREQ_ FISCAL_MONTH TOTAL_VISITS) SUM=;

RUN;

8 9 10

The array contents are summarized across records
Scenario 4: Transpose a Record Structure

This is how your data will look after you run Proc Means

<table>
<thead>
<tr>
<th></th>
<th>fy</th>
<th>pdmisi</th>
<th>svc</th>
<th>fcc4</th>
<th>result</th>
<th>skill</th>
<th>FM01</th>
<th>FM02</th>
<th>FM03</th>
<th>FM04</th>
<th>FM05</th>
<th>FM06</th>
<th>FM07</th>
<th>FM08</th>
<th>FM09</th>
<th>FM10</th>
<th>FM11</th>
<th>FM12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2008</td>
<td>0117</td>
<td>F</td>
<td>BAAA</td>
<td>asfte</td>
<td>1</td>
<td>69</td>
<td>68.15</td>
<td>68.05</td>
<td>68</td>
<td>64.7</td>
<td>.</td>
<td>66</td>
<td>55.5</td>
<td>91.6</td>
<td>83</td>
<td>84</td>
<td>80</td>
</tr>
</tbody>
</table>

Notice the missing value for FM06
Scenario 4(a1): Alternate Method

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */

1. DATA VISITS_TRANSPOSE (DROP = I) ;
2. SET TOTAL_VISITS;
3. ARRAY FMONTHS {12} FM1-FM12;
4. DO I = 1 TO 12;
5.   IF FISCAL_MONTH = I THEN
       FMONTHS(I) = TOT_VISITS;
     I = 13;   OR LEAVE;
   END;

EFFICIENCY: Leave the loop once corresponding element is identified

Otherwise, we always validate all 12 possible values in each iteration, even though MONTH can only be 1 of the 12
Scenario 4(a2): Another Alternate Method

/* COUNT VISITS FOR EACH MONTH FOR EACH CLINIC */

1 DATA VISITS_TRANSPOSE (DROP = I) ;
2 SET TOTAL_VISITS;
3 ARRAY FMONTHS {12} FM1-FM12;
4 DO I = 1 TO 12;
5   IF FISCAL_MONTH = I THEN
6     FMONTHS(FISCAL_MONTH) = TOT_VISITS;
7 END;

EFFICIENCY:
Don’t loop once corresponding element is identified

Only process 1 out of 12 possible values. However, there is NO VALIDATION – if FISCAL_MONTH = 13, the step aborts.
Scenario 5: Letting SAS Determine Subscript

/*----------------------------------------------------------*/
/* Count # months a procedure was performed */
/*----------------------------------------------------------*/

1. Data Month_Count (drop = i) ;
2. Set Visits_Transpose_2;
3. /* initialize cpt_count */
   FM_count = 0;
4. /* Declare array(s) here */
   array FMvalues {*} fm1-fm12;
5. /* Do Loop to create a count */
6. DO I = 1 TO 12;
7.   if FMvalues(i) ^= . then
      FM_count = FM_count + 1 ;
8. END;
9. RUN;
Scenario 5: Letting SAS Determine Subscript

/*------------------------------------------*/
/* Count # months a procedure was performed */
/*------------------------------------------*/

1. Data Month_Count (drop = i);
2. Set Visits_Transpose_2;
   /* Initialize cpt_count */
3. FM_count = 0;
   /* Declare array(s) here */
4. array FMvalues {*} fm1-fm12;
   /* Do Loop to create a count */
5. DO I = 1 TO 12;
6.   if FMvalues(i) ^= . then
          FM_count = FM_count + 1;
6.   END;
7. END;
8. RUN;

SAS will count the variables provided in the variable list, and will set the array size – in this instance to 12

If array element is not null, increment counter
Scenario 6: Multidimensional Array

/*---------------------------------*/
/* Accumulate day-by-day results */
/*---------------------------------*/

1. Data Month_Count (drop = i) ;
2. Set Visits_Transpose_2;
   /* Declare array(s) here */
   3. array DayResults {12,31} Day001-Day365;
   4. array DayPayment {12,31} Pay001-Day365;
   /* Do Loop to create a count */
   5. DayResults( Month(VisitDt), Day(VisitDt) ) + 1;
   6. DayPayment( Month(VisitDt), Day(VisitDt) ) =
      SUM(DayPayment( Month(VisitDt), Day(VisitDt) ),
      PatientBill );
7. RUN;
Scenario 6: Multidimensional Array

3. array DayResults {12,31} Day001-Day365;
4. array DayPayment {12,31} Pay001-Day365;

3 4 Declare 12 by 31 arrays (Corresponding to 12 months, with 31 days in longest months)
Scenario 6: Multidimensional Array

3 array DayResults {12,31} Day001-Day365; 366
4 array DayPayment {12,31} Pay001-Day365; 366

Leap Year
Scenario 6: Multidimensional Array

array DayResults {12,31} Day001-Day365,366
array DayPayment {12,31} Pay001-Day365,366

3 4 12 by 31 does not allow for flexible days per month

This means that the variable suffix will not correspond to the Julian date on / after march 1st
Scenario 6: Multidimensional Array

/*-------------------------------------------------------------*/
/* Accumulate day-by-day results */
/*-------------------------------------------------------------*/

1. Data Month_Count (drop = i);
2. Set Visits_Transpose_2;
3. /* Declare array(s) here */
4. array DayResults {12,31} Day001-Day365;
5. array DayPayment {12,31} Pay001-Day365;
6. /* Do Loop to create a count */
7. DayResults( Month(VisitDt), Day(VisitDt) ) + 1;
8. DayPayment( Month(VisitDt), Day(VisitDt) ) =
   SUM(DayPayment( Month(VisitDt), Day(VisitDt) ),
       PatientBill );

**Use functions to determine Array element Will by definition be 1-12 and 1-31, respectively**

**IMPORTANT SAFETY TIP:**
Do NOT give arrays the same name as SAS Functions
Scenario 7: Transpose Year/Day to Month/Day

1. Data Day_Results (drop = i);
2. ARRAY DAYRSLTS {12,31} DAYKEEP001-DAYKEEP372;
3. ARRAY MONTHDAYS [12] _TEMPORARY_
   ( 31 29 31 30 31 30 31 31 30 31 30 31 );
4. RETAIN MONTH DAY 1;
5. DO I = 1 TO 366;
6.   DAYRSLTS{MONTH,DAY} = FLOOR(RANUNI(0)*100) +1;
7.   IF DAY = MONTHDAYS( MONTH ) THEN DO;
8.     MONTH = MONTH + 1;
9.     DAY = 1;
10.    END;
11.   ELSE DAY = DAY + 1;
12. END;
13. RUN;
Scenario 7: Transpose Year/Day to Month/Day

1. Data Day_Results (drop = i);
2. ARRAY DAYRSLTS {12,31} DAYKEEP001-DAYKEEP372;
3. ARRAY MONTHDAYS [12] _TEMPORARY_
   ( 31 29 31 30 31 30 31 31 30 31 30 31 );
4. RETAIN MONTH DAY 1;
5. DO I = 1 TO 366;
6.   DAYRSLTS{MONTH,DAY} = FLOOR(RANUNI(0)*100) +1;
7.   IF DAY = MONTHDAYS( MONTH ) THEN DO;
8.     MONTH = MONTH + 1;
9.     DAY = 1;
10.   END;
11. ELSE DAY = DAY + 1;
12. END;
13. RUN;
Scenario 7: Transpose Year/Day to Month/Day

1. Data Day_Results (drop = i);
2. ARRAY DAYRSLTS {12,31} DAYKEEP001-DAYKEEP372;
3. ARRAY MONTHDAYS [12] TEMPORARY (31 29 31 30 31 30 31 31 30 31 30 31);
4. RETAIN MONTH DAY 1;
5. DO I = 1 TO 366;
6. DAYRSLTS{MONTH, DAY} = FLOOR(RANUNI(0)*100) + 1;
7. IF DAY = MONTHDAYS(MONTH) THEN DO;
8. MONTH = MONTH + 1;
9. DAY = 1;
10. END;
11. ELSE DAY = DAY + 1;
12. END;
13. RUN;

Temporary Array - Elements not specified nor written to permanent dataset
Scenario 7: Transpose Year/Day to Month/Day

1. Data Day_Results (drop = i);
2. ARRAY DAYRSLTS {12,31} DAYKEEP001-DAYKEEP372;
3. ARRAY MONTHDAYS [12] _TEMPORARY_
   ( 31 29 31 30 31 30 31 31 30 31 30 31 );
4. RETAIN MONTH DAY 1;
5. DO I = 1 TO 366;
6.   DAYRSLTS{MONTH,DAY} = FLOOR(RANUNI(0)*100) +1;
7.   IF DAY = MONTHDAYS (MONTH) THEN DO;
8.     MONTH = MONTH + 1;
9.     DAY   = 1;
10.   END;
11. ELSE DAY = DAY + 1
12. END;
13. RUN;

Note that different brackets may be interchanged when referring to same array.
Scenario 7: Transpose Year/Day to Month/Day

NOTE: THE DATA SET WORK.DAY_RESULTS HAS 1 OBSERVATIONS AND 374 VARIABLES.

Gap – undefined values between values #60 and #63 – which correspond to Julian Day 60 (Feb 29<sup>th</sup>) and 63 (Mar 1<sup>st</sup>).

374 Variables. Did <i>not</i> keep the 12 elements of MONTHDAYS, <i>BUT</i> we forgot to drop indexes Month and Day, explaining the 2 extras.
Other Real World Examples

The inclusion of this picture is not meant to promote illegal wagering!
Sudoku is basically a 9x9 array...with 9 sets of 3x3 arrays embedded within.
Other Related Topics

- VNAME( ) function
- DIM( ) function
- OF operator
- IN operator
VNAME( ) function

- **VNAME (parameter)**
  - Returns name of specified variable parameter

- **CALL VNAME (param1, param2)**
  - Stores variable name of param1 into param2
  - Remember to allow for 32 characters
array DayResultsKeep {12,31}
DayKeep001-DayKeep372;
array MonthDays (12)
_TEMPORARY_ ( 31 29 31 30
 31 30 31 30 31 30 31 );
...
DO I = 1 TO 12;
  VarName = VNAME(DayResultsKeep( I, 1 ));
  PUTLOG "INFO: " Varname= ;
END;
...
VarName = VNAME( I );
PUTLOG "INFO: " Varname= ;
...
INFO: VarName=DayKeep001
INFO: VarName=DayKeep032
INFO: VarName=DayKeep063
INFO: VarName=DayKeep094
INFO: VarName=DayKeep125
INFO: VarName=DayKeep156
INFO: VarName=DayKeep187
INFO: VarName=DayKeep218
INFO: VarName=DayKeep249
INFO: VarName=DayKeep280
INFO: VarName=DayKeep311
INFO: VarName=DayKeep342
INFO: VarName=I
Other Related Topics - VNAME( ) function

```sas
... array DayResultsKeep {12,31}  
  DayKeep001-DayKeep372;  
array MonthDays (12) _TEMPORARY_ ( 31 29 31 30  
  31 30 31 31 30 31 30 31 );  
... DO I = 1 TO 12;  
  VarName = VNAME(  
    DayResultsKeep( I, 1 ) );  
  PUTLOG "INFO: " Varname= ;  
END;  
... VarName = VNAME( I );  
PUTLOG "INFO: " Varname= ;  
...
```

ERROR: Array elements of _TEMPORARY_ arrays are not defined as variables and therefore do NOT have NAMES or LABELS.

NOTE: The SAS System stopped processing this step because of errors.

WARNING: The data set WORK.DAY_RESULTS may be incomplete. When this step was stopped there...
Other Related Topics

**DIM( ) function**

- **DIM(array-name)**
  - Returns number of elements in array-name

- **DIM2 ... DIMx**
  - Same function used to obtain corresponding dimension when using multidimensional arrays
Other Related Topics

DIM( ) function

- In addition
  - HBOUND (array-name)
    » Uppermost (Highest) boundary of array
  - LBOUND (array-name)
    » Lowest boundary of array
Other Related Topics – DIM() function

array DayResultsKeep {12,31} 
DayKeep001-DayKeep372;

DRK_Size =
   DIM(DayResultsKeep);
DRK_HB =
   HBOUND(DayResultsKeep);
DRK_LB =
   LBOUND(DayResultsKeep);
PUTLOG "INFO: " DRK_Size=
   "INFO: " DRK_HB=
   "INFO: " DRK_LB=

INFO: DRK_Size=12
INFO: DRK_HB=12
INFO: DRK_LB=1
Other Related Topics – DIM ( ) function

... array DayResultsKeep {12,31}  
    DayKeep001-DayKeep372;
...

  DRK_Size =  
    DIM2( DayResultsKeep );
  DRK_HB =  
    HBOUND2( DayResultsKeep );
  DRK_LB =  
    LBOUND2( DayResultsKeep );
  PUTLOG "INFO: " DRK_Size= 
    "INFO: " DRK_HB= 
    "INFO: " DRK_LB= ;
...

INFO: DRK_Size=31
INFO: DRK_HB=31
INFO: DRK_LB=1
Other Related Topics – DIM ( ) function

array DayResultsKeep {12,31}
  DayKeep001-DayKeep372;
...
DO I = 1 TO 2;
  DRK_Size =
    DIM(DayResultsKeep,I);
  DRK_HB   =
    HBOUND(DayResultsKeep,I);
  DRK_LB   =
    LBOUND(DayResultsKeep,I);
  PUTLOG "INFO: " DRK_Size=
    "INFO: " DRK_HB=
    "INFO: " DRK_LB=
...
INFO: DRK_Size=12
INFO: DRK_HB=12
INFO: DRK_LB=1
INFO: DRK_Size=31
INFO: DRK_HB=31
INFO: DRK_LB=1

This works for _Temporary_ arrays as well
Other Related Topics

OF operator

- Specifies a set of variables to be processed in function
Other Related Topics - OF operator

array DayResultsKeep \{12,31\}
  DayKeep001-DayKeep372;

YearSum1 = SUM(
  OF DayResultsKeep(*)
);
YearSum2 = SUM(
  OF DayKeep001-DayKeep372);

array DayResultsKeep \{12,31\}
  DayKeep001-DayKeep372;

YearSum3 = SUM(
  OF DayResultsKeep
);
ERROR: Illegal reference to the array DayResultsKeep.
array DayResultsKeep {12,31}
  DayKeep001-DayKeep372;

... YearSum1 = SUM(
  OF DayResultsKeep(*));
YearSum2 = SUM(
  OF DayKeep001-DayKeep372);
...

array DayResultsKeep {12,31}
  DayKeep001-DayKeep372;
...

YearSum4 = SUM(
  OF DayResultsKeep(*,*) );
  
- 22
  200

ERROR 22-322: Syntax error, expecting one of the following: ), ], }. ERROR 200-322: The symbol is not recognized and will be ignored.
array DayResultsKeep {12,31}
  DayKeep001-DayKeep372;

YearSum1 = SUM(
  OF DayResultsKeep(*));
YearSum2 = SUM(
  OF DayKeep001-DayKeep372);

array DayResultsKeep {12,31}
  DayKeep001-DayKeep372;

YearSum4 = SUM(
  OF DayResultsKeep(*,*) );

  - 22
    200

ERROR 22-322: Syntax error, expecting one of the following: ), ], }.  
ERROR 200-322: The symbol is not recognized and will be ignored.
WHICHC Search for `<STRING>`, return index of first matching value.

WHICHN Search for `<NUMERIC>`, return index of first matching value.

Data temp;
array MudvilleNine (9) $ 8
("Cooney" "Barrows"
 "Flynn" "Blake" "Casey"
 "Unknown" "Unknown"
 "Unknown" "Unknown" );
OurHero = WHICHC( "Casey", OF MudvilleNine (*) );
PUTLOG OurHero= ;
run;
OurHero=5
Other Related Topics

**IN operator**

- Search through a range for a specified value
Other Related Topics - IN operator

Data temp;
  array MudvilleNine (9) $ 8
    ("Cooney" "Barrows"
     "Flynn" "Blake" "Casey"
     "Unknown" "Unknown"
     "Unknown" "Unknown" );
  IF "Casey" IN MudvilleNine
    THEN PUTLOG "Our Hero!";
run;

Our Hero!
Advanced Topics – For Future Discussion

- What if you have a large array in which most elements are never populated?
- Memory and disk issues if you build that large array – whether or not you ever load the values, the placeholders are there
- Look up “Sparse Matrix” online (possible future paper – if interest)
Conclusion

- Quick overview of SAS Arrays
- Learn by watching
- Multiple ways to do something, multiple “right” ways to do something, which one(s) best for you?
- Balance machine efficiency with programmer efficiency
Contact Information

For further information, feel free to contact us

Andrew:  KuligowskiConference@gmail.com

Lisa:  MendezLa@sbcglobal.net