Paper 5720-2016

Arrays and DO Loops: Applications to Health-Care Diagnosis Fields

Ryan Ferland, Blue Cross Blue Shield of Arizona

ABSTRACT

Arrays and Do loops have been used for years by SAS® programmers who work with diagnosis fields in health-care data, and the opportunity to use these techniques has only grown since the launch of the Affordable Care Act (ACA) in the United States. Users new to SAS or to the health-care field may find an overview of existing (as well as new) applications helpful. Risk adjustment software, including the publicly available Health and Human Services (HHS) risk software that uses SAS and was released as part of the ACA implementation, is one example of code that is significantly improved by the use of arrays. Similar projects might include evaluations of diagnostic persistence, comparisons of diagnostic frequency or intensity between providers, and checks for unusual clusters of diagnosed conditions. This session reviews examples suitable for intermediate SAS users, including the application of two-dimensional arrays to diagnosis fields.

INTRODUCTION

Arrays are a useful tool for the SAS programmer and health-care analyst. Like any other programming tool, one programmer may use arrays extensively, while another may have a long and productive career with minimal use of arrays. I generally find it helpful to have as many tools as possible near to hand when working on a project.

This paper looks at arrays with particular application to diagnosis codes; this application has been of interest to SAS programmers for years, as the papers in the reference section show. We begin without arrays, introduce arrays, use arrays in a clearly appropriate piece of code, then show a use of arrays that appears at first unnecessary until the next steps in the analysis are considered.

Reviewing this paper and the references included should enable a SAS programmer new to health-care data or to arrays begin to modify or construct code related to diagnosis fields.

BASICS ON DIAGNOSIS FIELDS AND ARRAYS

REGARDING DIAGNOSIS FIELDS

Background

Our examples today draw on diagnosis fields in healthcare data. These fields are typically populated with alphanumeric codes based on underlying research done by the World Health Organization. This being the modern world, every illness, condition, injury or complaint has an associated code. For this paper we will use some dummy codes, defined as follows:

A100 is bronchitis. (A Respiratory diagnosis.)

A304 is pneumonia. (Respiratory)

A503 is asthma. (Respiratory)

B173 is chickenpox. (Infectious Disease)

R001-R009 are stubbed toes, with each code corresponding to a different toe. (Musculoskeletal)

R202 is arthritis of the right knee. (Musculoskeletal)

R205 is instability of the right knee. (Musculoskeletal)

S101 is an enlarged heart. (Circulatory)

S102 is a broken heart. (Circulatory)

Example of Visits and Diagnoses

When Patient A visits a healthcare provider (doctor, clinic, hospital, etc.) for chickenpox on July 1, the provider may document that Patient A has a bronchitis, asthma, chickenpox, and an enlarged heart.

When that same Patient A goes to a provider on July 3 for an ultrasound to better evaluate the enlarged heart, the only recorded diagnosis may be the enlarged heart even though the patient still has asthma and likely still suffers from the chickenpox.

When Patient A visits still another provider on December 1 for a stubbed right big toe, the provider may diagnose the toe and the enlarged heart (but not the asthma).

Finally, when Patient A has a physical on February 3 of the following year, the provider may document asthma, the enlarged heart, and a broken heart following a 5th-grade breakup.

To get a complete picture of a patient's diagnosed conditions, it may therefore be necessary to look at every visit over some period of time (rows) and to look at every diagnosis documented for each visit (columns).

Table 1 shows the diagnosis codes documented on each of Patient A's visits.

Patient ID	ClaimID	Date	DgCd1	DgCd2	DgCd3	DgCd4
A	111	7-1-15	A100	A503	B173	S101
A	112	7-3-15	S101			
A	113	12-1-15	R005	S101		
A	114	2-3-16	A503	S101	S102	

Table 1. Patient A Claims

Our basic goal is to organize the information for Patient A onto fewer rows and into fewer columns.

REGARDING ARRAYS

Arrays allow the SAS user to perform identical operations on multiple fields with relatively few lines of code. In SAS, array declarations do not allocate memory or resources as in some programming languages. Arrays simply organize SAS variables to allow more compact programs, which is a powerful benefit.

If we have four variables already set, we can organize them into an array as follows:

```
array AllDiagCodes {4} $ DgCd1 DgCd2 DgCd3 DgCd4;
```

where:

ARRAY is the statement;

AllDiagCodes is the name of the array;

{4} indicates that the array contains four variables, while {4,2} would indicate 8 variables arranged in two rows and four columns;

$ indicates that the variables are character strings, and if this were omitted, they would be numbers;

DgCd1 DgCd2 DgCd3 DgCd4 lists the included variables, and could also be written as DgCd1 – DgCd4 since the variable names end with sequential numbers.

DIAGNOSIS FIELDS AND ARRAYS

PART ONE: WITH AND WITHOUT AN ARRAY

We can generate an indicator that any of up to 25 diagnosis codes on a claim was related to respiratory issues. That is, we can check for codes A100, A304, or A503. Doing so without an array might look like this:

```
data ExampleA1;
  set TableA;
    RespInd = 0;
    if DgCd1 IN ('A100','A304','A503') then
        RespInd = 1;
   else if DgCd2 IN ('A100','A304','A503') then
        RespInd = 1;
    . . .
    . . .
   else if DgCd25 IN ('A100','A304','A503') then
        RespInd = 1;
run;
```

Table 2 shows claims for Patients A-C and flags lines with a Respiratory diagnosis.

Patient ID	ClaimID	ClaimDt	DgCd1	DgCd2	DgCd3	DgCd4	DgCd5-25	RespInd
A	111	7-1-15	A100	A503	B173	S101		1
A	112	7-3-15	S101					0
A	113	12-1-15	R005	S101				0
A	114	2-3-16	A503	S101	S102			1
B	115	10-10-14	A100	R007	R008	R009		1
B	116	1-15-16	S102					0
C	117	10-10-14	S101	A304				1
C	118	3-10-15	S101					0
C	119	5-5-16	A100					1

Table 2. Diagnosis Codes and a Respiratory Indicator

The lengthy if-else sequence in the code above requires quite a bit of typing or copying. Using an array removes the repeated text.

```
data ExampleA2;
  set TableA;
    array AllDiagCodes {25} $ DgCd1 - DgCd25;
    RespInd = 0;
    do i = 1 to 25;
       if AllDiagCodes{i} IN ('A100','A304','A503')
          then RespInd = 1;
    end;
    drop i;
run;
```

Table 3. This output is identical to the previous output.

Patient ID	ClaimID	ClaimDt	DgCd1	DgCd2	DgCd3	DgCd4	DgCd5-25	RespInd
A	111	7-1-15	A100	A503	B173	S101		1
A	112	7-3-15	S101					0
A	113	12-1-15	R005	S101				0
A	114	2-3-16	A503	S101	S102			1
B	115	10-10-14	A100	R007	R008	R009		1
B	116	1-15-16	S102					0
C	117	10-10-14	S101	A304				1
C	118	3-10-15	S101					0
C	119	5-5-16	A100					1

Table 3. Diagnosis Codes and a Respiratory Indicator

A look at our set of diagnosis codes shows that all the Respiratory codes start with 'A'. We can use the SUBSTR function to shorten the code.

So:

```
if AllDiagCodes{i} IN ('A100','A304','A503')
 then RespInd = 1;
```

Becomes:

```
format FirstDigitDgCd $1.;
FirstDigitDgCd = substr(AllDiagCodes{i},1,1);
if FirstDigitDgCd eq 'A' then RespInd = 1;
```

And appears in the complete code as:

```
data ExampleA3;
  set TableA;
    array AllDiagCodes {25} $ DiagCd1 - DiagCd25;
    RespiratoryInd = 0;
    format FirstDigitDgCd $1.;
    do i = 1 to 25;
       FirstDigitDgCd = substr(AllDiagCodes{i},1,1);
       if FirstDigitDgCd eq 'A' then RespInd = 1;
    end;
    drop i FirstDigitDgCd;
run;
```

The resulting table would be unchanged from the prior runs.

PART TWO: SETTING MULTIPLE INDICATORS WITH AND WITHOUT ARRAYS

We can go further. If we want to set four different indicators – RespInd, MuscInd, InfectInd, and CircInd – we can write each one within the DO loop:

```
data ExampleA4;
  set TableA;
    array AllDiagCodes {25} $ DiagCd1 - DiagCd25;
    RespInd = 0;
    InfectInd = 0;
    MuscInd = 0;
    CircInd = 0;
    do i = 1 to 25;
       format FirstDigitDgCd $1.;
       FirstDigitDgCd = substr(AllDiagCodes{i},1,1);
       if FirstDigitDgCd eq 'A' then RespInd = 1;
       else if FirstDigitDgCd) eq 'B' then InfectInd = 1;
       else if FirstDigitDgCd eq 'R' then MuscInd = 1;
       else if FirstDigitDgCd eq 'S' then CircInd = 1;
    end;
    drop i FirstDigitDgCd;
run;
```

This code works well enough and is fairly compact.

Table 4 shows diagnosis history and four indicators.

Patient ID	Claim ID	ClaimDt	DgCd1	DgCd2	DgCd3	DgCd4	DgCd5-25	Resp Ind	Infect Ind	Musc Ind	Circ Ind
A	111	7-1-15	A100	A503	B173	S101		1	1	0	1
A	112	7-3-15	S101					0	0	0	1
A	113	12-1-15	R005	S101				0	0	1	1
A	114	2-3-16	A503	S101	S102			1	0	0	1
B	115	10-10-14	A100	R007	R008	R009		1	0	1	0
B	116	1-15-16	S102					0	0	0	1
C	117	10-10-14	S101	A304				1	0	0	1
C	118	3-10-15	S101					0	0	0	1
C	119	5-5-16	A100					1	0	0	0

Table 4. Now Four Indicators are Populated

Let us further suppose that we have 80 different indicators that we are working to populate rather than 4. That would be 80 variables we would need to initialize to 0 and 80 lines of code anytime we wanted to perform any task on those indicators in any DATA step.

Placing the indicators in an array would then be helpful and permit us to use data steps in do loops.

First, though, we must set up macro variables that can be accessed from within the do loop. This will take 4 rows (or 80, if we had 80 indicators) but will then permit us to use do loops in subsequent steps:

```
%let RespCodes = 'A';
%let InfectCodes = 'B';
%let MuscCodes = 'R';
%let CircCodes = 'S';
```

```
data ExampleA5;
  set TableA;

    array AllDiagRanges {4} $ &RespCodes &InfectCodes
          &MuscCodes &CircCodes;

    array AllDiagIndicators {4} RespInd InfectInd
          MuscInd CircInd;

    array AllDiagCodes {25} $ DgCd1 - DgCd25;

    do i = to 4;
       AllDiagIndicators{i} = 0;
    end;

    format FirstDigitDgCd $1.;
    do i = 1 to 25;
       do j = 1 to 4;
          FirstDigitDgCd = substr(AllDiagCodes{i},1,1);
          if FirstDigitDgCd eq AllDiagRanges{j}
             then AllDiagIndicators{j} = 1;
       end;
    end;
    drop i j FirstDigitDgCd;
run;
```

The output does not change from the previous table. While there is more setup required to use arrays, the heart of the data step requires much less code.

Experience in other programming languages with arrays may be a distraction with SAS arrays. Note that we have not yet generated a two-dimensional matrix despite the nested do loops on i and j.

PART THREE: SUMMARIZING

We can now collapse our data set to a single row per Patient. This can be done via a variety of methods in SAS. We will use PROC SORT and PROC MEANS.

```
proc sort data = ExampleA5;
   by PatientID;
run;

proc means noprint data = ExampleA5;
   by PatientID;
   output out = SummaryA5
   (drop = _TYPE_ _FREQ_)
   max(RespInd) = RespInd
   max(CircInd) = CircInd
   max(InfectInd) = InfectInd
   max(MuscInd) = MuscInd;
run;
```

Table 5 has one row per patient.

Patient ID	Resp Ind	Infect Ind	Musc Ind	Circ Ind
A	1	1	1	1
B	1	0	1	1
C	1	0	0	1

Table 5. One Row Per Patient

We could also include the number of claims with the respective type of diagnosis.

```
proc sort data = ExampleA5;
   by PatientID;
run;

proc means noprint data = ExampleA5;
   by PatientID;
   output out = SummaryA5
   (drop = _TYPE_ _FREQ_)
   max(RespInd) = RespInd
   max(InfectInd) = InfectInd
   max(MuscInd) = MuscInd
   max(CircInd) = CircInd
   sum(RespInd) = RespCount
   sum(InfectInd) = InfectCount
   sum(MuscInd) = MuscCount
   sum(CircInd) = CircCount;
run;
```

Table 6.

Patient ID	Resp Ind	Infect Ind	Musc Ind	Circ Ind	Resp Count	Infect Count	Musc Count	Circ Count
A	1	1	1	1	2	1	1	4
B	1	0	1	1	1	0	1	1
C	1	0	0	1	2	0	0	2

Table 6. Indicators and Counts by Patient

PART FOUR: USING A TWO-DIMENSIONAL ARRAY

So far, we've done some reasonable work. Let us suppose, though, that we want to take further steps. We may want to track these indicators by patient and by year, and two-dimensional arrays will be useful.

Keep in mind that the array simply organizes the variables to assist the programmer in writing code. The array does not create any physical two-dimensional data structure. The variables all remain within the same row of the SAS table, as usual, and the program continues through the program data vector one row at a time as is customary.

To group the data by year and patient:

```
%let RespCodes = 'A';
%let InfectCodes = 'B';
%let MuscCodes = 'R';
%let CircCodes = 'S';
```

```
data ExampleA6;
  set TableA;

    array AllDiagRanges {4} $ &RespCodes &InfectCodes
          &MuscCodes &CircCodes;

    array AllDiagIndicators {3, 4}
          RespIndYr1 - RespIndYr3
          InfectIndYr1 - InfectIndYr3
          MuscIndYr1 - MuscIndYr3
          CircIndYr1 - CircIndYr3
          ;

    array AllDiagCodes {25} $ DgCd1 - DgCd25;

    do i = 1 to 3;
       do j = 1 to 4;
          AllDiagIndicators{i,j} = 0;
       end;
    end;

    ClaimYr = year(ClaimDt);
    format FirstDigitDgCd $1.;
    if ClaimYr eq 2014 then k = 1;
    else if ClaimYr eq 2015 then k = 2;
    else if ClaimYr eq 2016 then k = 3;

    do i = 1 to 25;
       do j = 1 to 4;
          FirstDigitDgCd = substr(AllDiagCodes{i},1,1);
          if FirstDigitDgCd eq AllDiagRanges{j}
             then AllDiagIndicators{k,j} = 1; *Populated the 2D array;
       end;
    end;
    drop i j k FirstDigitDgCd ClaimYr;
run;
```

Table 7 shows diagnosis history and four indicators, now split by year.

Patient ID	Claim ID	ClaimDt	DgCd 1-25	Resp IndYr1	Resp IndYr2	Resp IndYr3	Other Indicators
A	111	7-1-15		0	1	0	
A	112	7-3-15		0	0	0	
A	113	12-1-15		0	0	0	
A	114	2-3-16		0	0	1	
B	115	10-10-14		1	0	0	
B	116	1-15-16		0	0	0	
C	117	10-10-14		1	0	0	
C	118	3-10-15		0	0	0	
C	119	5-5-16		0	0	1	

Table 7. Indicators Split By Year

PART FIVE: SUMMARIZING TWO DIMENSIONS IN ONE ROW

And we can summarize by year, as well as patient:

```
proc sort data = ExampleA6;
   by PatientID;
run;

proc means noprint data = ExampleA6;
   by PatientID;
   output out = SummaryA6
   (drop = _TYPE_ _FREQ_)
   max(RespIndYr1) = RespIndYr1
   max(RespIndYr2) = RespIndYr2
   max(RespIndYr3) = RespIndYr3
   max(InfectIndYr1) = InfectIndYr1
   max(InfectIndYr2) = InfectIndYr2
   max(InfectIndYr3) = InfectIndYr3
   max(MuscIndYr1) = MuscIndYr1
   max(MuscIndYr2) = MuscIndYr2
   max(MuscIndYr3) = MuscIndYr3;
   max(CircIndYr1) = CircIndYr1
   max(CircIndYr2) = CircIndYr2
   max(CircIndYr3) = CircIndYr3
run;
```

Table 8.

Patient ID	Resp IndYr1	Resp IndYr2	Resp IndYr3	Infect IndYr1	Infect IndYr2	Infect IndYr3	Other Indicators
A	0	1	1	0	1	0	
B	1	0	0	0	0	0	
C	1	0	1	0	0	0	

Table 8. Patient Indicators by Year

The task of grouping diagnosis indicators by year could also have been accomplished by summarizing on PatientID and on ClaimYr, yielding three rows for each PatientID. That would have actually taken less code. The advantage to the approach taken here is that it is now easy to perform operations that compare the years for the same patient. If the years were each on their own row, you might need to use LAG, RETAIN, TRANSPOSE or other SAS concepts in order to compare them.

PART SIX: ON YOUR OWN

If we had summarized across 36 months rather than across three years, we could then use arrays to perform operations across the 36 variables for each indicator. I leave it to you to consider the potential uses of this and similar manipulations.

ANOTHER IMPLEMENTATION

An alternative approach that also uses arrays and is generally more advanced (and robust) than can be reviewed in a single paper can be found in the Risk Adjustment Model Algorithm Software developed and released by Health and Human Services back in 2014.

CONCLUSION

I hope that you have found this interesting, and that you now go forth to clad all your programs in just the right amount of arrays.

This paper moves from simple to moderately complicated code rather quickly. If you find arrays potentially useful then you may want to review this paper, the associated conference presentation, and the references cited below. Most of them at least touch upon diagnosis codes as an example of implementation.

REFERENCES

Center for Consumer Information and Insurance Oversight. "2014 Benefit Year Risk Adjustment HHS-Developed Risk Adjustment Model Algorithm Software." *Regulations and Guidance*. Accessed March 7, 2016. Available at https://www.cms.gov/CCIIO/Resources/Regulations-and-Guidance/index.html.

Cody, Ron. 1999. "Transforming SAS Data Sets Using Arrays." SAS Institute Inc. 1999. *Proceedings of the Twenty-Fourth Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc.* Available at http://www2.sas.com/proceedings/sugi24/Advtutor/p48-24.pdf.

Pillay, Rahul G. 2015. "Doing More with SAS Arrays." Western Users of SAS Software. *WUSS 2015 Proceedings, San Diego, CA.* http://wuss.org/Proceedings15/85_Final_Paper_PDF.pdf.

Scerbo, Marge. 2004. "Array Tutorial (2) $ beginning intermediate." SAS Institute Inc. 2004. *Proceedings of the Twenty-Ninth Annual SAS® Users Group International Conference. Cary, NC: SAS Institute Inc.* Available at http://www2.sas.com/proceedings/sugi29/259-29.pdf.

Waller, Jennifer L. 2006. "Many to One Using a SAS DATA Step and PROC MEANS." *SESUG 2006: The Proceedings of the SouthEast SAS Users Group, Atlanta, GA, 2006*. Available at http://analytics.ncsu.edu/sesug/2006/ET05_06.PDF.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ryan Ferland
Blue Cross Blue Shield of Arizona
ryan.ferland@azblue.com