SESUG Paper 56-2021

Finding variable names and count of variables with missing values at various positions within an observation using SAS® arrays.

Kiran Venna, SMACT Works, Inc

Abstract

SAS arrays make it easy to find variable names or count of variables with missing values at various positions within an observation. In this paper, we will discuss four different scenarios of finding variable names/counts of missing values. The first scenario is finding all variable names which have missing values within an observation. The second one is finding the variable name of the first occurrence of missing value within an observation. The third one is finding the count of missing variables that are present in observation after at least one non-missing value. Finally, the last one is finding the total count of consecutive missing values only from the beginning and end of observation. All these scenarios will be discussed with relevant examples along with sample data and code.

Keywords

SAS Array, do loop, dim function, vname function.

Introduction

This introduction will contain separate sections, one for business aspect /case study and another section will be about technical aspects of arrays.For business aspect, we would like to discuss product sale for each month for each customer. Some of interesting business questions are mentioned below.

1. In which month's product sale have missing values for a particular customer?
2. What is first month of missing product sale for a particular customer?
3. What is the count of missing product sale for customer at the beginning and end of year in a particular year?
4. What is the count of missing values that are present in year after a least non-missing value?

All this information gives a better understanding of data and overall business. To answer all of these questions we need to look into many variables and process many variables. Whenever many variables need to be analyzed, the concept of arrays becomes important. Arrays help us to process many variables easily. Instead of covering the basics of Arrays, we would like to refer to a couple of SAS papers of which the first paper which is given in reference below acts as a primer for understanding the basics of Array[1]. The second paper is given in reference below about how arrays work in a more detailed fashion[2]. For simplicity, we will discussing only for numeric variables in an observation but same logic can applied to character variables.

Finding Variable names with missing Values

For this topic, the case study/example shown below has customer id and 6 months of data. The goal is to find which month's data has been missing.

```
data have;
       input id $ jansales febsales Marsales Aprsales Maysales Junsales;
       datalines;
1 . 100 . 200 . 300
2 90 110 . 109 . .
3 100 . 600 . 700 800
;
run;
```

Using SAS array and looping through each variable we can find variables that have missing values. Vname function helps to capture variable names and catx function helps further to concatenate all the missing variable names in an observation

```
data want;
       set have;
       length numeric_missing $50;
       array nu(*) _numeric_;
       do i=1 to dim(nu);
       if missing(nu{i}) then
       numeric_missing=catx(' ', numeric_missing, vname(nu{i}));
       end;
       drop i;
run;
```

PROC PRINT of the above created dataset shows the below result.

Obs	id	jansales	febsales	Marsales	Aprsales	Maysales	Junsales	numeric_missing
1	1	.	100	.	200	.	300	jansales Marsales Maysales
2	2	90	110	.	109	.	.	Marsales Maysales Junsales
3	3	100	.	600	.	700	800	febsales Aprsales

Finding Variable names with first missing value

For this topic, we will be using the same sample dataset shown above, Finding variable names with the first missing values can be done by understanding the code above. Instead of concatenating the variable names all we need to do is to capture the variable name for the first occurrence of missing value and then do not check any other variable and this can be done by using a leave statement in the do loop. Below is the code for the same.

```
data final;
	set have;
	array h(*) _numeric_;

	do i=1 to dim(h);

	if missing(h{i}) then
		do;
			colname=vname(h{i});
			leave;
		end;
	end;
	drop i;
run;
```

Proc print of the above created dataset shows the below result.

Obs	id	jansales	febsales	Marsales	Aprsales	Maysales	Junsales	colname
1	1	.	100	.	200	.	300	jansales
2	2	90	110	.	109	.	.	Marsales
3	3	100	.	600	.	700	800	febsales

Finding count of Missing values which appear only after at least one non-missing Value

The goal for this topic is a more complicated than what has been discussed previously and needs more logic. In this scenario, any missing value which comes before at least one non-missing value should not be counted. This can be understood more clearly by looking into the below DATA step, where this dataset is created. This dataset contains id variable and 12 numeric variables and variables are named from Jan to Dec. For ID 003, 004, and 005 there are 4 missing values at the beginning that is we have Jan, Feb, Mar, and Apr variables with missing values and they should be not counted.

```
data example;
	input @1 id:$3. Jan Feb Mar Apr May June Jul Aug Sep Oct Nov Dec;
	cards;
001 100 200 300 . . . . . 100 100 100 300
002 300 300 200 100 300 200 100 . . . . .
003 . . . . 100 200 300 100 200 300 200 .
004 . . . . 300 100 100 . 300 . 100 . .
005 . . . . 100 300 100 300 100 300 . .
006 300 200 100 . . . . . 100 100 100 300
;
run;
```

To achieve this goal arrays are very useful. Following logic is applied that is first we need to find the total number of missing values in an observation. This could be achieved by using CMISS function. The next one is to find at what position first non-missing values start. This can be done by using array and checking each variable for a non-missing value. Once we find the first non-missing value position and we need to

subtract by 1 it gives the number of missing values before a non-missing value. So finally to get the total number of missing values as per the requirement we need to subtract the total missing from the number of missing values before a non-missing value. Below is the code for the same.

```
data want;
	set example;
	total_miss=cmiss(of Jan--Dec);
	array myvar(*) Jan--Dec;

	do i=1 to dim(myvar);

		if not missing(myvar{i}) then
		leave;
	end;
/* we are using i as number of missing values before a non missing value starts*/
i=i-1;
total_miss=total_miss- i;
drop i;
run;
```

Proc print of the above dataset created shows results as follows.

Obs	id	Jan	Feb	Mar	Apr	May	June	Jul	Aug	Sep	Oct	Nov	Dec	total_miss
1	001	100	200	300	.	.	.	.	.	100	100	100	300	5
2	002	300	300	200	100	300	200	100	.	.	.	.	.	5
3	003	.	.	.	.	100	200	300	100	200	300	200	.	1
4	004	.	.	.	.	300	100	100	.	300	.	100	.	3
5	005	.	.	.	.	100	300	100	300	100	300	.	.	2
6	006	300	200	100	.	.	.	.	.	100	100	100	300	5

Total count of consecutive missing values only from the beginning and end of observation.

This one is more complicated than all of the previously mentioned scenarios. To solve this problem we will use the example dataset which we have previously discussed. The solution for this problem needs multiple steps. The first one is to count missing values from starting point in an array that is to start from the first element and then stop counting once a non-missing value is found. The second one is to count missing values from the end of observation and this can be achieved by looping from the last element to first and this can be easily achieved by using a do loop and looping from dim(array) to 1. One another important thing to make sure is if it has all missing values which we can find from count from looping from the front we do not further count from backward. As we are interested in consecutive values at the beginning and end of observation we are going to discard observation with one missing value. Finally, we are going to add a

count of consecutive missing values from the front and back to get the final number of missing variables as per requirement.

```
data want;
	set example;
	array nu(*) Jan--Dec;
	count_front=0;
	count_back=0;

	do i=1 to dim(nu);

		if nu{i}=. then
			count_front=count_front+1;
			else if nu{i} ne . then
		leave; `
	end;
	do j=dim(nu) to 1 by -1;

		/* if all are missing then do not check from back and leave the loop*/
		if count_front=dim(nu) then
			leave;
		if nu{j}=. then
			count_back=count_back+1;
		else if nu{j} ne . then
			leave;
	end;
	if count_front=1 then
		count_front=0;
	if count_back=1 then
		count_back=0;
	/* add count of consequtive missing values from beggining with end*/
	count=count_front+count_back;
	drop i j count_front count_back;
run;
```

Proc print of above code shows results as follows

Obs	id	Jan	Feb	Mar	Apr	May	June	Jul	Aug	Sep	Oct	Nov	Dec	count
1	001	100	200	300	.	.	.	.	.	100	100	100	300	0
2	002	300	300	200	100	300	200	100	.	.	.	.	.	5
3	003	.	.	.	.	100	200	300	100	200	300	200	.	4
4	004	.	.	.	.	300	100	100	.	300	.	100	.	4
5	005	.	.	.	.	100	300	100	300	100	300	.	.	6
6	006	300	200	100	.	.	.	.	.	100	100	100	300	0

CONCLUSIONS

Arrays are very helpful when you need to handle or process many variables. They are extremely useful when the dataset has many variables and we need to process them. The code written using an array makes it elegant and less error-prone. In this paper, we have illustrated how to find missing values in various scenarios in an observation. Without using SAS arrays some of these scenarios would have been pretty difficult to solve.

REFERENCES

1. https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/242-30.pdf
2. https://support.sas.com/resources/papers/proceedings11/244-2011.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the authors for more information:

Kiran Venna
SMACT Works Inc.
kvenna@smactworks.com