Paper 3367- 2015

Colon(:)izing My Programs

Jinson J. Erinjeri, The Emmes Corporation, Rockville, MD

ABSTRACT

There is a plethora of uses of the colon (:) in SAS programming. The colon is used as a data/variable name wild card, a macro variable creator, an operator modifier, and so forth. The usage of the colon helps in writing clear, concise and compact codes. The main objective of this paper is to encourage the effective use of the colon in writing crisp codes. This paper presents the real time scenarios of application of the colon in day to day programming. In addition, this paper also presents cases where the colon limits the programmer's wish.

INTRODUCTION

Colon (:) has been in Base SAS since version 5 and was specifically used to compare character data. The colon punctuation is not only used in SAS but other programming languages (such as C, Java, VB, etc.) for various purposes. Currently in SAS, there are multiple uses of colon and is a powerful tool in writing clear, concise and compact codes. This paper presents real time scenarios where colon can be applied to reap the above mentioned benefits. All the examples presented in the paper have been verified using Base SAS version 9.3. The various applications of colon in SAS are as follows:

1) Data Name Wild Card: A colon following a data set name prefix selects all the data sets starting with that prefix. This feature enables both combining (SET/MERGE) and deleting data sets starting with that prefix and thereby enabling compact coding. For example, if we want a single data set from a number of data sets, then we can apply the colon as shown in Table 1.

SAS Code	SAS Log

`data st;`
`  set st_:;`
`run;`
	`NOTE: There were 1 observations read from the data set WORK.ST_FL.`
	`NOTE: There were 2 observations read from the data set WORK.ST_MD.`
	`NOTE: There were 3 observations read from the data set WORK.ST_TX.`
	`NOTE: There were 3 observations read from the data set WORK.ST_VA.`
	`NOTE: There were 1 observations read from the data set WORK.ST_WA.`
	`NOTE: The data set WORK.ST has 10 observations and 16 variables.`

Table 1. Application of Colon as Data Name Wild Card in Combining Data Sets

The log in Table1 shows the data created with all data sets prefixing with st_. It is important to note that variables of interest cannot be selected from selective data sets using the DROP and KEEP statement while using colon. Also, the widely used IN statement will not be possible while combining data sets with colon.

The use of colon is very helpful when a large number of datasets created using macros needs to be deleted in order to clear up space in the directory. The usage of colon in deleting multiple data sets with PROC DATASETS is shown in Table 2.

SAS Code	SAS Log

`proc datasets lib=work;`
`  delete st_:;`
`run;`
	`NOTE: Deleting WORK.ST_FL (memtype=DATA).`
	`NOTE: Deleting WORK.ST_MD (memtype=DATA).`
	`NOTE: Deleting WORK.ST_TX (memtype=DATA).`
	`NOTE: Deleting WORK.ST_VA (memtype=DATA).`
	`NOTE: Deleting WORK.ST_WA (memtype=DATA).`

Table 2. Application of Colon as Data Name Wild Card in Deleting Data Sets

2) Variable Name Wild Card: Similar to the data name wild card, a colon following a variable name prefix selects all the variables starting with that prefix. The advantage of this feature is that it can be applied to statements, procedures and functions. The examples in Table 3 and 4 show various applications of colon as variable name wild card.
The variables in the KEEP statement will consider all variables following sbp and dbp respectively. The colon can be applied in a similar fashion to the commonly used LENGTH, DROP, FORMAT and INFORMAT statements.

SAS Code	SAS Log

`data keep_vars;`
`  set clinical;`
`  keep id sbp: dbp:;`
`run;`
	`NOTE: There were 20 observations read from the data set WORK.CLINICAL.`
	`NOTE: The data set WORK.KEEP_VARS has 20 observations and 7 variables.`

Table 3. Application of Colon as Variable Name Wild Card in Statements

The variable name wild card can be applied to functions (e.g. SUM, MEAN, MAX, MIN) as well as PROC's (e.g. PRINT, UNIVARIATE, FREQ) in the same way as shown in Table 4 for the variables starting with sbp.

SAS Code (Functions)

```
data functions_vars;
   set clinical;
   sum_sbp  =sum(of sbp:);
   mean_sbp =mean(of sbp:);
   max_sbp  =max(of sbp:);
   min_sbp  =min(of sbp:);
run;
```

SAS Output (of data set function_vars)

Obs	id	sum_sbp	mean_sbp	max_sbp	min_sbp
1	123	142	142.000	142	142
2	278	326	108.667	112	104
3	444	388	129.333	132	128
4	756	446	148.667	155	141

SAS Code (PROC's)

```
proc print data=clinical;
   where state="Virginia";
   var id sbp: dbp:;
run;
```

SAS Output

Obs	id	sbp1	sbp2	sbp3	dbp1	dbp2	dbp3
3	444	132	128	128	63	70	62
4	756	155	141	150	100	91	96
15	959	200	198	176	98	100	98

Table 4. Application of Colon as Variable Name Wild Card in Functions and PROC's

For all of the above cases, it should be cautioned that all variables prefixing the colon will be selected and this might include unnecessary variables (those prefixed right but not wanted in the analysis) which will give incorrect results. Also, the use of colon in PROC's such as UNIVARIATE, FREQ is best suited to listing output of each colon-prefixed variable. Creation of output data sets for all the colon-prefixed variables is a major limitation while using colon in the PROC's.

3) Creating Macro Variables: Colon is considered as a keyword component in creating macro variables in PROC SQL. The SELECT INTO *:macro-variable* is one of the most efficient way of creating macro variables in SAS. The example below depicts an efficient way of creating macro variables.

SAS Code (PROC's)	SAS Log

`proc sql;`
`  select quote(id)`
`  into :id_vit separated by ','`
`  from clinical`
`  where vitamins='1';`
`quit;`
`%put &id_vit;`
	`%put &id_vit;`
	`"123","444","193","","978","586","919","324","338","959"`
	`,"007"`

Table 5. Application of Colon in Creating Macro Variables

The log in Table 5 shows the resolved macro variable (id_vit) values separated by commas and each enclosed by quotes. It is important for a programmer to note that the length of the macro variable generated cannot exceed 32K characters.

4) Operator Modifier: The usage of colon in conjunction with character comparison operators (such as =, <,>, <=,>=, ne, lt, gt, in) modifies the nature of comparison from an "exact" match to "begins with" match. Table 6 lists some of the applications of the colon as an operator modifier.

SAS Code	SAS Output and Description

`proc print data=clinical;`
`  where id =: '0';`
`run;`
	Outputs all records where the variable id begins with character '0'.
	Obs id gender dob

	12 012 F 17814 13 012 F 17814 16 007 F 1570

`proc print data=clinical;`
`  where id in: ('0');`
`run;`
	Outputs all records where the variable id begins with character '0'.
	Obs id gender dob
	12 012 F 17814
	13 012 F 17814
	16 007 F 1570

`proc print data=clinical;`
`  where state >: ('Vd');`
`run;`
	Outputs all records where the variable state begins with character 'Ve' and up alphabetically.
	Obs id gender state
	3 444 F Virginia
	4 756 M Virginia
	5 811 F Washington
	13 012 F Wyoming
	15 959 F Virginia
	18 984 F Vermont
	19 669 M Vermont
	20 999 F Vermont

`proc print data=clinical;`
`  where '669'<=:id<=:'999';`
`run;`
	Outputs all records where the variable id is between characters '669' and '999'.
	Obs id gender dob
	4 756 M 1089
	5 811 F 5541
	8 978 M 12597
	10 919 F 12302
	15 959 F 2175
	18 984 F 7296
	19 669 M 6268
	20 999 F 12784

Table 6. Application of Colon as an Operator Modifier

While comparing character strings in conjunction with the colon modifier, SAS truncates the longer string to the length of the shorter string, thereby enabling the comparison of character strings prefix possible. It is important to remember this functionality while writing conditional statements because this can lead to unnoticeable errors.

5) Colon Modifier: Colons are used in the input statement for reading unorganized as well as unorganized data. For example, the statement "input name : $7." means that SAS will read the observation from column one up to the occurrence of the first delimiter with a length of 7. Space is the default delimiter but the DLM=/DELIMITER= option in the INPUT statement can be used to change this as needed. The example provided in Table 7 presents a simple application of colon modifier.

SAS Code	SAS Output (of data set x)

`data x;`
`infile datalines dlm=',';`
`input name : $7. gender : $1. zip : $5. age;`
`datalines;`
`AAAAA,M,12203,59`
`BBBBBBB,F,02138,24`
`;`
`run;`
	Obs name gender zip age
	1 AAAAA M 12203 59
	2 BBBBBBB F 02138 24

Table 7. Application of Colon as a Colon Modifier in Reading Data

The example in Table 8 is a classic example of using colon modifier in reading messy data. This kind of data is very common in log files such as weblogs where one is interested in filenames and dates accessed. The data is truly messy since variables of interest do not have the same length nor do they line up.

SAS Data Set and Code

Data of sam_web_log.txt
121.189.70.222 - - [05/Jul/2010:23:53:32 -0700] "GET /jin.htm HTTP/1.0" 343 152
128.33.444.5 - - [05/Jul/2010:23:53:39 -0700] "GET /hello.htm HTTP/1.0" 343 5555
128.33.0.5 - - [05/Jul/2010:23:54:15 -0700] "GET /dude.htm HTTP/1.0" 343 56555

```
data web_log;
  infile "C:\Jinson\SAS\SAS_Global_2015\sam_web_log.txt";
  input @'[' date_access date11.@'GET' file_name :$20.;
run;
proc print data=web_log;
  format date_access mmddyy8.;
run;
```

SAS Output

Obs	date_ access	File_name
1	07/05/10	/jin.htm
2	07/05/10	/hello.htm
3	07/05/10	/dude.htm

Table 8. Application of Colon as a Colon Modifier in Reading Messy Data

For reading messy data as shown in Table 8, the @ column pointer locates the position of the variables of interest and the colon modifier is used to account for the varying lengths of the variables (data_access and file_name). The format statement is used to display the data in date format rather than SAS dates.

6) Label Identifier: The colon is used as a label identifier to alter the flow of execution in SAS programs. This is mainly used in the EOF=label in an INFILE statement, a LINK statement and GOTO statement. The HEADER=label in a FILE statement and %LABEL in conjunction with %GOTO statement in SAS macros are also used to sequence the program flow. The example below shows a few applications of using colon as a label identifier.

The example in Table 9 shows the use of colon in LINK statement. The colon directs the execution to initialize when first patient (first.patid) is encountered and returns the execution to the statements below the link statements for the remaining observations for the same patient.

SAS Code

```
data check;
  set dats.doi(keep=patid distdt
               disttm disamp diroute
               didoamt1);
  by patid;
  if first.patid then link initialize;
    dosej+1;
    totdidoamt1+didoamt1;
   return;

   initialize:
   dosej=0;
   totdidoamt1=0;
   return;
run;
```

SAS Output (of data set check)

PATID	DIDOAMT1	dosej	totdidoamt1
CAC01022	0.5	1	0.5
CAC01023	0.25	1	0.25
CAC01023	0.5	2	0.75
CAC01023	0.75	3	1.5
CAC01023	0.25	4	1.75
CAC01023	0.75	5	2.5
CAC01024	0.3	1	0.3
CAC01024	0.5	2	0.8
CAC01024	0.3	3	1.1
CAC01024	0.5	4	1.6
CAC01024	0.3	5	1.9
CAC01024	0.5	6	2.4
CAC01024	0.3	7	2.7

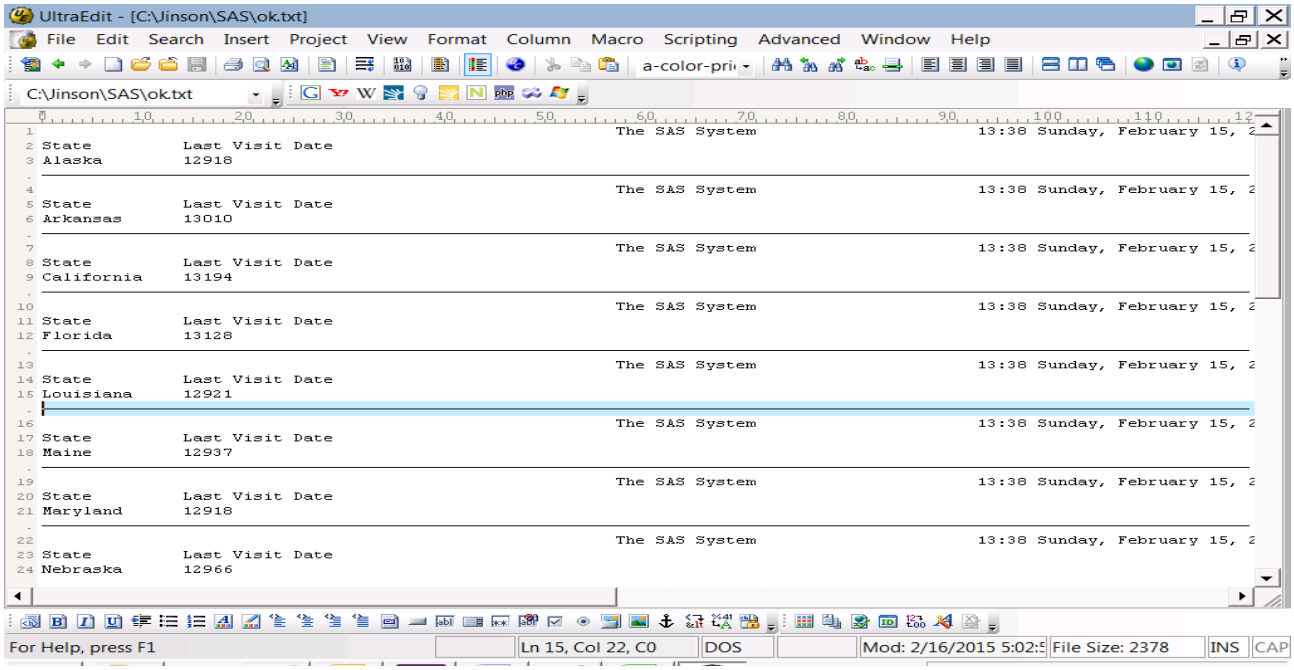
Table 9. Application of Colon as a Label Identifier in the DLINK Statement

On the other hand, the usage of GOTO statement will direct the execution to initialize when it is the first patient and for all other subsequent observations for the same patient, the statements below the GOTO statement will be executed.

The example below illustrates the usage of colon as a label identifier in HEADER=option in the FILE statements. In Display 1, every page has a common header which is due to header=newpage used in the FILE statement.

```
filename otpt "C:\Jinson\sas\ok.txt";
data _null_;
       set clinical;
       by state visit;
       file  otpt print header=newpage;
```

```
        if last.state and last.visit then do;
             output;
             put _page_;
             put @1 state @15 visit;
        end;
      return;
   newpage:
             put @1 'State' @15 'Last Visit Date';
      return;
run;
```



Display 1. Screen Shot of Application of Colon as a Label Identifier in the Header Option of the File Statement

%LABEL is used in macros especially with %GOTO to provide a path to branch out. This is particularly used in macros to provide exit points when error occurs. For example, in the macro %check_day, we use the %exit (%LABEL) to terminate the macro if it is not the fifteenth day of the month.

```
%macro check_day;
%If %substr(&sysdate,1,2)=15 %then /*fifteenth day of the month*/
       %put  &sysdate.;
%else %goto exit;
%exit: %mend check_day;
%check_day;
```

7) Array Variable Name Wild Card and Array Bound Delimiter: Colon is used as variable name wild card as described before and the same can be extended to arrays. For example, if an array has to be defined with many variables, say years from 1901 to 2015 (year1901 to year2015). This can be declared as

```
array year[*]  year:;
       do i= 1 to dim[year];
             statements……
       end;
```

The advantage of using the above approach is that the colon list all variables starting with year and the argument '*' counts the elements in array with the DIM function retrieving the same while iterating the DO loop. This becomes advantageous when there unknown number of variables starting with prefix year. The downside is that the variables starting with prefix year should be the one needed for analysis and as stated before, the knowledge of the data prior to

any data manipulation is helpful. In an array statement, the subscript in each dimension ranges from 1 to n, where n represents the number of elements in that dimension. For example, in the statement

```
array example[2,3] example1-example6
```

the bounds of the first dimension are 1(lower bound) and 2(upper bound) and those of the second dimension are 1(lower bound) and 3(upper bound). There are scenarios where bounds are to be explicitly specified and in such cases one can define the array in the bounded fashion. For the example above, we redefine the statement as

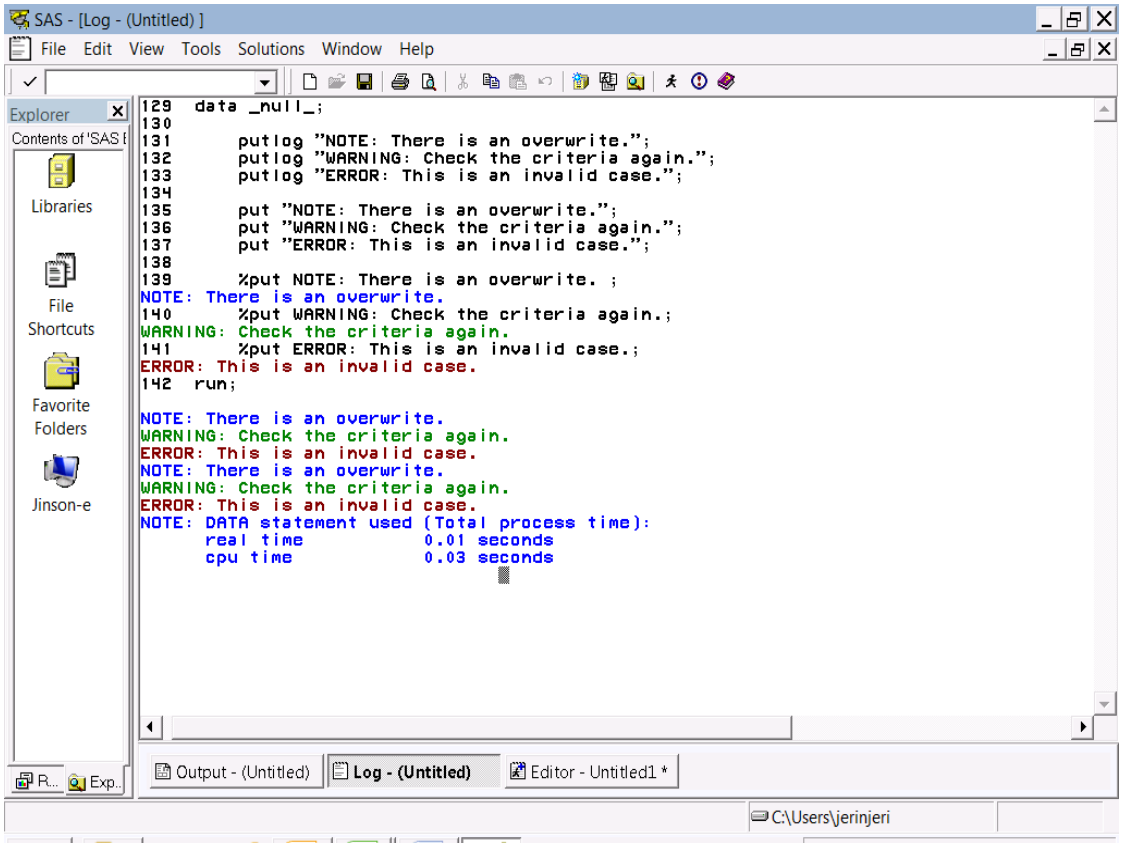
```
array example[1:2,1:3] example1-example6
```

Colon is used as the delimiter to specify the bounds of the dimension in an array. The specification of bounds in arrays brings clarity to the code when dealing with large number of similar variables such as the time-related ones. For example,

```
array dob{1985:2015} dob1985-dob2015;
do i=lbound(dob) to hbound(dob);
  if dob{i}=99 then dob{i}=.;
end;
```

Using the lbound and hbound function and the iterative process of DO loop, changes can be made specific to the variables of interest. In the above example, we set the bound between 1985 and 2015 and assign the values of variable dob(date of birth) equal to missing when the value is 99.

8) Log-Message Indicator: The colon is used in conjunction with keywords ERROR, WARNING and NOTE in PUTLOG, PUT and %PUT statements to generate customized messages in SAS logs as shown in Display 2.



Display 2: Screen Shot of Application of Colon as Log-Message Indicator

This indicator is very useful in creating error-traps and for troubleshooting purposes both in development as well as production environment. It is important to note the default colors of messages with NOTE in blue, WARNING in green and ERROR in red. It is worth noting that this feature is limited to SAS editor only.

CONCLUSION

Colon is helpful in writing crisp codes as shown in this paper with its usefulness as data/variable name wild card, operator modifier, label identifier, macro variable creator and so forth. Therefore, it is recommended to employ colons wherever feasible to write crisp codes.

REFERENCES

William C. Murphy. 2004. Colonoscopy for the SAS Programmer.
Available at: http://www2.sas.com/proceedings/sugi29/054-29.pdf

Sudhir Singh. 2012. Using SAS Colon Effectively.
Available at: http://www.pharmasug.org/proceedings/2012/TA/PharmaSUG-2012-TA05.pdf

http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a000206879.htm.
Accessed March 7, 2015.

http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000739626.htm.
Accessed March 7, 2015.

ACKNOWLEDGMENTS

The author would like to thank Andrew Lewandowski for his suggestions while reviewing this paper.

CONTACT INFORMATION

Your comments/questions/criticisms are valued and encouraged. Please contact the author at:

Jinson J. Erinjeri
The Emmes Corporation
401 N Washington St.
Rockville, MD 20850
Work Phone: 301-251-1161(x 2917)
E-mail: jerinjeri@emmes.com
Web: www.emmes.com