

The Variable List Wildcard Character

Malachy J. Foley

University of North Carolina at Chapel Hill, NC

ABSTRACT

A very simple and powerful, yet relatively unknown, programming technique is the use of a wildcard character in variable lists. The wildcard is the colon. This paper shows how to use the technique, when it works, and when it does not work. Finally, this article presents code to make the wildcard work in many cases.

INTRODUCTION

The lowly : (colon) character can greatly simplify a SAS® programmer's life. When used in a variable list, it reduces keying errors, keying time and keying drudgery. Anyone who uses the SAS BASE product can benefit from this article.

The best way to introduce the colon wildcard character is through examples. To this end, the next section shows some test data. The subsequent sections give examples of the wildcard's use. Throughout the paper there is a discussion of various aspects of the colon wildcard. The conclusion of this paper summarizes the features of the colon wildcard.

A TEST DATA SET

The following is part of the PROC CONTENTS of a test data set. This data will be used to illustrate the wildcard character. Note that the file is called TEST1, has 24 variables, and its contents are in position order.

Exhibit 1: Contents of TEST1 file.

#	Variable	Type	Len	Format
1	ID_NUM	Char	4	\$4.
2	ID_SEQ	Char	2	
3	IVAA1	Char	1	
4	XIVAA1	Char	3	
5	IVAA2A	Char	1	
6	XIVAA2A	Char	3	
7	IVAA2B	Num	8	
8	XIVAA2B	Char	3	
9	IVAA3	Char	1	
10	XIVAA3	Char	3	
11	IVAB1	Num	4	
12	XIVAB1	Char	3	

```
13  IVAC1      Num      8
14  XIVAC1     Char      3
15  IVAC2A1    Num      8
16  XIVAC2A1   Char      3
17  IVAC2A2    Num      8
18  XIVAC2A2   Char      3
19  IVAC2B1    Num      8
20  XIVAC2B1   Char      3
21  IVAC2B2    Num      8
22  XIVAC2B2   Char      3
23  IVAC3A     Num      8
24  XIVAC3A    Char      3
```

A SIMPLE EXAMPLE – Dropping Variables

The above file has a certain semblance of order. The first two variables (ID_NUM and ID_SEQ) identify each record. After the identifying variables, there is a series of data variables in the file. The data variable names all start with either IVA or XIVA. The IVA variables are actual data values needed for processing. The XIVA variables contain information on how the corresponding IVA variable was keyed.

To illustrate the use of the : (colon) wildcard character, all XIVA variables will be dropped from the TEST1 file given in Exhibit 1. Notice how the XIVA are interspersed with the IVA variables. Normally, you would need to list all the XIVA variables in a drop statement to eliminate these variables. For example, one would have to type the following:

```
DROP  XIVAA1  XIVAA2A  XIVAA2B  . . .
XIVAC3A2;
```

There are more than 100 keystrokes in the above DROP statement. To type out the entire statement is a cumbersome, time-consuming and error-prone process. It is fairly easy to mis-key one of the 100 characters in the statement. However, there is a simpler and less error-prone way to drop all the variables beginning with XIVA, namely:

```
DROP XIVA: ;
```

Or even more simply:

```
DROP X: ;
```

The "X:" in the previous statement can be read as "all variables starting with the character X". The next exhibit shows how this DROP statement works.

```
Exhibit 2: Drop all vars starting with X.
-----
DATA TEST2;
  SET TEST1;
  DROP X: ;
```

```

RUN;
-----
-----
Contents of the output file TEST2
-----
#  Variable  Type  Len  Format
-----
1  ID_NUM    Char   4   $4.
2  ID_SEQ    Char   2
3  IVAA1     Char   1
4  IVAA2A    Char   1
5  IVAA2B    Num    8
6  IVAA3     Char   1
7  IVAB1     Num    4
8  IVAC1     Num    8
9  IVAC2A1   Num    8
10 IVAC2A2   Num    8
11 IVAC2B1   Num    8
12 IVAC2B2   Num    8
13 IVAC3A    Num    8
-----

```

EXAMPLE with KEEP, LENGTH, & FORMAT

Exhibit 2 demonstrates how the colon acts as a wildcard character in naming a list of variables in a DROP statement. This wildcard character can be used just about everywhere a variable list is required. In the next example, the colon is successfully used in the KEEP, LENGTH and FORMAT statements.

Exhibit 3: Using the : in 3 SAS Statements.

```

-----
DATA TEST3;
  SET TEST2;
  KEEP ID: IVAC: ;
  LENGTH IVAC2: 4;
  FORMAT IVAC2: 6.;
RUN;
-----
-----
Contents of the output file TEST3
-----
#  Variable  Type  Len  Format
-----
1  ID_NUM    Char   4   $4.
2  ID_SEQ    Char   2
3  IVAC1     Num    8
4  IVAC2A1   Num    4   6.
5  IVAC2A2   Num    4   6.
6  IVAC2B1   Num    4   6.
7  IVAC2B2   Num    4   6.
8  IVAC3A    Num    8
-----

```

WHERE THE COLON WORKS

The previous examples show how the : (colon) wildcard can work with the DROP, KEEP, FORMAT, and LENGTH statements. The colon works with the DROP and KEEP SAS Data Set Options as well.

In fact, it works just about anywhere a *variable-list* is called for in the SAS syntax. As such, the colon functions in the ARRAY, BY, and PUT statements. It also functions in SAS PROC's. For instance, it works in the VAR statement of PROC PRINT:

Exhibit 4: Using the : in a SAS PROC

```

-----
PROC PRINT DATA=TEST1;
  VAR IVAA:;
RUN;
-----
-----
Output from the PROC PRINT (LST file)
-----
OBS    IVAA1    IVAA2A    IVAA2B    IVAA3
-----
1      N      Y      1      Y
2      N      Y      1      Y
3      N      N      .      Y
4      N      N      .      N
-----

```

HOW THE WILDCARD WORKS

Notice how the wildcard worked in the prior exhibit. When the colon is used in statements like KEEP and LENGTH the order of the variables is usually not important. However, when using the : in statements like the VAR or BY, order can be important or even critical.

So how does SAS construct a variable list when a colon is used? Exhibit 4 reveals the process involved. When SAS sees a colon at the end of a variable name, it looks through the PROC CONTENTS in position order and selects any variable it finds that has a name that starts with the characters that precede the colon. If a variable name is exactly equal to the characters that precede the colon, that variable is included in the list as well.

Thus the variable list created by the colon wildcard is in position order from the descriptor part of the file as given in the PROC CONTENTS. Strictly speaking, the variables and position order is taken from the Program Data Vector (PDV) rather than the CONTENTS. However, the two almost always provide exactly the same results. The two might differ in a SAS Step that creates a new variable that matches the wildcard specification. For more information on the Program Data Vector see pages 16-19 of the *SAS Language Reference* and Whitlock's paper (given in REFERENCES).

EXAMPLE with PUT

Below is an example of how the wildcard is used with the PUT statement. This case is mentioned because it may not be obvious. Page 451 of the *SAS Language Reference* outlines the relevant syntax.

```
PUT ...(variable-list) (format-list)...;
```

In the PUT statement a variable list must be enclosed in parenthesis. Furthermore, a parenthesized variable list must be followed by a parenthesized format list. Thus,

Exhibit 5: Using the : in a PUT Statement

```
DATA _NULL_;
```

```

SET TEST1;
PUT "*** PUT = " (IVAA:) (5.);
RUN;
-----
Output from the DATA Step (LOG file)
-----
*** PUT = N      Y      1      Y
*** PUT = N      Y      1      Y
*** PUT = N      N      .      Y
*** PUT = N      N      .      N
-----

```

Notice that the IVAA: in the PUT Statement really means IVAA1 IVAA2A IVAA2B IVAA3. Just like in the PROC PRINT example of Exhibit 4, the variables are in position order.

Also notice that the format-list did not correspond to the variables! Nonetheless, the PUT Statement operated properly. In this case, SAS was smart enough to find a format that worked with each of the variables in the list generated by the wildcard.

WHERE THE COLON DOES NOT WORK

The colon wildcard does not work in all the places you might expect it should. For instance, it does not work in SAS Functions like NMISS and MEAN. Namely, when SAS encounters the following statement it gives an error message that the colon “symbol is not recognized”.

```
NUM_MIS=NMISS(OF IVAC2: );
```

However, in the case of the NMISS and other SAS functions, there is a simple way to change the code so that the functions will work. The next section presents this change.

THE WORKAROUND

As explained in the previous section, variable lists are not always allowed in SAS code where you would like to use the colon. One way around this predicament is to put the wildcard variable list in an ARRAY and specify the ARRAY in the code. This technique is presented in the following Exhibit.

Exhibit 6: The Workaround for a SAS Function

```

-----
DATA _NULL_;
SET TEST2;
ARRAY LST(*) IVAC2;;
NUM_MIS=NMISS(OF LST(*));
PUT "*** PUT = " (LST(*) )(5.) NUM_MIS=;
RUN;
-----
-----

```

```

Output from the DATA Step (LOG file)
-----
*** PUT = 164      .      160      80      NUM_MIS=1
*** PUT = 190      .      180      .      NUM_MIS=2
*** PUT =      .      80      .      NUM_MIS=3
*** PUT =      .      80      180      .      NUM_MIS=2
*** PUT = 144      72      146      70      NUM_MIS=0
-----

```

Observe that in the above illustration all the variables in the ARRAY and in the wildcard list are of the same type, namely numeric. SAS requires that all the variables listed in an ARRAY statement be of the same type (either all character or all numeric). Similarly, many functions have the same requirement. Therefore, the workaround is valid for most functions and can be used in any situation where an ARRAY can be used.

CONCLUSION

A wildcard character is a special symbol used to represent, or replace, one or more characters. Most operating systems and many applications use wildcard characters. SAS is no exception. When it comes to list of SAS variables, the : (colon) can be used as a suffix wildcard.

When a set of characters is followed by a colon in a variable list (ex: “IVAA:”), it implies that several variables are to be specified in the list. SAS creates the implied variables by scanning the Program Data Vector and selecting all the variables that start with the characters preceding the colon. For example “IVAA:” would mean “IVAA1 IVAA2A IVAA2B IVAA3” for the data set given in Exhibit 2. This process is explained in more detail in the section entitled “How the Wildcard Works”.

The wildcard can select variables with mixed types (character and numeric). The IVAA: is a case in point. IVAA2A is Character and IVAA2B is numeric.

The colon wildcard may be employed almost anywhere a variable list is allowed in SAS. Exhibits 2-6 demonstrate the use of the wildcard in the DROP, KEEP, LENGTH, FORMAT, VAR, PUT, BY and ARRAY statements. The wildcard can be used in the KEEP and DROP data set options as well.

In some cases SAS does not allow the use of the wildcard in a variable list. For instance, a colon can not be used in a SAS function like NMISS and MEAN. In these cases the ARRAY workaround can be used. Exhibit 6 gives an example of this workaround. Note that the ARRAY’s can not accept mixed variable types.

Using the wildcard in variable lists is easier, faster and more accurate than typing the entire variable list. As such, the colon wildcard is the SAS programmer’s friend.

REFERENCES

SAS Institute, Inc. (1990), SAS Language: Reference, Version 6, First Edition, Cary, NC: SAS Institute Inc.

Whitlock, Marianne. (1998) "The Program Data Vector As an Aid to DATA step Reasoning" Proceedings of the Sixth Annual Conference of the SouthEast SAS Users Group, 229-238.

TRADEMARKS

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

AUTHOR CONTACT

The author welcomes comments, questions, corrections and suggestions.

Malachy J. Foley
2502 Foxwood Dr.
Chapel Hill, NC 27514

Email: FOLEY@unc.edu