

YOUR FRIEND THE COLON: SOME EFFICIENCY TECHNIQUES FOR COMPARING OR SELECTING CHARACTER DATA

S. David Riba, JADE Tech, Inc., Clearwater, FL

ABSTRACT

There are many techniques that can be used when comparing or selecting character data. One of the most useful is also probably one of the least known. When used to modify comparison operators, the colon (:) can greatly improve the flexibility of the comparison operator while considerably simplifying the resulting SAS program code. This modifier has been part of Base SAS® since the days of Version 5 or earlier. This paper will review some typical uses where the colon modifier can improve on traditional programming methods when comparing or selecting character data.

INTRODUCTION

Have you ever seen this comparison operator?

=:

Probably not, although this operator has been around in the SAS® System for a long time. In fact, I found it back in the Version 5 documentation. In Version 6 it is documented on page 125 of the *SAS Language: Reference manual*. It is intended for character comparisons, and modifies the existing comparison operators.

The colon after the equal sign changes the operator from an equality operator (test if A equals B) to an operator that compares all values that start with a given value.

For example, contrast the following

```
if charvar = 'A' ;
if charvar =: 'A' ;
```

The first example will find a match for every observation where the variable CHARVAR is equal to the character A. The second example will find a match for every observation where the variable CHARVAR begins with the character A. With this simple change to the equality operator, the IF test will now compare a much broader range of values without the need to code and evaluate the results of multiple SUBSTR functions.

THE COLON MODIFIER

The colon can be used to modify any character comparison operator. It changes the nature of the comparison from an exact match to a "begins with" match. When used to modify an EQUALS operator (=:), the comparison is between either two text strings or a data set variable and a text string. For example,

```
if charvar =: 'A' ;
```

The colon modifier has no effect when used between two data set variables, like

```
if charvar1 =: charvar2 ;
```

In this case, only an exact match between CHARVAR1 and CHARVAR2 will evaluate correctly.

Depending on your data, you can now do comparisons like

```
if UPCASE(charvar) =: 'SM' ;
```

to find all records that begin with the letters SM (Smith, Smythe, Smithfield, Smedley, and so on). Note that since this is a character comparison, the case of the text is significant. Using the UPCASE function normalizes the data for a better match.

There is another advantage to this particular construction over using the SUBSTR function to locate matches on the beginning of character strings. With the SUBSTR function, you must code additional parameters, including the beginning range (a 1 to start at the beginning) and (optionally) an ending range of the string to extract. With the =: operator, all that is necessary is to enter as much of the string as is necessary to define the unique match. For example,

```
if UPCASE(charvar) =: 'SMIT' ;
```

will find all records that begin with the letters SMIT (Smith, Smithfield, and so on), regardless of the case of the character data.

MORE IDEAS FOR THE COLON MODIFIER

When used to compare two text strings, the longest string will be truncated to the length of the shortest one for purposes of evaluation. Both of the following comparisons will evaluate as true:

```
if 'Abc' =: 'Ab' ;  
if 'Abc' =: 'Abcd' ;
```

All of the examples to this point have looked at using the colon to modify the equals operator (=). In fact, the colon can modify the other character operators as well.

Consider the following examples:

```
if charvar >: 'A' ;
```

will return all records where the variable CHARVAR begins with the upper case letter 'A'.

```
if charvar in: ('A','B','C');
```

will return all records where CHARVAR starts with either the upper case letters 'A', 'B', or 'C'.

```
if charvar in: ('Sm','Wilk','Z');
```

Notice that the length of the character strings no longer matters? The SUBSTR function will fail if the ending range is greater than the length of the string. For example,

```
if SUBSTR(charvar,1,4) eq 'Wilk';
```

will fail if the length of CHARVAR is less than four characters. However, with the colon modifier on the IN operator, the previous example works correctly. Imagine how much extra coding would have been required without the colon modifier!

CONCLUSION

One of the lesser known features of Base SAS for many years, the colon can be used to extend the functionality of the character comparison operators. Used in conjunction with any of the comparison operators, the colon modifies each operator from an exact match on specific characters to a match on only the beginning characters. It is very useful to simplify many SAS coding challenges.

REFERENCES

SAS Institute, Inc., *SAS® Language: Reference, Version 6, First Edition*, Cary, NC. SAS Institute Inc., 1990. pp 125

Riba, S. David (1997). Efficient Comparison of Nonprecise Character Data. *Observations*. Vol 6, No 2. pp. 17-20.

TRADEMARK INFORMATION

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

ACKNOWLEDGMENTS

I would like to thank Jørn Ibsen of SAS Denmark and Ian Whitlock for their helpful comments and ideas.

AUTHOR

The author welcomes comments, suggestions, and questions by phone (813-726-6099) or by e-mail JADETECH@MCIMAIL.COM.

The author may be contacted at:

S. David Riba
JADE Tech, Inc.
P O Box 4517
Clearwater, FL 33758
(813) 726-6099
INTERNET: jadetech@mcimail.com

