

Paper 88-25

The SAS SUBSTR Function – A Beginner's Tutorial

Paul D. McDonald, SPIKEware, Inc., Schaumburg, IL

ABSTRACT

This paper is written for SAS Users and SAS Programmers of any experience level. It reviews the syntax, basic applications, and troubleshooting of the SUBSTR Data Step Function.

INTRODUCTION

The SAS data step function SUBSTR (commonly pronounced “substring”) function is used to work with a specific position or positions of characters within a defined character variable. The function focuses on a portion of a string and can go on either side of the “=” sign in a data step statement.

SYNTAX

The SUBSTR function has three arguments:

```
SUBSTR(SOURCE, POSITION, N)
```

The function returns *N* characters, beginning at character number *POSITION* from the string *SOURCE*.

- ◆ **SOURCE**—This is the larger or reference string. It can be a variable or a string of characters.
- ◆ **POSITION**—This value is a positive integer and references the starting point to begin reading the internal group of characters.
- ◆ **N**—This value is a positive integer and references the number of characters to read from the starting point *POSITION* in the field *SOURCE*.

APPLICATIONS

There are two basic applications of the SUBSTR function: on the “right” side of an assignment statement or on the left side of an assignment statement.

RIGHT SIDE APPLICATION

In this example, we are going to assign the area code of a telephone number to a variable called AREA_CD.

```
%let phone = (312) 555-1212 ;

data _null_ ;
  phone = '(312) 555-1212' ;

  area_cd = substr(phone, 2, 3) ;
  area_cd = substr('(312) 555-1212', 2, 3) ;
  area_cd = substr("&phone", 2, 3) ;
run ;
```

Each time the function is applied, the *SOURCE* is a different type, but the end result is the same.

1. The first use pulls from the variable PHONE
 2. The second pulls from hard-coded string
 3. The third pulls from the quoted macro variable &PHONE.
- Result: '312' assigned to the variable AREA_CD each time.

LEFT SIDE APPLICATION

Suppose you want to change just a few characters of a variable—use the SUBSTR function on the left side of the assignment statement.

```
data _null_ ;
  phone = '(312) 555-1212' ;
  substr(phone, 2, 3) = '773' ;
```

```
run ;
```

In this example, the area code of the variable PHONE was changed from '312' to '773'.

TROUBLESHOOTING**VALUE TOO LARGE FOR POSITION**

The second argument *POSITION* cannot point past the end of the *SOURCE*.

```
1 data _null_ ;
2 var1 = '12345' ;
3 var2 = substr(var1, 7, 2) ;
4 run ;
```

NOTE: Invalid second argument to function SUBSTR at line 3 column 12.

```
VAR1=12345 VAR2=12 _ERROR_=1 _N_=1
```

In the above example, VAR1 has a length of \$5 and the SUBSTR function is supposed to read 2 characters, beginning in position 7. Since that 2-character read from position 7 will attempt to start the pointer past the length of *SOURCE* VAR1, the SUBSTR function cannot work properly.

The result of this type of incorrect usage is a **NOTE:** in the SAS log.

VALUE TOO LARGE FOR N

The third argument *N* cannot read past the end of the *SOURCE*.

```
1 data _null_ ;
2 var1 = '12345' ;
3 var2 = substr(var1, 2, 7) ;
4 run ;
```

NOTE: Invalid third argument to function SUBSTR at line 3 column 11.

```
VAR1=12345 VAR2=2345 _ERROR_=1 _N_=1
```

In the above example, VAR1 has a length of \$5, and the SUBSTR function is supposed to read 7 characters, beginning in position 2. Since that 7-character read from position 2 will put the pointer past the length of *SOURCE* VAR1, the SUBSTR function cannot work properly.

The result of this type of incorrect usage is a **NOTE:** in the SAS log.

VALUE NOT POSITIVE INTEGER

The second and third arguments *POSITION* and *N* must be positive integers.

```
1 data _null_ ;
2 var1 = '12345' ;
3 var2 = substr(var1, -2, 1) ;
4 run ;
```

NOTE: Invalid second argument to function SUBSTR at line 3 column 12.

```
VAR1=12345 VAR2=1 _ERROR_=1 _N_=1
```

In the above example, argument VAR1 has a length of \$5 and the SUBSTR function is supposed to read 1 characters beginning in position -2. Since it is impossible to begin in position -2 before the

start of the string (just as it is impossible to begin after the end of the string), the `SUBSTR` function cannot work properly.

This rule holds true when trying to read a negative string in the opposite direction: a negative number may not be placed in the third argument *N*.

The result of this type of incorrect usage is a `NOTE:` in the SAS log.

TROUBLESHOOTING SUMMARY

- ◆ Any deviation from valid arguments results in a `NOTE` and not a `WARNING` or `ERROR` message in the SAS Log
- ◆ Any occurrence of an invalid argument results in all variables from that observation being written to the log
- ◆ Be sure to check your log each time you use the `SUBSTR` function

CONTACT INFORMATION

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

Paul D. McDonald
SPIKEware, Inc.
1450 East American Lane Suite 1400
Schaumburg, IL 60173
Work Phone: (847) 330-4460
Fax: (847) 798-0732
Email: PDM@SPIKEware.com
Web: <http://www.SPIKEware.com/>