ABSTRACT

Being flexible and highlighting important details in your output is critical. The use of ODS ESCAPECHAR allows the SAS® programmer to insert inline formatting functions into variable values through the DATA step, and it makes for a quick and easy way to highlight specific data values or modify the style of the table cells in your output. What is an easier and more efficient way to concatenate those inline formatting functions to the variable values? This paper shows how the CAT functions can simplify this task.

INTRODUCTION

In your Output Delivery System (ODS) output, you can define an escape character (for display purposes, we will use the caret ^ in this paper) with the ODS ESCAPECHAR = statement. The ODS escape character allows you to use inline formatting functions. These functions enable you to set the style for text strings like ODS text fields, titles and footnotes. By using string concatenation functions, the inline formatting functions also can be inserted into individual variable values in the DATA step and thus provide the ability to modify and enhance the text style within data cells.

Below is a simple SAS data set (Table 1) where each Var1 value was concatenated with an inline function to create the Var2 value:

<table>
<thead>
<tr>
<th>Var1</th>
<th>Var2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Dagger</td>
<td>A Dagger^{dagger}</td>
</tr>
<tr>
<td>An Alert</td>
<td>^{style [foreground=red]An Alert}</td>
</tr>
<tr>
<td>A Superscript</td>
<td>A Superscript^{super 1}</td>
</tr>
</tbody>
</table>

Table 1. A SAS Data Set with Concatenating Inline Formatting to the Variable Values

If a PROC PRINT were used on the data set above, the result in the ODS output (RTF destination) would be the following:

<table>
<thead>
<tr>
<th>Obs</th>
<th>Var1</th>
<th>Var2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A Dagger</td>
<td>A Dagger^{dagger}</td>
</tr>
<tr>
<td>2</td>
<td>An Alert</td>
<td>An Alert</td>
</tr>
<tr>
<td>3</td>
<td>A Superscript</td>
<td>A Superscript^{super 1}</td>
</tr>
</tbody>
</table>

Output 1. RTF Output from Table 1

There are two general ways to concatenate strings together. One way is to use the CAT family of functions (CAT, CATS, CATT, and CATX functions) which were introduced in SAS® Version 9. The other way to concatenate strings together is to use the concatenation (double vertical bar) operator, ||, with functions such as TRIM and/or LEFT, but this method can be less compact than using one of the functions in the CAT family of functions. For example,

Without CAT Function: var2=trim(left(var1))||"^{dagger}" ;

With CAT Function: var2=cats(var1,"^{dagger}");

In this paper, we will first briefly review some of the inline functions that can be used with ODS ESCAPECHAR, then explore the features of CAT family functions, and finally we will demonstrate how the CAT functions are an efficient and easy way to catch the inline formatting functions for creating the style you desire.
ODS INLINE FORMATTING FUNCTION: GIVE YOUR OUTPUT A STYLE

Inline formatting functions enable you to modify styles in titles and footnotes, text strings, and table cells; however, before you can perform inline formatting with ODS, you must first use the ODS ESCAPECHAR statement to define an escape character, such as:

```
ODS ESCAPECHAR='^';
```

The general format for the inline style syntax is:

```
^{function-name<<arg-1 <arg-2<arg-n>>>}
```

Where "^^" is the ODS escape character, "{" and "}" are the inline style grouping characters, arg-1 ... arg-n are the arguments to the given function, and function-name is the name of the inline function.

Below is an example that uses the STYLE function in a title:

```
Title 'The Title is ^{style [color=blue]blue}';
```

This example would produce the following result:

*The Title is blue*

ODS ESCAPECHAR is designed to be used in output designated for the Output Delivery System. The inline formatting functions will affect all open destinations except for the LISTING destination. But some functions can only be used with certain destinations, like the PAGEOF function, ^{PAGEOF} , which only works with the RTF and TAGSETS.RTF destinations for showing the Page X of Y.

Table 2 lists the functions that work for most destinations. (All are available in SAS® V9.2)

<table>
<thead>
<tr>
<th>Function Syntax</th>
<th>Description</th>
<th>Example</th>
<th>RTF Output (Generated from: ODSINLINE.SAS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>^{dagger}</td>
<td>Generates the Greek dagger sign †</td>
<td>MyValue^{dagger}</td>
<td>MyValue†</td>
</tr>
<tr>
<td>^{nbspace &lt;count&gt;}</td>
<td>Count is the number of spaces that you want to insert; default value is 1</td>
<td>MyValue1^{nbspace 2}MyValue2</td>
<td>MyValue1 MyValue2</td>
</tr>
<tr>
<td>^{newline &lt;count&gt;}</td>
<td>Count is the number of lines that you want to insert; default value is 1</td>
<td>MyValue1^{newline 1}MyValue2</td>
<td>MyValue1 MyValue2</td>
</tr>
<tr>
<td>^{sigma}</td>
<td>Produces the Greek Sigma sign σ</td>
<td>MyValue^{sigma}</td>
<td>MyValueσ</td>
</tr>
<tr>
<td>^{style</td>
<td>&lt;style-element&gt;&lt;[attributes]&gt;} formatted text}</td>
<td>The style attributes or elements remain in effect until they are overridden by another style</td>
<td>^{style [foreground=red]} MyValue]</td>
</tr>
<tr>
<td>^{sub subscript-text}</td>
<td>The subscript-text can be a numeric, alphanumeric, or a character value. This value is written below and immediately to one side of another character</td>
<td>MyValue^{sub 1}}</td>
<td>MyValue1</td>
</tr>
<tr>
<td>^{super superscript-text}</td>
<td>The superscript-text can be a numeric, alphanumeric, or a character value. This value is written above and immediately to one side of another character</td>
<td>MyValue^{super *}</td>
<td>MyValue*</td>
</tr>
<tr>
<td>^{Unicode &lt;Hex [name]&gt;}</td>
<td>Prints any special character by name or an actual four-place hexadecimal Unicode value.</td>
<td>MyValue1^{Unicode 2265}MyValue2</td>
<td>MyValue1≥MyValue2</td>
</tr>
</tbody>
</table>

Table 2. Some Inline Functions that can be used with ODS ESCAPECHAR
Program ODSINLINE.sas creates several observations in which different in-line formatting functions have been added for controlling their styles in the data cells. By using string concatenation functions, the in-line formatting can also be inserted into an existing data value in the DATA step; next we will explore the features of the CAT family of functions family to find out why it is the best choice to simplify the concatenation task.

THE CAT FUNCTIONS: WHAT IS THE CAT DOING?

In SAS you can combine strings of characters together with string concatenation functions. In Version 9, SAS introduced the CAT family of functions (CAT, CATS, CATT, and CATX functions). These string concatenation functions can take the place of two older functions, TRIM and LEFT, when they are paired with the concatenation operator (||).

As shown in the Introduction, the CAT family of functions gives you an easier and more compact way to streamline your SAS code when concatenating character strings. In addition, you can use the CAT functions with the OF operator to refer to an entire group of sequentially numbered variables without listing each one, for example,

Without OF Syntax: \( X_1 || X_2 || X_3 || X_4 \) or CAT(X1, X2, X3, X4)

With OF Syntax: \( CAT(\text{of } X_1-X_4) \)

The lengths of the resulting variables created by the two concatenation methods may be different. In a DATA step, if the CAT function returns a value to a variable that has not previously been assigned a length, then the length of that variable defaults to 200 bytes. In contrast, if the concatenation operator (||) returns a value to a variable that has not previously been assigned a length, then the length of that variable is the sum of the lengths of the source variables. It is important to set the length of your variables that you create by using the CAT functions, so that you keep reasonable processing/storage costs while keeping the variable lengths long enough to hold the longest string created by concatenation.

Table 3 shows equivalents of the CAT, CATS, CATT, and CATX functions. The variables X1 through X4 specify character variables.
<table>
<thead>
<tr>
<th>Function</th>
<th>Equivalent Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT(of X1-X4)</td>
<td>X1</td>
<td></td>
</tr>
<tr>
<td>CATS(of X1-X4)</td>
<td>TRIM(LEFT(X1))</td>
<td></td>
</tr>
<tr>
<td>CATT(of X1-X4)</td>
<td>TRIM(X1)</td>
<td></td>
</tr>
<tr>
<td>CATX(SP, of X1-X4)</td>
<td>TRIM(LEFT(X1))</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Equivalent Statements to the CAT Family of Functions

If the variables X1 through X4 specify numeric variables, some of the equivalents in Table 3 will not apply. Let us see an example below to find out the differences:

**PROGRAM: CATNUM.SAS**

```sas
data catnum;
  X1=1;
  X2=23;
  X3=456;
  X4=7890;
  MyVar_Cat=cat(of X1-X4);
  MyVar_noCat=X1||X2||X3||X4;
run;
```

Display 1. SAS Log Produced By Running Program CATNUM.SAS
The CAT functions can be used with numeric variables without producing a note in the SAS log about numeric-to-character conversion; however, the concatenation operator (||) would produce such a message.

The concatenation operator converts each numeric variable to character by using the default format BEST12., and therefore the resulting variable “MyVar_noCat” has blanks in it. In contrast, the CAT function uses the format W., where W represents the width of the each variable X1 to X4, which is equivalent to the following two lines of code:

```sas
length MyVar $200;
MyVar=put(X1,1.)||put(X2,2.)||put(X3,3.)||put(X4,4.);
```

Again, since we didn’t assign the length for the resulting variables MyVar_Cat and MyVar_noCat, the length of MyVar_cat defaults to 200, and MyVar_noCat has the length of sum of each source variable which is 12*4=48.

**AN EXAMPLE: LET THE CAT CATCH A STYLE**

We have explored the features and advantages of both inline formatting functions and the CAT family of functions. Now we will demonstrate how to utilize the advantages of the CAT functions to concatenate the inline formatting functions to existing variables in the DATA step.

**SOURCE DATA SET:**

For our example, we made a data set that is a subset of the SASHELP.CARS data set, and we limited the variables and values as shown in the following table:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Type</th>
<th>Variable Values Used In Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>Character</td>
<td>'Cadillac', 'Lexus', 'Mazda', 'Mitsubishi'</td>
</tr>
<tr>
<td>Type</td>
<td>Character</td>
<td>'Sedan', 'Sports', 'SUV'</td>
</tr>
<tr>
<td>Model</td>
<td>Character</td>
<td></td>
</tr>
<tr>
<td>MPG_City</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td>MPG_Highway</td>
<td>Numeric</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4. Variables and Values Selected from SASHELP.CARS**
After calculating the average (mean) MPG in the City and MPG on the Highway for each categorized car type, our SAS data set looked like this:

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
<th>Type</th>
<th>MPG (City)</th>
<th>MPG (Highway)</th>
<th>highway_mean</th>
<th>city_mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cadillac</td>
<td>Sedan</td>
<td>18</td>
<td>25</td>
<td>27.5</td>
<td>20.5</td>
</tr>
<tr>
<td>2</td>
<td>Cadillac</td>
<td>Sedan</td>
<td>18</td>
<td>25</td>
<td>27.5</td>
<td>20.5</td>
</tr>
<tr>
<td>3</td>
<td>Cadillac</td>
<td>Sedan</td>
<td>18</td>
<td>25</td>
<td>27.5</td>
<td>20.5</td>
</tr>
<tr>
<td>4</td>
<td>Mitsubishi</td>
<td>Sedan</td>
<td>18</td>
<td>25</td>
<td>27.5</td>
<td>20.5</td>
</tr>
<tr>
<td>5</td>
<td>Lexus</td>
<td>Sedan</td>
<td>20</td>
<td>23</td>
<td>27.5</td>
<td>20.5</td>
</tr>
<tr>
<td>6</td>
<td>Mitsubishi</td>
<td>Sports</td>
<td>21</td>
<td>20</td>
<td>26.1</td>
<td>13.7</td>
</tr>
<tr>
<td>7</td>
<td>Mitsubishi</td>
<td>Sports</td>
<td>21</td>
<td>20</td>
<td>26.1</td>
<td>13.7</td>
</tr>
</tbody>
</table>

Display 4. SAS Data Set of the Example (Partial View)

**IDEAL OUTPUT:**

We wanted to generate a summary report (a WORD file) with output that looked like this:

**Comparing both CITY and HIGHWAY MPG**

<table>
<thead>
<tr>
<th>MAKE</th>
<th>SUV</th>
<th>TYPE</th>
<th>Sedan</th>
<th>Sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cadillac</td>
<td>Escalade</td>
<td>CTS</td>
<td>VVT 4dr</td>
<td>XLR convertible 2dr</td>
</tr>
<tr>
<td></td>
<td>SRX V8</td>
<td>Deville 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deville DTS 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Seville SLS 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexus</td>
<td>GX 470</td>
<td>ES 330 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LX 470</td>
<td>GS 300 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RX 350†</td>
<td>GS 430 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IS 300 4dr auto</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IS 300 4dr manual</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LS 430 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mazda</td>
<td>Tribute DX 2.0†</td>
<td>MPV ES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mazda6 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mazda6i 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>Endeavor XLS</td>
<td>Diamante LS 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Montero XLS</td>
<td>Galant ES 2.4L 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outlander LS†</td>
<td>Galant GTS 4dr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lancer ES 4dr†</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lancer LS 4dr†</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lancer OZ Rally 4dr auto†</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Eclipse GTS 2dr†</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Eclipse Spyder GT convertible 2dr†</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† This model has both higher CITY and HIGHWAY MPG than the average for its Type

Display 5. Desired ODS RTF Output

In our example summary report, we wanted to list the car models one per line in the Make/Type categorized cell, and we wanted to emphasize or highlight those car models that have both higher CITY and HIGHWAY MPG than the average for their Type. We wanted do this both by highlighting the car model with a different colored font and by adding a dagger sign at the end of the car model’s name.

**SOLUTIONS:**

First, we inserted the inline formatting functions ^{style [foreground=blue]} and ^{dagger} to those car models which have greater mpg_city and mpg_highway than the mean mpg for their Type.
Then, we transposed the data set to one record per make and type. Since each record has a different number of models, some model columns would be blank. The following shows a portion of the transposed data set:

![Display 6. Transposed SAS Data Set (Partial View)](image)

Next, we used the string concatenation function to join all the non-blank model names along with the inline formatting function `{newline} as a delimiter so that the output will have one model per line in the table’s cells.

The following table is a portion of the resulting data set:

![Display 7. Inline Formatting Functions Added to the “allmodel” Variable (Partial View)](image)

Finally, we used another PROC TRANSPOSE to make our final SAS data set ready for the ODS output.

**THE SAS CODE:**

The following two sections of SAS code show the differences between using the CAT functions and using TRIM and LEFT with the concatenation operator(||) for the concatenation.

```
**The First Concatenation;**
data mycars;
  format model $100.;
  set mycars;
  if mpg_city>city_mean and mpg_highway>highway_mean then
    model= "^{style [foreground=blue]|trim(left(model))|"^^{dagger}};"
  run;

**The Second Concatenation;**
data mycars;
  length allmodel $350;
  set mycars;
  array arr {*} model:;
  do i = 1 to dim(arr);
    if i=1 then allmodel=arr(i);
    else if i>1 and arr(i) ne " " then
      allmodel =trim(left(allmodel))|^{newline 1}|arr(i);
  end;
  drop i model:;
  run;
```

![Display 8. SAS Code Example Without CAT Functions](image)
**The First Concatenation;**
data mycars;
    format model $100.;
    set mycars;
    if mpg_city>city_mean and mpg_highway>highway_mean then
        model= cats("^{style [foreground=blue],model,"}","^{dagger}");
run;

**The Second Concatenation;**
data mycars;
    length allmodel $350;
    set mycars;
    allmodel = catx("^{newline 1}",of model:);
    drop model: ;
run;

Display 9. SAS Code Example With CAT Functions

By comparing the highlighted codes, you can see that by using CAT functions not only did we use fewer statements, we also simplified the logic and made the code more readable and understandable. Especially in the second concatenation, by using the CATX, we didn’t have to conditionally determine the value of each model to see if it is blank, since CATX conveniently only adds the inline function ^{newline} between the non-blank strings.

CONCLUSION

The inline formatting functions provide a very flexible way to enhance your ODS output. By concatenating those functions to existing variable values through the DATA step, the programmer can better customize the output of data cells. When compared to the original string concatenation method (using || with TRIM and LEFT), the CAT family of functions provides an easier and more efficient way to do this task.

REFERENCES


ACKNOWLEDGMENTS

The authors wish to thank the Data Management Center at Cincinnati Children’s Hospital Medical Center for its support, and particularly Matthew Fenchel and Kelly Olano for their helpful feedback.

CONTACT INFORMATION

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

Yanhong Liu
Cincinnati Children’s Hospital Medical Center
3333 Burnet Ave.
Cincinnati, OH 45255
(513) 803-2614
E-mail: Yanhong.Liu@cchmc.org

Justin Bates
Cincinnati Children’s Hospital Medical Center
3333 Burnet Ave.
Cincinnati, OH 45255
(513) 803-2756
E-mail: Justin.Bates@cchmc.org
APPENDIX

The Full SAS Program for the Example:

data cars;
set sashelp.cars;
if type in ('Sedan', 'Sports', 'SUV') and make in ('Cadillac', 'Lexus', 'Mazda', 'Mitsubishi');
keep type make mpg_city mpg_highway model;
proc sort; by model;
run;

proc sql noprint;
create table mycars as
  select a.*, b.highway_mean, b.city_mean from cars as a full join
  (select model, avg(mpg_city) as city_mean format 4.1, avg(mpg_highway) as highway_mean format 4.1
  from cars
  group by type) as b on a.model=b.model;
quit;

**The First Concatenation;
data mycars;
  format model $100.;
  set mycars;
  if mpg_city>city_mean and mpg_highway>highway_mean then model=cats("^\{style [foreground=blue]"", model,"}\{dagger\}"");
proc sort; by make type;
run;

proc transpose data=mycars out=mycars(drop=_name_) prefix=model;
  by make type;
  var model;
run;

**The Second Concatenation;
data mycars;
  length allmodel $350;
  set mycars;
  allmodel = catx("^\{newline 1}","{\newline 1}",of model:);
  drop model;
run;

proc transpose data=mycars out=mycars(drop=_name_);
  by make;
  id type;
  var allmodel;
run;
ods listing close;
ods rtf file="mycars.rtf" style=printer;
ods escapechar="^";
options nonumber nodate orientation=portrait;

Title "Comparing both CITY and HIGHWAY MPG";
proc report data=mycars
  NOWINDOWS
  SPACING=1
  HEADING
  HEADSPLIT="|"
style(header)=[background=lightgrey];

columns ("TYPE| SUV Sedan Sports");
define Make /display style={cellwidth=3 cm} style(header)=[bordertopcolor=lightgrey] "MAKE ";
define SUV /display left style={cellwidth=5 cm} "SUV";
define Sedan /display left style={cellwidth=5 cm} "Sedan";
define Sports /display left style={cellwidth=5 cm} "Sports";
run;

ods rtf text="This model has both higher CITY and HIGHWAY MPG than the average for its Type";
ods rtf close;
ods listing;