

## Paper 205-2010

**Purrfectly Fabulous Feline Functions**

Louise S. Hadden, Abt Associates Inc.

**ABSTRACT**

Explore the fabulous feline functions and calls available in SAS® 9.1 and later. Using CAT functions and CAT CALLS gives you an easier way to streamline your SAS code and facilitate concatenation of character strings. So, leave verbose coding, myriad functions, and the vertical bar concatenation operators behind! SAS® 9.2 enhancements will also be demonstrated.

**INTRODUCTION**

```
LENGTH stcounty $ 5 citystzip $ 60;

stcounty=TRIM(LEFT(PUT(state,Z2.))) || TRIM(LEFT(PUT(county,Z3.)));

citystzip=TRIM(LEFT(city)) || ', ' || TRIM(LEFT(statecode)) || ' ' || TRIM(LEFT(zip));
```

Version 9 introduced four new functions (CAT, CATS, CATT, and CATX) and three new CALL routines (CALL CATS, CALL CATT, and CALL CATX) that replace the clunky syntax shown above, and add some additional enhanced capability to character string concatenation. In Version 9.2, CATQ was added. We will explore the five (fabulous) CAT functions and three CAT CALLS and demonstrate how string concatenation can be accomplished more easily and efficiently using these functions and call routines. In addition, the differences between the CAT functions and CAT CALLS will be briefly discussed. Examples were run with SAS 9.2 on Windows XP, and SAS 9.2 on Windows Server (x64). Leaving appropriate spaces out is deliberate so that the action of the functions and CALL routines can be observed. Performance using the concatenation operator, CAT functions and CAT CALL routines will be compared on the two different platforms using an identical one million record file.

**ORIGINS OF CAT CALLS AND FUNCTIONS**

The customary syntax seen above, `varx=TRIM(LEFT(vary)) || TRIM(LEFT(varz))`, consists of two functions (TRIM and LEFT) paired with the concatenation operator ( || ). TRIM removes trailing blanks, while LEFT left aligns text values. LEFT does not remove leading blanks. It simply moves leading blanks to the end of the string, which is why you usually see LEFT used in conjunction with (and inside) the TRIM function for string concatenation. Over the years, SAS has enhanced the function (pardon the pun) of TRIM and LEFT. A newer function TRIMN also strips trailing blanks: the difference between TRIM and TRIMN is the end result in the case of the argument being missing. TRIM results in a single blank (length 1) for a missing argument while TRIMN results in a null (length 0.) This might come in handy, for example, when concatenating first, middle and last names, in which middle names are frequently missing. Another newer function, STRIP, is the equivalent of TRIMN(LEFT(varx)) but is obviously more convenient. STRIP removes both leading and trailing blanks. The CAT CALLS and functions build on the original and derivative functions to offer a complete array of concatenation options.

All examples shown below were run on a Windows X64 server using SAS version 9.2, on one million records. In this case, all arguments being concatenated are character, and the second (of three) arguments is blank. Note that it is not necessary for all arguments to be character with the CAT CALLS and functions, and that arguments may be of mixed type. In addition, results of CAT functions (but not CAT CALLs) may be numeric. Unlike with the concatenation operator, you will not get a warning in your log if you use numeric arguments with the CAT CALLs and functions.

**EXAMPLES AND RESULTS OF USING THE CONCATENATION OPERATOR AND THE CAT FUNCTION****Example 1: Using concatenation operator****SAS Log:**

```

35      data temp1;
36          set dd.sub2003asm ;
37          fullname=aala||aalb||aalc;
38      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP1 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.54 seconds
      cpu time           0.54 seconds

```

**SAS list:**

fullname

ERMA FLETCHER

**PROC CONTENTS output:**

Variables in Creation Order				
#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	31	

**Example 2: Using concatenation operator and trim/trimn/left functions****SAS Log:**

```

49      data temp2;
50          set dd.sub2003asm ;
51          fullname=TRIM(LEFT(aala))||trimn(left(aalb))||TRIM(LEFT(aalc));
52      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP2 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.76 seconds
      cpu time           0.76 seconds

```

**SAS list:**

fullname

ERMAFLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	31	

**Example 3: Using the CAT function****SAS Log:**

```

63      data temp3;
64          set dd.sub2003asm ;
65          fullname=cat(aala,aalb,aalc);
66      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP3 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.68 seconds
      cpu time           0.68 seconds

```

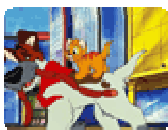
**SAS list:**

fullname

ERMA FLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	200	

**DIFFERENCES BETWEEN CALL ROUTINES AND FUNCTIONS**

The primary differences between the CAT CALLS and the corresponding functions are in syntax and performance with one exception. Results for the CAT CALLS MUST be initialized, and MUST be character, while results for CAT functions SHOULD be initialized (but don't HAVE to be.) Results are equivalent for both CAT CALLS and CAT functions. CALL routines have slightly better performance (i.e. they run faster) than functions that result in the same product.

The syntax differs in that in call routines, the result is incorporated into the statement, or CALL. In functions, the result is on the left hand side of an equal sign and the function statement.

**EXAMPLES AND RESULTS OF THE CATS FUNCTION AND CALL CATS****Example 4: Using the CATS function with a length statement****SAS Log:**

```

139      data temp7a;
140          length fullname $ 40;
141          set dd.sub2003asm ;
142          fullname=cats(aala,aalb,aalc);
143      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP7A has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.53 seconds
      cpu time           0.53 seconds

```

**SAS list:**

fullname

ERMAFLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

**Example 5: Using CALL CATS (initialization required)****SAS Log:**

```

138      data temp8;
139          length fullname $ 40;
140          set dd.sub2003asm ;
141          call cats(fullname,aala,aalb,aalc);
142      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP8 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.50 seconds
      cpu time           0.50 seconds

```

**SAS list:**

fullname

ERMAFLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

In the end, anything that increases processing speed and efficiency while reducing the keystrokes necessary to accomplish goals is purrfectly fine with most SAS programmers! For concatenation, the hierarchy of efficiency is CAT CALLS at the top, CAT FUNCTIONS in the middle, and an assemblage of functions and the concatenation operator at the bottom.

**IMPORTANT NOTE ON FELINE FUNCTIONS AND VARIABLE LENGTH**

SAS calculates a length for resulting variables created by using the concatenation operator by adding the lengths of the source variables, or arguments, together. This is very different from how the CAT functions operate. If not specified in a length statement prior to the invocation of a function (or length specified via some other method), the length of the result of the CAT functions defaults to 200. It is important to set the length of your variables that you will create from a character function to keep variable lengths, processing costs and storage costs reasonable. Conversely, it is also important to set the length of created variables long enough to accommodate the longest string created by concatenation, so that your result is not truncated. In the case of truncation with the CAT functions, you receive a warning in your log, but the variable will be created (incorrectly.)

**Log Snippet courtesy of Mike Zdeb:**

```
152 data test;
153 length full1 full2 $15;
154 input (first last) (: $15.);
155 full1 = catx(' ',last,first);
156 full2 = trim(last) || ' ' || first;
157 datalines;
```

WARNING: In a call to the CATX function, the buffer allocated for the result was not long enough to contain the concatenation of all the arguments.  
The correct result would contain 17 characters, but the actual result may either be truncated to 15 character(s) or be completely blank, depending on the calling environment. The following note indicates the left-most argument that caused truncation.

NOTE: Argument 3 to function CATX at line 155 column 9 is invalid.

```
RULE:      ----+----1-----2-----3-----4-----5-----6-----7---
-+-----8-----9-----0
```

```
160          ARMONDO D'BOUVIER
full1= full2=D'BOUVIER,ARMON first=ARMONDO last=D'BOUVIER _ERROR_=1 _N_=3
```

As you can see in the examples below, use of the length statement also significantly increases processing speed when you reduce the length of the variable from 200 to 40.

**EXAMPLES AND RESULTS OF THE CAT FUNCTION WITH AND WITHOUT A LENGTH STATEMENT****Example 6: Using the CAT function****SAS Log:**

```

63      data temp3;
64          set dd.sub2003asm ;
65          fullname=cat(aala,aalb,aalc);
66      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP3 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.68 seconds
      cpu time           0.68 seconds

```

**SAS list:**

fullname

ERMA            FLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
2	RESIDENT	Num	7	RES-INT-ID
3	FACILITY	Num	7	FAC-INT-ID
4	AA1A	Char	12	AA1A-FIRST-NM
5	AA1B	Char	1	AA1B-MIDDLE-INTIAL
6	AA1C	Char	18	AA1C-LAST-NM
7	fullname	Char	200	

**Example 7: Using the CAT statement with a length statement****SAS Log:**

```

109     data temp6;
110         length fullname $ 40;
111         set dd.sub2003asm ;
112         fullname=cat(aala,aalb,aalc);
113     run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP6 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.45 seconds
      cpu time           0.45 seconds

```

**SAS list:**

fullname

ERMA            FLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

**THE CAT FUNCTIONS (CAT, CATS, CATT, CATX AND CATQ)****CAT (available in 9.1)**

The result of the CAT function is identical to simply using the concatenation operator ( || ) to string character literals, variables or expressions. No leading or trailing blanks are removed from character arguments; however, leading zeroes are removed from numeric arguments consistent with the use of the BEST format to convert numeric arguments.

See examples 6 and 7 above.

**CATS (available in 9.1)**

The CATS function removes both leading and trailing blanks before concatenating arguments. It is the equivalent of using the STRIP function and the concatenation operator. The end result will not contain any blanks between arguments, even if you specify a blank as an argument, which might (or might not) be what you want. A good mnemonic device is that "S" is equivalent to "STRIP". Character arguments are stripped of any leading or trailing blanks before being joined. Embedded blanks within character arguments are preserved (for example, a first name of "Elly Mae".)

See example 4 above.

**CATT (available in 9.1)**

The CATT function removes only trailing blanks before concatenating arguments. The end result will contain blanks if any arguments have LEADING or EMBEDDED blanks. A good mnemonic for this function is "T" is equivalent to "TRAILING" or "TRUNCATE." Trailing blanks are removed from arguments before they are joined.

**EXAMPLE AND RESULTS OF THE CATT FUNCTION**

Example 8: Using the CATT function with a code shortcut

**SAS Log:**

```

210      data temp12;
211          length fullname $ 40;
212          set dd.sub2003asm (rename=(aala=var1 aalb=var2 aalc=var3)) ;
213          fullname=catt(of var1-var3);
214      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP12 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

real time          0.51 seconds
cpu time           0.51 seconds

```

**SAS list:**

fullname

ERMAFLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	var1	Char	12	AA1A-FIRST-NM
6	var2	Char	1	AA1B-MIDDLE-INTIAL
7	var3	Char	18	AA1C-LAST-NM

**CATX (available in 9.1)**

The CATX function performs just like the CATS function, i.e. stripping trailing and leading blanks from arguments before joining, but it adds the eXtra capability of inserting the delimiter of your choice between arguments. This is handy if you want a space or comma (or something else) between arguments (such as a full name or address) but don't want any eXtra spaces. While you could use another function, COMPBL, to remove excess blanks, CATX does it all at once. A good mnemonic for CATX is "X" is for "eXtra" – the CATX function "adds" a delimiter (or something extra) between arguments.

**EXAMPLE AND RESULTS OF THE CATX FUNCTION**

Example 9: Using the CATX function

**SAS Log:**

```

224      data temp13;
225          length fullname $ 40;
226          set dd.sub2003asm ;
227          fullname=catx(" ",aala,aalb,aalc);
228      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP13 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.56 seconds
      cpu time           0.56 seconds

```

**SAS list:**

fullname

ERMA FLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID



4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM

### CATQ (available in 9.2)

The CATQ function is the newest member of the fabulous feline function litter. It is similar to the CATX function, with the added option of quotation marks around arguments in the end result. There is an array of modifiers that can be used with this function to finely control the result. The online documentation for SAS 9.2 (Language Reference: Dictionary) is a good and comprehensive discussion of the many modifiers available and the syntax for using these modifiers, among which are [not a comprehensive list!]: 1 (use single quotation marks), 2 (use double quotation marks), a (add quotation marks to all arguments), c (use a comma as a delimiter), s (trim leading and trailing blanks from arguments), t (trim trailing blanks from arguments), and x (convert item arguments to hexadecimal literals if they contain nonprintable characters.) A good mnemonic for CATQ is "Q" is for "quotation marks"; the CATQ function has the capacity to add single or double quotation marks to selection portions of the result if desired using the many modifiers available. Remember to add some extra length for the quotation marks, or you will get an error if the results of the CATQ function exceed the specified length.

### EXAMPLE AND RESULTS OF THE CATQ FUNCTION

Example 10: Using the CATQ function

#### SAS Log:

```

252      data temp15;
253          length fullname $ 50;
254          set dd.sub2003asm ;
255          fullname=catq('as',aala,aalb,aalc);
256      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP15 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.60 seconds
      cpu time           0.60 seconds

```

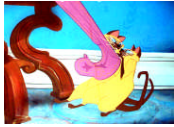
#### SAS list:

fullname

"ERMA" " " "FLETCHER"

#### PROC CONTENTS output:

#	Variable	Type	Len	Label
1	fullname	Char	50	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM



## IMPORTANT NOTE ON CALLING CATS AND VARIABLE LENGTH

The result variable **MUST** be initialized prior to the CALL routine, and must be initialized as character. If the result variable is not initialized as character, SAS assumes the default (that the result variable is numeric) and an error occurs (a missing value is produced.) If you initialize the result variable to a single blank, the result will be truncated at a length of one. Therefore, as with CAT functions, it is a good practice to initialize by specifying the length of the resulting variable directly under the data statement, making sure to set the length of created variables long enough to accommodate the longest string (or number) created by concatenation.



## CALLING THE CATS

CALL CATS, CALL CATT and CALL CATX all behave more or less like their corresponding functions described above, except that the resulting character variable is included in the CALL. For example:

```
CALL CATS(result,item1,item2,...itemn);
```

Items, or arguments, can be numeric instead of character; numeric arguments are converted to character using BESTw. Format. The result **MUST** be a character variable, and the length of the character variable should be long enough to accommodate the result. As discussed above, this result variable must be initialized (and the appropriate length set) before invocation of the CAT CALL.

You can also use the short cut of (of var1-varn) in CAT CALL routines and CAT functions. For example:

```
CALL CATT(result,of item1 - itemn);
```

Also see example 8 above and example 11 below.

It is important to note that unlike the CATX function, CALL CATX does not include a delimiter surrounding blank arguments. This may, or may not, be what you want.

For an example of CALL CATS, see Example 5 above.

## EXAMPLE AND RESULTS OF CALL CATT

Example 11: Using CALL CATT (initialization Required)

### SAS Log:

```
195      data temp11;
196          length fullname $ 40;
197          set dd.sub2003asm (rename=(aala=var1 aalb=var2 aalc=var3)) ;
198          call catt(fullname,of var1-var3);
199      run;
```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP11 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```
real time          0.48 seconds
cpu time           0.48 seconds
```

**SAS list:**

fullname

ERMAFLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	var1	Char	12	AA1A-FIRST-NM
6	var2	Char	1	AA1B-MIDDLE-INTIAL
7	var3	Char	18	AA1C-LAST-NM

**EXAMPLE AND RESULTS OF CALL CATX**

-----

Example 12: Using CALL CATX (initialization required)

**SAS Log:**

```

238      data temp14;
239          length fullname $ 40;
240          set dd.sub2003asm ;
241          call catx(" ",fullname,aala,aalb,aalc);
242      run;

```

NOTE: There were 1000000 observations read from the data set DD.SUB2003ASM.

NOTE: The data set WORK.TEMP14 has 1000000 observations and 7 variables.

NOTE: DATA statement used (Total process time):

```

      real time          0.53 seconds
      cpu time           0.53 seconds

```

**SAS list:**

fullname

ERMA FLETCHER

**PROC CONTENTS output:**

#	Variable	Type	Len	Label
1	fullname	Char	40	
2	STATE	Char	2	3.3: ALPHA STATE CODE FROM FIRST MDS RECORD IN STAY
3	RESIDENT	Num	7	RES-INT-ID
4	FACILITY	Num	7	FAC-INT-ID
5	AA1A	Char	12	AA1A-FIRST-NM
6	AA1B	Char	1	AA1B-MIDDLE-INTIAL
7	AA1C	Char	18	AA1C-LAST-NM



## SOME PRACTICAL USES

In addition to the obvious utility of combining strings to create a full name or address, there are many possible uses of these powerful functions and call routines. A search on SAS-L or sascommunity.org will come up with a number of examples, a couple of which are shown below. Also highly recommended is Mike Zdeb's NESUG 2009 paper, "Searching for Variable Values with CAT Functions: An Alternative to Arrays and Loops."

**On SAS-L (courtesy of a tip from Mike Zdeb):**

### Search for characters in numeric variables

```
data dxplus;
  set dx;
  array prx(3) diag1-dia3;
  do _n_ = 1 to 3 until (tag eq 1);
    tag = (cat(prx(_n_)) in : ('171' '172'));
  end;
run;
```

### CAT and FIND with repeated concatenation

```
data cat_find_repeat;
  set temp;
  dia = (find(catx('*', '*', adx, pdx, of odx:), '*250') gt 0);
  ast = (find(catx('*', '*', adx, pdx, of odx:), '*493') gt 0);
  ami = (find(catx('*', '*', adx, pdx, of odx:), '*410') gt 0);
  brc = (find(catx('*', '*', adx, pdx, of odx:), '*174') gt 0);
  flu = (find(catx('*', '*', adx, pdx, of odx:), '*487') gt 0);
run;
```

**From Louise Hadden**

### Create a paragraph to print in a report (partial code)

```
data page2_blurb;
  length blurb blurb1 blurb2 $ 8128 sentence7a sentence7b sentence7c
  sentence7d $ 200;

  sentence7a="^n^n - Based on this total score, the top 10% of nursing homes
  nationwide";
  sentence7b=' get 5 stars, the bottom 20% get 1 star, and the middle 70% of
  nursing homes receive';
  sentence7c=' 2, 3 or 4 stars, with an equal proportion (23.33%) in each
  category.';
  sentence7d=' The thresholds for this distribution are also fixed, rather than
  being recalculated each month.';

  blurb2=catt(sentence6a,sentence6b,sentence6c,sentence6d,sentence6e,sentence7a,
  sentence7b,sentence7c,sentence7d);
  blurb=catt(blurb1,blurb2);

  label blurb =" ";

run;
```

**Results in:**

- Points are assigned for each QM based on what quintile the nursing home falls into, in comparison to other nursing homes. For the 2 ADL QMs, these quintile thresholds are set each month; for the other 8 QMs, fixed quintile boundaries are used. Points for each QM are added together for a total point score.



- Based on this total score, the top 10% of nursing homes nationwide get 5 stars, the bottom 20% get 1 star, and the middle 70% of nursing homes receive 2, 3 or 4 stars, with an equal proportion (23.33%) in each category. The thresholds for this distribution are also fixed, rather than being recalculated each month.

*Special Tip: Use an invisible character (ALT-255) to label variables you want to suppress a label for. That's not really a null in the label statement above.*

## THE LITTER BOX (OR CATNIPPED!)

Along with these powerful functions and CALL routines there are some warnings above and beyond the default length of 200.

- Using mixed variable types in conjunction with `_all_` in CAT functions can result in a numeric outcome with unexpected results. To avoid this, use a length statement for your outcome variable.
- If you are searching a number of strings for a particular set of characters (as shown in above), and you concatenate the strings before searching without an argument separating diagnoses, SAS does not see a difference between 250250 and 025025 as far as the search goes. It will find a code 250 (or code 025) in either string. The solution is to use CATX which inserts a delimiter between diagnoses.
- To avoid the danger of truncation of pre-initialized results, take the maximum length of all input variables (note – use the LENGTHC function instead of the LENGTH or LENGTHN function to get the trailing blanks if using CAT!) and add them together, making sure to add space for any delimiters and/or quotes you add with CATX, ALL CATX or CATQ.

## CONCLUSION

Use of the fabulous feline functions and CALL routines saves time in both coding and in processing. While results of using the combination method, CAT functions and CAT CALL routines are all the same, the feline functions and CALL routines use less code and process concatenations faster.

It is purrfectly clear that the CAT functions and CALL routines are valuable additions to the SAS® programmer's toolbox. Try them out, and you'll be become a CAT lover too!

## REFERENCES & RECOMMENDED READING

<http://support.sas.com/documentation/onlinedoc/base/index.html>

<http://support.sas.com/rnd/papers>

<http://support.sas.com/samples>

<http://support.sas.com/documentation/cd/en/lrdict/59540/HTML/default/a000245852.htm>

<http://support.sas.com/documentation/cd/en/whatsnew/61982/HTML/default/lrdictwhatsnew902.htm#a002986580>

<http://support.sas.com/kb/24/513.html>

Cody, Ron. 2004. *SAS® Functions by Example*. Cary, NC: SAS Institute Inc.

Tsykalov, Eugene and Yeh, Shi Tao. "The Invisible Character Alt-255 – Hidden Dragon: Hiding And Aligning Text in SAS® Output." Proceedings of the Thirty-First Annual SAS® Users Group International Conference. March 2006.

Zdeb, Mike. "Searching for Variable Values with CAT Functions: An Alternative to Arrays and Loops." Proceedings of NESUG 2009 Conference, September 2009.

## ACKNOWLEDGMENTS & DISCLAIMERS

I owe many thanks for the invaluable assistance of Mike Zdeb, Jason Secoskey, and Ron Cody in the writing of this paper and my search for understanding of the fabulous feline functions (no CAT calls here!)

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks 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.

All images are free to download from [www.disney.com](http://www.disney.com). They are from the movies Lady and the Tramp (the famous Si and Am), the Aristocats, and Oliver and Company.

## CONTACT INFORMATION

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



Louise Hadden  
Abt Associates Inc.  
55 Wheeler St.  
Cambridge, MA 02138

Email: [louise\\_hadden@abtassoc.com](mailto:louise_hadden@abtassoc.com)

Sample code is available from the author upon request.

## Appendix A: Benchmarks on CAT functions and CAT CALLs, one million records

Category	CPU Time Windows Server x64 with multiple processors SAS 9.2	CPU Time Windows XP P4 SAS 9.2	Length of Result
Concatenation operations alone / no length statement	.54sec	.66sec	31 (sum of variable lengths)
Concatenation operations with TRIM/TRIMN/LEFT functions / no length statement	.76sec	.88sec	31
CAT function / no length statement	.68sec	1.34sec	<b>200</b>
Concatenation operations alone / length statement	.51sec	.70sec	40
Concatenation operations with TRIM/TRIMN/LEFT functions / length statement	.79sec	.95sec	40
CAT function / length statement	.46sec	.93sec	40
CATS function / no length statement	.73sec	1.38sec	<b>200</b>
CATS function / length statement	.53sec	.78sec	40
CALL CATS*	.49sec	.99sec	40
CATT function / no length statement	.71sec	1.12sec	<b>200</b>
CATT function / length statement / code shortcut	.51sec	.89sec	40
CALL CATT*	.48sec	.92sec	40
CALL CATT* / code shortcut	.48sec	.80sec	40
CATX function / length statement / insert blank	.56sec	.85sec	40
CALL CATX* / insert blank	.53sec	1.01sec	40
CATQ function / length statement	.60	1.04sec	<b>50</b>

\*Initialization of result variable required in CAT CALLs