Let the CAT Out of the Bag: String Concatenation in SAS 9

Joshua Horstman
Nested Loop Consulting

Outline

- Life before SAS 9: TRIM and LEFT functions
- CAT Family of Functions – definitions, advantages, warnings
- Examples of clever use of CAT functions

Does your code look like this?

```
name = trim(left(last_name))||', '||
       trim(left(first_name))||' '||
       trim(left(middle_initial))||'. ';
```

Things Best Left in the 90's

TRIM(LEFT(var1)) || TRIM(LEFT(var2))

Before SAS 9: TRIM and LEFT

- TRIM: Removes trailing blanks from a character string
- LEFT: Left-aligns a character string (moving leading blanks to the end)
- TRIM and LEFT are frequently used to remove both leading and trailing blanks, often prior to concatenation.

What’s wrong with TRIM/LEFT?

- Nothing - both still have their place.
- Code can become cumbersome and difficult to read/edit
- SAS 9 gives us new options

The STRIP Function

- New in version 9
- Returns the same result as TRIMN(LEFT(…))
- TRIMN returns zero-length string for a blank string, whereas TRIM returns a single blank.
- STRIP improves the readability of our code, but we can still do better…

The CAT Family of Functions

- New in SAS 9.0: CAT, CATS, CATT, CATX
- New in SAS 9.2: CATQ

CAT

- Concatenates string variables

CAT(var1, var2, var3)
is equivalent to
var1 || var2 || var3

CATT Function

- Removes trailing blanks,
  then concatenates string variables

CATT(var1, var2, var3)

is equivalent to

TRIM(var1) || TRIM(var2) || TRIM(var3)

CATS Function

- Removes leading and trailing blanks, then concatenates string variables

CATS(var1, var2, var3)
is equivalent to
TRIM(LEFT(var1)) || TRIM(LEFT(var2)) || TRIM(LEFT(var3))

CATX Function

- Removes leading and trailing blanks, inserts delimiters, then concatenates string variables

CATX(' ',var1, var2, var3)
is equivalent to
TRIM(LEFT(var1)) || ' ' || TRIM(LEFT(var2)) || ' ' || TRIM(LEFT(var3))

CATQ Function

- Inserts delimiters, quotes strings with embedded delimiters, then concatenates.
- Numerous options available to modify behavior, many of which can be combined.
- Can trim or strip blanks, quote all items, quote strings containing quotes, and more.

Advantages of CAT Functions

- Availability of OF syntax for variable lists:

CAT(OF X1-X4)

is equivalent to

X1 || X2 || X3 || X4

Advantages of CAT Functions

- No need to write complex logic to prevent duplicate delimiters due to missing values.
- CATX and CATQ functions handle this automatically.

Advantages of CAT Functions

- Numeric values are automatically converted without additional notes to the log.
- Numeric values with traditional concatenation operators will generate a “NOTE: Numeric values have been converted to character values …” unless explicitly converted (e.g. using the PUT function).

A note about Variable Lengths

- CAT functions return variable with length 200 unless previously assigned another length.
- Concatenation operator returns a variable with length equal to the sum of the lengths of the values being concatenated.
- This can cause results to differ.

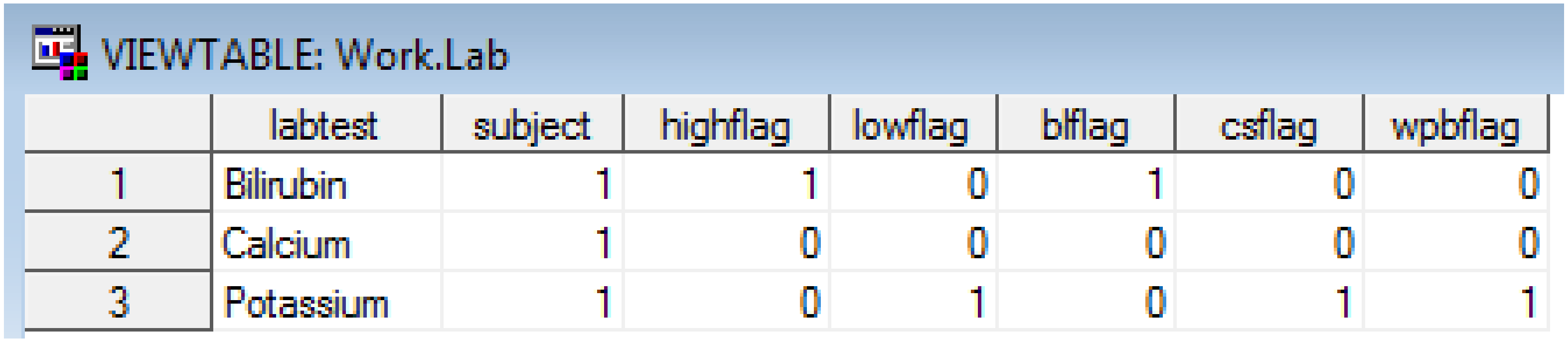
Example #1 – Setup

- Goal: Add variable to clinical laboratory data with applicable flags separated by commas.

- A record may have zero or more flags:
  - H = High (above normal range)
  - L = Low (below normal range)
  - CS = Clinically Significant
  - B = Baseline result
  - WPB = Worst-case post-baseline result

- Example adapted from Fecht 2012 paper.

Example #1 - Data



VIEWTABLE: Work.Lab

	labtest	subject	highflag	lowflag	blflag	csflag	wpbflag
1	Bilirubin	1	1	0	1	0	0
2	Calcium	1	0	0	0	0	0
3	Potassium	1	0	1	0	1	1

Example #1 – Old School Solution

```
data lab2a;
   set lab;
   need_comma = 0;
   length flaglist $12;
   if highflag then do;
      flaglist = 'H';
      need_comma = 1;
   end;
   if lowflag then do;
      if need_comma then flaglist = trim(left(flaglist))||',';
      flaglist = trim(left(flaglist))||'L';
      need_comma = 1;
   end;
   if blflag then do;
      if need_comma then flaglist = trim(left(flaglist))||',';
      flaglist = trim(left(flaglist))||'B';
      need_comma = 1;
   end;
   if csflag then do;
      if need_comma then flaglist = trim(left(flaglist))||',';
      flaglist = trim(left(flaglist))||'CS';
      need_comma = 1;
   end;
   if wpbflag then do;
      if need_comma then flaglist = trim(left(flaglist))||', ';
      flaglist = trim(left(flaglist))||'WPB';
   end;
run;
```

Example #1 – Modern Solution

```
data lab2b;
   set lab;
   flaglist = catx(',',
      ifc(highflag,'H'  ,''),
      ifc(lowflag ,'L'  ,''),
      ifc(blflag  ,'B'  ,''),
      ifc(csflag  ,'CS' ,''),
      ifc(wpbflag ,'WPB',''),
      );
run;
```

Example #1 - Results

```
proc print data=lab2b noobs;
    var subject labtest flaglist;
run;
```

```
   subject      labtest      flaglist

      1        Bilirubin     H,B
      1        Calcium
      1        Potassium     L,CS,WPB
```

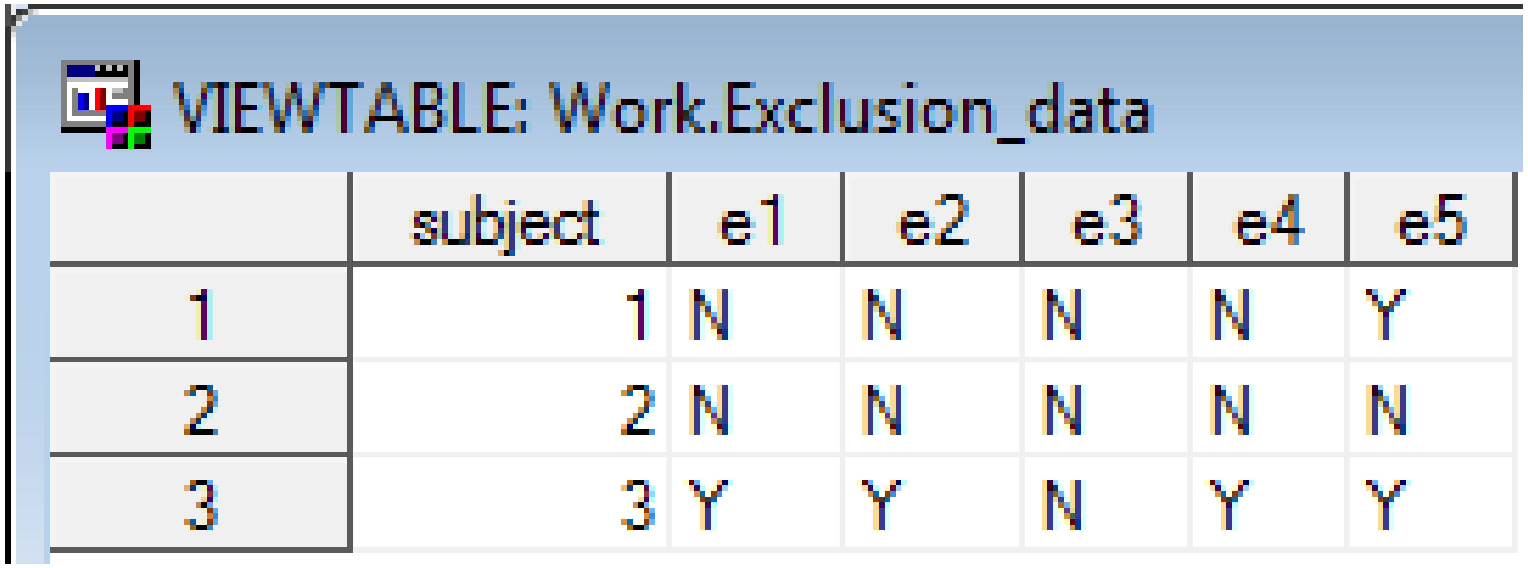
Example #2 - Setup

- Data: Yes/No responses to a series of exclusion criteria questions given to potential subjects in a clinical trial

- Goal: Extract list of subjects with one or more Yes answer

- Example adapted from Zdeb 2009 paper.

Example #2 - Data


VIEWTABLE: Work.Exclusion_data

	subject	e1	e2	e3	e4	e5
1	1	N	N	N	N	Y
2	2	N	N	N	N	N
3	3	Y	Y	N	Y	Y

Example #2 – Old School Solution

```
data excluded_subjects;
   set exclusion_data;
   array e(5);
   do i=1 to 5 until (e(i) = 'Y');
   end;
   if i < 6;
   drop i;
run;
```

Example #2 – Modern Solution

```
data excluded_subjects2;
   set exclusion_data;
   if find(cat(of e1-e5),'Y');
run;
```

Example #2 – Results

```
proc print data=excluded_subjects2 noobs;
    var subject;
run;
```

```
subject

   1
   3
```

Conclusions

- CAT functions:
  - Are convenient for the programmer.
  - Can simplify your code and improve readability.
  - Offer new ways to solve old problems.
- Savvy SAS programmers use both legacy and new functions as needed for the job.

References

Fecht, Marje. Quick Hits - My Favorite SAS® Tricks. Proceedings of the SouthEast SAS Users Group, Durham, NC, 2012.

Hadden, Louise S. Purrfectly Fabulous Feline Functions. Proceedings of the SAS® Global Forum 2010 Conference. Cary, NC: SAS Institute Inc., 2010.

Zdeb, Mike. Searching for Variable Values with CAT Functions: An Alternative to Arrays and Loops. Proceedings of the NorthEast SAS Users Group, Burlington, VT, 2009.

Contact Information

Joshua M. Horstman
Nested Loop Consulting
317-815-5899
josh@nestedloopconsulting.com