Paper 090-2008

Dynamic Decimal Precision and Alignment in Clinical Trial Laboratory Summary Tables and Patient Data Listings

Stephen Hunt, ICON Clinical Research, Redwood City, CA
Brian Fairfield-Carter, ICON Clinical Research, Redwood City, CA

ABSTRACT

When presenting descriptive statistics for laboratory parameters in clinical trial reporting, the experienced SAS® programmer frequently laments the disparity of numeric precision between varying parameters contained within the same dataset. Not only does this present an annoyance in summary tables, occasionally, supporting listings will require a 'vertical' structure in which the results from multiple parameters are stacked within the same column of the report. When faced with such a task, many programmers typically resort to a solution based almost entirely on an extended set of 'if/then' statements for assigning a particular presentation to a specific analyte(s) based on observational knowledge of the data at hand. In contrast, this paper demonstrates a simple yet effective method for dynamically presenting mixed laboratory results in both summary tables and vertical listings while preserving consistent decimal alignment as well as appropriate numeric precision.

INTRODUCTION

This paper demonstrates a relatively straight-forward means of achieving dynamic decimal alignment and numeric summary precision when presenting laboratory results in a clinical reporting setting. The primary tool utilized in achieving these ends is revealed via intermediate-level data manipulation of the 'BEST.' format and use of the SAS® macro language. However, before launching into an onslaught of macro syntax, it's helpful to get an introductory idea of what is meant by decimal alignment and numeric precision.

DECIMAL ALIGNMENT

A precept of reporting (and perhaps a minor justification for the very existence of clinical trial SAS programmers) is that numeric results should be concise, representative, and readable. The focus of decimal alignment is specifically with the readability aspect. In the following examples, platelets (expressed as an absolute count) and hematocrit (reported as a ratio) are juxtaposed within a typical summary table and supporting listing.

Decimal alignment within a summary table…

Platelet Count (10^9/L)		
Baseline	N	73
	Mean (SD)	259.7 (88.57)
	Median (Q1-Q3)	238.0 (200.0-295.0)
	Min-Max	92-499
Hematocrit (Ratio)		
Baseline	N	73
	Mean (SD)	0.423 (0.0634)
	Median (Q1-Q3)	0.430 (0.380-0.470)
	Min-Max	0.29-0.53

Decimal alignment within the supporting listing…

Subject	Parameter	Visit	Date	Result	Unit
123-456	Hematocrit	Baseline	27OCT2005	0.32	Ratio
		Visit 1	28OCT2005	0.34	
		Visit 2	29OCT2005	0.33	
		Visit 3	31OCT2005	0.29	
	Platelets	Baseline	27OCT2005	135	10^9/L
		Visit 1	28OCT2005	115	
		Visit 2	29OCT2005	123	
		Visit 3	31OCT2005	137	

When creating character fields intended to be decimal aligned in a vertical presentation, it's key to keep in mind the way SAS interprets numeric formats. For instance, forcing a numeric field into character by applying a "8.2" format

tells SAS that the overall number of decimal places allowed (including the decimal itself) is 8, and the number of places allocated past the decimal is 2 (i.e., XXXXX.XX). In order for an integer to align with a character field created via the aforementioned 8.2 numeric format, a width would need to be applied that takes into account the absence of any decimal (e.g., XXXXX). Practically speaking, any format applied to an integer value can be used as a 'starting' point to set up a cascading set of numeric formats for aligning decimals via the following algorithm:

Integer (0 decimal places): x [e.g., y=put(z, 5.0)]
1 decimal place: (x+2) <decimal> 1 [e.g., y=put(z, 7.1)]
2 decimal places: (x+3) <decimal> 2 [e.g., y=put(z, 8.2)]
3 decimal places: (x+4) <decimal> 3 [e.g., y=put(z, 9.3)]
Etc.…

The example above uses an assumption of 5 places to the left of the decimal as adequate to present any of the possible integer values of z found in the data, and then it follows by assigning match-width formats to align results of varying numbers of places past the decimal with the initial, hypothetical integer.

NUMERIC PRECISION (I.E., HOW MANY DECIMAL PLACES ARE APPROPRIATE?)

As mentioned above, readability, while useful, certainly isn't the primary function of summary statistics. Accuracy is a far more important aspect. Your favorite statistician and/or medical writer would be quite distressed to discover a standard deviation for specific gravity formatted to only 2 places past the decimal, no matter the tidiness of the presentation. However, such reporting would be perfectly appropriate when summarizing platelet counts.

Numeric precision is defined as a measure of the detail in which a quantity is expressed. To that end, a reasonable rule-of-thumb (and there are many) for numeric precision in clinical trial tables is suggested in the following:

(1) Minima and maxima should be displayed with the greatest number of decimals as collected on the CRF or presented in the database (qualifier: non-rounded converted values are exempt from the latter part of this rule)
(2) Mean, median, and any quartiles should be displayed with one decimal place further to the right than maxima
(3) Standard deviation and standard error should be displayed with two decimal places further to the right than maxima

WHAT IS THE 'BEST' PRECISION ANYWAY?

To a certain extent, SAS has a built-in method to evaluate any numeric field with regard to precision when no format has been explicitly applied to a host of data. This is the "BEST." format. For example, in the sample data below, the "BEST." format has been used to convert numeric values into a character. Notice that precision appears to be defined as the minimum number of places past the decimal that can be presented without sacrificing any accuracy.

Numeric Result	Character Result
1.60000	1.6
1.43000	1.43
0.60000	0.6
5.30000	5.3
1.10000	1.1
0.66738	0.66738
123.00000	123
7.00000	7

The important point here is that the "BEST." format applies it's evaluation of precision on a record-by-record basis. In contrast, attributing any numeric format to a variable will result in ALL values in the entire dataset conforming to the specified format. By extension, applying the "BEST." format to results of varying precision can be considered a first step in allowing SAS to help one decide what degree of actual precision to apply overall to a given set of values.

THE 'USUAL' PROGRAMMING SOLUTION (IF/THEN APPROACH)

Handling numeric precision and decimal alignment typically involves plenty of monotonous code meant to inclusively present a large collection of varying results. For example:

```
if param="Platelets" then do;
  if stat=1 then col=put(n,5.);
  else if stat=2 then col=put(mean,7.1);
  . . .
```

```
end;
*** REPEAT FOR EACH PARAMETER.;
```

There's nothing inherently wrong with this approach, as long as you don't mind lots of typing (and the ever-associated risk of mistyping). However, the remainder of this paper will describe how one can translate the information available from the above mentioned record-level "BEST." assessment into a dynamic set of numeric formats and put statements for presenting precise and aligned laboratory results without the production and maintenance of a large set of if-then statements.

PROGRAM EXAMPLES

EXAMPLE 1 – DYNAMIC ALIGNMENT AND PRECISION WITHIN A PATIENT SUMMARY LISTING

This example demonstrates applying dynamic decimal alignment and numeric precision to a set of lab results presented in a vertical listing.

The first step is to evaluate the minimum number of decimal places necessary at each result and pass this information into a temporary dataset variable.

```
*** CREATE TEMPORARY DATASET TO EVALUATE THE PRECISION OF EACH PARAMETER.;
proc sort data=labs out=temp(keep=testname labreslt);
   by testname;
   where labreslt>.z;
run;

*** EXTRACT THE INFORMATION AVAILABLE FROM THE 'BEST.' FORMAT REGARDING THE NUMBER OF DECIMAL
    PLACES FOR EACH RESULT, BY PARAMETER.;
data temp;
   length charval $200;
   set temp;
   by testname;

   charval=trim(left(put(compress(labreslt),best.)));

   *** FIND THE MAXIMUM NUMER OF PLACES PAST THE DECIMAL FOR EACH ANALYTE.;
   if int(labreslt)=labreslt then pastdec=0;
   else if indexc(charval,'.')>=0 then do;
      pastdec=length(trim(left(scan(charval,2,'.'))));
   end;
run;
```

Following this step, each record now contains a numeric variable (PASTDEC) that provides the minimum number of decimal places for the result field.

charval	testname	labreslt	pastdec
0.32	Hematocrit	0.32	2
0.34	Hematocrit	0.34	2
0.33	Hematocrit	0.33	2
0.29	Hematocrit	0.29	2
0.45	Hematocrit	0.45	2
0.49	Hematocrit	0.49	2
0.46	Hematocrit	0.46	2
0.43	Hematocrit	0.43	2
0.27	Hematocrit	0.27	2
0.31	Hematocrit	0.31	2
0.32	Hematocrit	0.32	2
0.4	Hematocrit	0.4	1

Notice that most parameters will have a varying number of decimal places presented in the dataset. In order to accurately represent ALL values for a particular parameter, it's necessary to select a record containing the maximum value of the PASTDEC variable for each parameter.

```
proc sort data=temp;
   by testname pastdec;
```

```
run;

*** FIND THE GREATEST NUMBER OF PLACES PAST THE DECIMAL PER ANALYTE.;
data temp;
   set temp;
   by testname pastdec;

   if last.testname;
run;
```

Once the maximum number of decimal places per analyte is obtained, one simply needs to pass this information into a character variable containing a representation of the appropriate numeric format as described in the algorithm presented above for achieving decimal alignment. In the code below, an integer format of “8.0” is selected as the starting point for alignment with all other results of varying precision.

```
data temp;
   length form_dec $200;
   set temp;

   form_dec=trim(left(put(8+(pastdec>0)+pastdec,2.)))||"."||trim(left(put(pastdec,1.)));
run;
```

The final step involves the creation of a set of macro variables representing the above created format field as well as the parameter name followed by the passing of this dynamic format into the actual laboratory results data to create a character field for final presentation.

```
%macro add_char;
   data _null_;
      length testnum macnum $200;
      set temp end=eof;

      *** CREATE A TEMPORARY RECORD-COUNT VARIABLE, C.;
      c+1;

      *** APPLY A GENERIC RECORD-COUNT LABEL TO EACH PARAMETER NAME.;
      testnum="test"||compress(c);
      call symput(testnum,trim(left(testname)));

      *** APPLY A GENERIC RECORD-COUNT LABEL TO PAIR THE NECESSARY FORMAT WITH THE
          ACCOMPANYING PARAMETER.;
      macnum ="_"||trim(left(put(c,8.)));
      call symput(macnum,trim(left(form_dec)));

      *** PASS THE TOTAL NUMBER OF PARAMETERS INTO A MACRO VARIABLE.;
      if eof then call symput('toptest',compress(c));
   run;

   data labs;
      length value_c $15;
      set labs;

      *** APPLY THE DYNAMIC CONDITIONAL FORMATTING TO THE ACTUAL NUMERIC LAB RESULTS.;
      %do T=1 %to &toptest;
         if testname="&&test&T" then do;
            value_c=put(labreslt,&&_&T);
         end;
      %end;
   run;
%mend add_char;

%add_char;
```

The final PROC REPORT output below demonstrates the difference between the aligned field and the original numeric result.

Parameter	Result	Result Aligned
Hematocrit	0.34	0.34
	0.38	0.38
	0.47	0.47
	0.3	0.30
	0.38	0.38
Platelets	153	153
	224	224
	237	237
	203	203
	373	373

EXAMPLE 2 – PRESENTATION OF A LABORATORY SUMMARY STATISTICS IN A TABLE

The general strategy of obtaining the number of places past the decimal for each parameter and then passing this information back into the originating data via macro variables in order to create a character field for presentation remains the same when applied to summary table presentation. However, with tables, a few subtle differences exist. The first is that allowing the actual number of decimal places as shown in the data to be reflected in the table may result in too extreme of a presentation. For example, presenting a mean and SD for an analyte containing a value presented to 4 places past the decimal will tend to overly crowd a table, and will likely provide way more precision than is actually required. As a result, it's good to have a general idea of the degree of overall greatest precision necessary (i.e. consult your study statistician), and then limit the dynamic aspect to a maximum of that particular precision. As an alternative to using the maximum number of decimal places as a starting point, the median number may be the best approach for summary tables.

```
data temp;
   length charval $200.;
   set labs;

   if labreslt>.z then do;
      charval=trim(left(put(compress(labreslt),best.)));
   end;

   *** FIND THE MAXIMUM NUMER OF PLACES PAST THE DECIMAL FOR EACH ANALYTE.;
   if int(labreslt)=labreslt then pastdec=0;
   else if indexc(charval,'.')>=0 then do;
      pastdec=length(trim(left(scan(charval,2,'.'))));
   end;

   *** ONLY TAKE THE FURTHEST TO 3 PLACES PAST THE DECIMAL.;
   if pastdec>3 then pastdec=3;
run;

proc sort data=temp;
   by testname pastdec;
run;

proc univariate data=temp noprint;
   by testname;
   var pastdec;
   output out=temp median=pastdec;
run;
```

An additional requirement for tables is that the dynamic format that will eventually be applied needs to also be flexible to both the overall format length as well as the number of places past the decimal. This is in order to account for differing numeric precision requirements for the different summary components to be presented (e.g., min/max vs SD with regard to the number of decimal places to present):

```
data temp;
   set temp;

   *** EXPRESS THE EVENTUAL FORMAT IN TERMS OF 2 VARABLES: THE TOTAL LENGTH AND THE
       NUMBER PAST THE DECIMAL.;
   form1=8+(pastdec>0)+pastdec;
   form2=pastdec;
run;

data temp;
   length labnum $200;
   set temp end=eof;

   *** CREATE A TEMPORARLY RECORD-COUNT VARIABLE.;
   count+1;

   *** APPLY A GENERIC RECORD-COUNT LABEL TO EACH PARAMETER NAME.;
   labnum="l_"||compress(count);
   call symput(labnum,trim(left(testname)));

   *** PASS THE TOTAL NUMBER OF PARAMETERS INTO A MACRO VARIABLE.;
   if eof then call symput('toptest',compress(count));
run;

data temp;
   length totfnum backnum $10;
   set temp end=eof;

   *** APPLY GENERIC RECORD-COUNT LABELS TO EACH ASPECT OF THE FORMAT LENGTH
       (TOTAL VS PAST DECIMAL).;
   totfnum ="f_"||compress(count);
   backnum ="b_"||compress(count);

   call symput(totfnum,trim(left(form1)));
   call symput(backnum,trim(left(form2)));
run;
```

The final step involves passing both macro variables into a few ‘put’ statements that are set up to be dynamic to the particular statistic being presented:

```
proc univariate data=labs noprint;
   by treatment testname visit;
   var labreslt;
   output out=labs n=n mean=mean std=std median=median min=min max=max q1=q1 q3=q3;
run;

data labs;
   length _col $50;
   set labs;

   array values (*) n meansd medianqs minmax;

   do stat=1 to 4;
      %do w=1 %to &toptest;
          if trim(left(testname))="&&l_&w" then do;
               if stat=1 then do;
                  %if %eval(&&b_&w=0) %then %do;
                     _col=put(values(stat),%eval(&&f_&w-1).);
                  %end;
                  %else %do;
```

```
                  _col=put(values(stat),%eval(&&f_&w-1-&&b_&w).);
               %end;
            end;
            else if stat=2 then do;
               _col=put(mean,%eval(&&f_&w+1).%eval(&&b_&w+1))||
                    "("||trim(left(put(std,%eval(&&f_&w+2).%eval(&&b_&w+2))))||")";
            end;
            else if stat=3 then do;
               _col=put(median,%eval(&&f_&w+1).%eval(&&b_&w+1))||
                    "("||trim(left(put(q1,%eval(&&f_&w+1).%eval(&&b_&w+1))))||
                    "-"||trim(left(put(q3,%eval(&&f_&w+1).%eval(&&b_&w+1))))||")";
            end;
            else if stat=4 then do;
               %if %eval(&&b_&w=0) %then %do;
                  _col=put(min,%eval(&&f_&w-1).%eval(&&b_&w))||
                       "-"||trim(left(put(max,%eval(&&f_&w-1).%eval(&&b_&w))));
               %end;
               %else %do;
                  _col=put(min,%eval(&&f_&w).%eval(&&b_&w))||
                       "-"||trim(left(put(max,%eval(&&f_&w).%eval(&&b_&w))));
               %end;
            end;
         end;
      %end;
   output;
end;
run;

proc sort data=labs;
   by testname visit stat;
run;

proc transpose data=labs out=labs(drop=_name_);
   by testname visit stat;
   var _col;
   id treatment;
run;
```

DISCLAIMER

Low intra-parameter value variation in addition to a reasonable consistency between the number of decimal places actually recorded in the dataset and the range of values typical of presentation for those parameters are the two cornerstones of the method I've presented. Value conversions from one unit to another can increase the variability in the number of decimal places within parameters to an unmanageable level, potentially confounding an attempt to control them dynamically in any meaningful way. In short, use of this or any 'dynamic' programming tool should always be carefully considered with a full awareness of the data at hand.

CONCLUSION

Creating character fields from varying numeric parameters with the intent of preserving decimal alignment and numeric precision doesn't have to come down to a set of hard-coded conditional clauses. SAS has both the ways and the means to allow for dynamic presentation of varying numeric results in both tables and listings. The methods presented above are also easily transferable to summaries involving vital signs, PK, or any other precision-varying data commonly group together.

ACKNOWLEDGMENTS

We would like to thank Cara, PJ, Nicholas, Cole, our friends in Biostatistical Services at ICON Clinical Research, and Tim Williams for their consistent encouragement and support.

CONTACT INFORMATION

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

Stephen Hunt
ICON Clinical Research
Suite 400, 555 Twin Dolphin Rd.
Redwood City, CA 94065
Email: hunts@iconus.com

Brian Fairfield-Carter
ICON Clinical Research
Suite 400, 555 Twin Dolphin Rd.
Redwood City, CA 94065
Email: fairfieldcarterb@iconus.com