Ten Ways to Improve the Efficiency of Clinical Statistical Programming

Amos Shu, INC Research, Inc., King of Prussia, PA

ABSTRACT
The competition among Clinical Research Organizations (CROs) is extremely fierce. Improving programming efficiency is a way to lower cost in the clinical statistical analysis area. This paper discusses ten ways to improve the programming efficiency: 1) avoid unnecessary sorts, 2) use sortseq = UCA (numeric_collation =on) option, 3) use criterion option in PROC COMPARE, 4) use compute block to add more than 10 footnotes, 5) use alias method of columns in PROC REPORT to create wide reports, 6) find the first and last day of the same month, 7) use picture statement in PROC FORMAT to align decimal places, 8) add alphabet letters to easily sort statistics, 9) use dummy datasets to make up tables that do not have data available, 10) create treatment "total".

INTRODUCTION
Many Clinical Trial Organizations (CROs) are looking to low cost in order to win the competition. Improving programming efficiencies is a way to save cost. This paper discusses ten common ways to improve programming efficiencies.

All programs presented in this paper were developed in Sever SAS 9.2 in the Windows environment.

1. AVOID UNECESSARY SORTS
To generate tables, we usually use PROC FREQ, PROC SQL, or PROC SUMMARY to summarize data first, then use PROC TRANSPOSE to have data for each treatment. The table looks like this:

<table>
<thead>
<tr>
<th>Treat1</th>
<th>Treat2</th>
<th>Treat3</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxx</td>
<td>yyy</td>
<td>zzz</td>
</tr>
</tbody>
</table>

To avoid unnecessary sorts before using PROC TRANSPOSE, the variable (i.e. treat) in the ID statement of PROC TRANSPOSE should always be the last one in the TABLES statement of PROC FREQ, or the SELECT statement of PROC SQL, or the CLASS statement of PROC SUMMARY. The order of variables (i.e. var1  var2  var3  cls1  cls2  cls3 ) in BY statement of PROC TRANSPOSE should be the same as ones in the beginning of the TABLES statement of PROC FREQ, or the SELECT statement of PROC SQL, or the CLASS statement of PROC SUMMARY.

The following programs illustrate the position of the "treat" variable:

```
PROC FREQ data = xyz;
  TABLES var1*var2*var3*treat/ out= outp nopercent;
RUN;
```

```
Proc SQL noprint;
  CREATE  table  outp as
  SELECT    var1, var2, var3, treat
  FROM xyz;
QUIT;
```

```
PROC SUMMARY data = xyz nway;
  VAR   count;
  CLASS cls1 cls2 cls3 treat;
  OUTPUT OUT = outp n=n;
RUN;
```

```
PROC TRANSPOSE data = outp out = outp_t;
  VAR   count;
  ID    treat;
  BY    var1 var2 var3;
    /* cls1 cls2 cls3 */
RUN;
```
2. USE SORTSEQ = UCA (NUMERIC_COLLATION=ON) OPTION
The NUMERIC_COLLATION option allows integers, expressed as text in a character string, to be ordered numerically. We can use this option to sort values such as ‘1’, ‘2’, ‘3’, etc numerically without creating a corresponding numeric variable just for sorting purposes. For example, in SDTM datasets generation, we need to sort a sponsor defined identifier variable such as MHSPID in MH (Medical History):

```
PROC SORT data = MH SORTSEQ = UCA (NUMERIC_COLLATION=ON);
   BY USUBJID MHSPID;
RUN;
```

3. USE CRITERION OPTION IN PROC COMPARE
Many companies do parallel programming to control the data quality. PROC COMPARE is the best tool to accurately compare data. Sometimes two comparing datasets cannot agree with each other even though they have the same number, like the following example:

```
<table>
<thead>
<tr>
<th>Visit Number</th>
<th>Base VISITNUM</th>
<th>Compare VISITNUM</th>
<th>Diff.</th>
<th>% Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>2062</td>
<td>2.0200</td>
<td>2.0200</td>
<td>-4.44E-16</td>
<td>-2.2E-14</td>
</tr>
<tr>
<td>2070</td>
<td>2.0200</td>
<td>2.0200</td>
<td>-4.44E-16</td>
<td>-2.2E-14</td>
</tr>
<tr>
<td>2077</td>
<td>2.0200</td>
<td>2.0200</td>
<td>-4.44E-16</td>
<td>-2.2E-14</td>
</tr>
<tr>
<td>2084</td>
<td>2.0200</td>
<td>2.0200</td>
<td>-4.44E-16</td>
<td>-2.2E-14</td>
</tr>
<tr>
<td>2091</td>
<td>2.0200</td>
<td>2.0200</td>
<td>-4.44E-16</td>
<td>-2.2E-14</td>
</tr>
</tbody>
</table>
```

When it is not worthwhile to spend time figuring out the reason or if the difference is deemed too small to be important, the criterion = option will be helpful. This option allows values to be compared only out of a certain number of decimal places. You can set a value in an option like this:

```
PROC COMPARE base = xyz comp = xyzqc CRITERION = 0.0001;
RUN;
```

4. USE COMPUTE BLOCK TO ADD MORE THAN 10 FOOTNOTES
The FOOTNOTE statement can only add 10 footnotes. If you have more than 10 footnotes, which is common in many clinical studies, the COMPUTE BLOCK in PROC REPORT will be useful. Here is an example:

```
PROC REPORT data = xyz;
  ......  
  COMPUTE after _page_
  LINE @1 ”[1] ......; 
  LINE @1 ”[2] ......; 
  LINE @1 ”[3] ......; 
  ...... 
  LINE @1 ”[15] ......; 
  ENDCOMP;
RUN;
```

5. USE ALIAS METHOD OF COLUMN IN PROC REPORT TO CREATE WIDE REPORTS
Occasionally we have to create a wide report that needs to fit within two or more regular letter sheets. For example, an exclusion listing may contain 35 criteria. The alias method of column will be useful to repeat some columns across all sheets. Though the ID statement is another way to do this, the alias method is more flexible. In the following example, var1a and var1b repeat column var1, and var2a and var2b repeat column var2 in the output.
PROC REPORT data = xyz;
COLUMNS page order order2  var1  var2  ("Rai Stage|---" a  b  c  d )
  var1 =var1a  var2 =var2a  ("Beta - 2|--" a2  b2  c2  d2)
  var1 =var1b  var2 =var2b  ("Maximum|--" a3  b3  c3  d3)
DEFINE page .......
DEFINE var1 .......
DEFINE var2 .......
DEFINE var1a .......
DEFINE var2a .......
DEFINE var1b .......
DEFINE var2b .......
RUN;

6. FIND FIRST and LAST DAY OF THE SAME MONTH
There are a lot of missing or partial missing dates in clinical trial data. To handle a missing day, we often assign the first or last day of the same month to the missing day. The easiest way to do this is the INTNX function with the alignment option. Here is an example for assigning the last day in the same month:

\[
\text{Lastdt} = \text{INTNX('month', missdt, 0, 'E');}
\]
To assign the first day in the same month, just change the alignment option ‘E’ to ‘B’ or leave it as blank because the beginning day is the default.

7. USE PICTURE STATEMENT IN PROC FORMAT TO ALIGN DECIMAL PLACES
To make the summary table pretty, we often like to align the decimal and parenthesis places in tables like the following:

<table>
<thead>
<tr>
<th></th>
<th>Placebo</th>
<th>Treat1</th>
<th>Treat2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Female</td>
<td>19 (1.7%)</td>
<td>2 (40.0%)</td>
<td>7 (1.8%)</td>
</tr>
<tr>
<td>Male</td>
<td>41 (68.3%)</td>
<td>3 (60.0%)</td>
<td>15 (68.2%)</td>
</tr>
<tr>
<td>Race</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>African American</td>
<td>9 (15.0%)</td>
<td>1 (20.0%)</td>
<td>6 (27.3%)</td>
</tr>
<tr>
<td>Caucasian</td>
<td>50 (83.3%)</td>
<td>4 (4.5%)</td>
<td>15 (68.2%)</td>
</tr>
</tbody>
</table>

An efficient way to do this is to create a picture format. Here is an example:

\[
\begin{align*}
\text{PROC FORMAT;} \\
\phantom{\text{PROC FORMAT;}} \text{PICTURE pctd } 0-1000 = '0009.9%') (prefix='(') \\
\phantom{\text{PROC FORMAT;}} \text{Other = '';}
\end{align*}
\]

\[
\text{DATA xyz;}
\text{SET xyz;}
\text{a =put(a, 3.)||put(round(100*(a/total),.1), pctd.);}
\]

8. ADD ALPHABET LETTERS TO EASILY SORT STATISTICS
Descriptive statistics such as n, mean, median, minimum, maximum, range, and standard deviation are the most frequently displayed items in summary tables. To display those items in a certain order, a very easy way is to add alphabetic letters in front of those items when generating them. Here is an example:
PROC SUMMARY data = XYZ nway;
VAR result;
CLASS a b c d treat;
OUTPUT OUT = xyz_s (drop= _FREQ_ _TYPE_)
    N=an mean=bMean std = cSD median = dMedian min=eMin max=fMax;
RUN;

PROC TRANSPOSE data = xyz_s out = xyz_t;
VAR an bMean cSD dMedian eMin fMax;
ID treat;
BY a b c d;
RUN;

Finally, a SUBSTR function or PROC FORMAT can be used to remove all alphabet letters you added and keep statistics in desired order like this:

<table>
<thead>
<tr>
<th>Placebo (N=45)</th>
<th>Treat1 (N=43)</th>
<th>Treat2 (N=43)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>45</td>
<td>43</td>
</tr>
<tr>
<td>Mean</td>
<td>0.031</td>
<td>0.030</td>
</tr>
<tr>
<td>SD</td>
<td>0.0435</td>
<td>0.0465</td>
</tr>
<tr>
<td>Median</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Min</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Max</td>
<td>0.10</td>
<td>0.10</td>
</tr>
</tbody>
</table>

9. **USE DUMMY DATASETS TO MAKE UP TABLES THAT DO NOT HAVE DATA AVAILABLE**

Some tables such as physical examination contains all medical findings such as 'Normal', 'Abnormal', and 'Not Done' for each organs regardless of if the actual data is available. They appear like the following table:

<table>
<thead>
<tr>
<th>Placebo (N=45)</th>
<th>Treat1 (N=43)</th>
<th>Treat2 (N=43)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heart</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Abnormal</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Not Done</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Lungs/Chest</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Abnormal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Not Done</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

In real practice, not all subjects have all findings for all organs. For example, there may be only 'Normal' Heart data, no 'Abnormal' and 'Not Done' data available for Heart. In this case, creating a dummy dataset will be very helpful to generate this type of tables.
DATA dummy;
   DO x = 1 to 9;
       DO y = 1 to 3;
       Output;
       END;
   END;
RUN;

Then associate x with each organ and associate y with the finding values - 'Normal', 'Abnormal', and 'Not Done'. Finally merge with raw data, the desired table will be easily generated.

10. CREATE TOTAL TREATMENT
Most studies require listing a total treatment that combines all treatment groups. An easy way to add the total treatment is to add a 'Total' treatment into the treatment variable in the beginning, so the rest of programming will include the 'Total' the same as other individual treatment groups. The following code is an example:

DATA xyz;
   SET XYZ ;
   OUTPUT;
   TREAT = 'Total';
   OUTPUT;
RUN;

CONCLUSION
Efficiency is an issue existing in every organization regardless of where the organization is located. If we can map out each minor step in the whole programming process and improve it, we can really make a difference. Each of the ten scenarios above is very minor and may usually be neglected by people. However, they can really help improve programming efficiency.

ACKNOWLEDGEMENTS
Special thanks to Thomas Kinghorn for his help.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Amos Shu
INC Research, Inc.
2200 Renaissance Blvd.
King of Prussia, PA 19406
Email: hshu@incresearch.com

TRADEMARK INFORMATION
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 trademarks of their respective companies.