ABSTRACT
One of the major tasks for pharmaceutical companies after clinical trial is to create production tables, listings, and graphs (TLGs) for regulatory submission. The production tables are created by unique SAS utility macros maintained by the company. Usually, a parallel quality control (QC) process is needed to ensure the accuracy of the content in the production tables. A programmer who performs the QC needs to write different SAS programs so that the results can be compared with those in the production tables. Usually the production tables have fixed layout, so creating QC results with similar layout allows QC programmer to conduct visual check line by line easily, thus improving the quality of QC and saving time. This paper uses a real example to illustrate how to apply two macros, ie, %GETFRQ and %GETSTAT, to accomplish such QC. Conceptual design and detail programs are provided related to these two macros.

BACKGROUND
Production tables are commonly used to meet the requirement for regulatory submission after clinical trials. Most pharmaceutical companies use unique SAS utility macros to create these tables. Once those tables are generated, a QC process is necessary to check the accuracy of the content in the tables against the production table and table shell requirements.

The general layout of a production table includes titles, column headers, row subtitles, table contents, footnotes. Table 1 (a production table) is one example. The QC programmer needs to make sure every part of the table is correct. First, for the titles and footnotes, because they do not involve detail calculation, it is easy to conduct visual checks. Second, the QC programmer needs to check if all sections are included as required by the table shell. If the table column headers have total case counts in each treatment group, the QC programmer needs to check those big N numbers. Mostly importantly, within each section, the QC programmer needs to check if all the row subtitles are correct and if all the numbers match between the production tables and those from the QC programs.

Using Table 1 as an example, it includes titles, footnotes, column headers, section row grand title, row subtitles, with column content reporting frequency count and incidence by treatment group for categorical variables, and N, mean, standard deviation (SD), median, quintile 1 (Q1) and quintile (Q3), and minimum (Min) and maximum (Max) for numeric variables. A common approach to conduct QC is to run "proc freq" for those categorical variables (i.e., sex, race) and run "proc means" or "proc univariate" for those numeric variables (i.e., age). The incidence listed in the parentheses is calculated. The QC programmer check the results in the SAS list file created by the QC programs against the numbers in the production tables. Unfortunately, the results generated by the QC programs if using this approach are often very scattered if there are many body content sections. As a result, the disadvantages of this traditional QC processes include 1) time consuming; 2) easy to miss one or more checking items; 3) programming repeatedly using "proc freq", "proc means" or "proc univariate" if need to QC different production tables. The following sections will discuss how to create QC macros that can generate results with similar layouts to those in the production tables so that the QC programmer can easily eyeball and compare the results and thus make the whole QC process more efficient and accurate.
Table 1. Baseline Demographics

Full Analysis Set

<table>
<thead>
<tr>
<th></th>
<th>Placebo (N=2)</th>
<th>Treatment (N=4)</th>
<th>All Subjects (N=6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex – n (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Male</td>
<td>1 (50)</td>
<td>3 (75)</td>
<td>4 (67)</td>
</tr>
<tr>
<td>Female</td>
<td>1 (50)</td>
<td>1 (25)</td>
<td>2 (33)</td>
</tr>
<tr>
<td>Race – n (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>White or Caucasian</td>
<td>2 (100)</td>
<td>4 (100)</td>
<td>6 (100)</td>
</tr>
<tr>
<td>Black or African American</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Hispanic or Latino</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Asian</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Japanese</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>American Indians or Alaska Native</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Native Hawaiian or Other Pacific Islander</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Aborigine</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Other</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Age (Years)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Mean</td>
<td>56.0</td>
<td>54.3</td>
<td>54.8</td>
</tr>
<tr>
<td>SD</td>
<td>1.41</td>
<td>7.93</td>
<td>6.24</td>
</tr>
<tr>
<td>Median</td>
<td>56.0</td>
<td>54.0</td>
<td>55.5</td>
</tr>
<tr>
<td>Q1, Q3</td>
<td>55, 57</td>
<td>49, 60</td>
<td>52, 57</td>
</tr>
<tr>
<td>Min, Max</td>
<td>55, 57</td>
<td>45, 64</td>
<td>45, 64</td>
</tr>
</tbody>
</table>

Note: Percentages are based on subjects randomized.

Program: /study/analysis/final/tables/program/t_base_dmg.sas
Output: t14-02_001_001_base_dmg_full.rtf (Date Generated: 09NOV2006: 9:11:04)
Source Data: /study/analysis/final/statdata/crt/demo, baseline, hxothr, vitals.sas7bdat

CONCEPTUAL DESIGN OF %GETFRQ AND %GETSTAT MACROS

QC macros such as %GETFRQ and %GETSTAT can be used to generate results with similar layouts to those in the production tables. Again using Table 1 as the example, there are primarily two types of sections in this table: one is the numeric variable reporting section and the other is the categorical variable reporting section. In order to create SAS results with similar layout to that in Table 1, two separate macros are needed to deal with numeric and categorical variables, respectively. A blank line needs to be created between any two consecutive sections. Then the results need to be combined from more than one section. The following macros need to be called upon: %GETFRQ (sex), %GETFRQ (race), and %GETSTAT (age). One caveat is that if there are more than two reporting sections, multiple calls of macros may potentially cause confusion between the current and previous call of macro. So, the intermediate data sets needed to be cleaned up before the next macro call. Finally, each section’s output results need to be concatenated into one final reporting data set. Using this data set, a QC report that has similar layout to those in the production table can be easily generated.

The followings are some functions of the proposed macros:

- Can handle different reporting variables from different data sets by treating the reporting variable and data set as parameters in the macros
- Can generate section row grand title, i.e., Sex – n (%), Race - n(%), Age (Years), which are not in the real data
- Can generate row subtitles, e.g., Male and Female, which have corresponding number of 0 or 1 for the SEXCD variable in data set
- (%GETFRQ) can generate frequency counts and calculate the incidence
- (%GETSTAT) can generate N, mean, SD, Median, Q1, Q3, Min, Max, and adjust their decimal points
- Can exclude some types of cases, e.g., only keep evalitt='Yes’ cases
- Can adjust the width of reporting columns
- Can report row subtitle with zero count for both treatment groups, and report both treatment and placebo group even one group has zero count
- Can control the order of sections and row subtitles
TECHNIQUES USED IN %GETFRQ AND %GETSTAT MACROS

FINAL DATA SET STRUCTURE
The first to consider is the structure of the data set created by the %GETFRQ and %GETSTAT macros. The final QC data set named as QC_FINRPT may contain five or six variables depending upon whether or not the grant total is included. Those variables are: column1 (for row subtitles), TRT_i (for placebo group), TRT_j (for treatment group), TRT_n (for grand total), and, two extra variables to control the section order and row subtitle order within each section, which were named SECTION and ORDER, respectively. The %GETFRQ and %GETSTAT assign different values to the TRT_i, TRT_j, and TRT_n variables. For categorical variables, the value will be counts (incidence). For the numeric variable, the value will be N, Mean, SD, Median, Q1 and Q3, or Min and Max. Because each call of %GETFRQ or %GETSTAT yields a data set with same variables and attributes, the proc append 'plus 'force' option can then be used to concatenate different data sets from different calls of the macros.

The detail codes are as follows:
```plaintext
proc append base=qc_finrpt data=&outdb force;
run;
```

CREATING ROW GRAND TITLE AND BLANK LINE
Similar techniques may be used to create row grand title and blank line after a section. Specifically, a macro parameter named SECTION_TITLE passes its value to the aforementioned variable named Column 1 and assign missing value to other variables TRT_i, TRT_j, TRT_n. It is saved as a record on top of other records for the same section. Using the same technique, a flag parameter SECTION_BLANK can be created to control whether or not an extra blank line is added after each section. In this case, missing values are assigned to each variable and the record is saved at the end of each section.

The detail codes are as follows:
```plaintext
*get grand row title for a section;
%if &section_title^= %then %do;
  data section_title;
  format column1 $200.;
  column1="&section_title";
  section=put(&section,4.0);
  output;
run;

data &outdb;
  set section_title &outdb ;
  order=_n_;
run;
%end;

%if &section_blank=Y %then %do;
*get blank line between two sections;
data blank;
  section=put(&section,4.0);
  output;
run;
data &outdb;
  set &outdb blank;
  order=_n_;
run;
%end;
```

CREATING ROW SUBTITLES
How to design a %GETFRQ macro to assign row subtitles in the QC data set and make sure the row subtitles include all reporting categories even if there is zero count in some categories. This is very challenging due to the following reasons:

- Different reporting variables may have different row subtitles
- Categorical reporting variable may have numeric or character values
- Need to generate row subtitle corresponding to the values in the original data set
- Need to control the order of row subtitles
Need to include the full list of row subtitles and its corresponding values in the variables TRT\textsubscript{i}, TRT\textsubscript{j}, and TRT\textsubscript{n}. For those with zero count for both treatment and placebo group, zero will be reported for the variables TRT\textsubscript{i}, TRT\textsubscript{j}, and TRT\textsubscript{n}.

Need to dynamically process all of the above steps. Some programmers try to achieve this in several data steps and they have to assign all the values of those five-six variables. In order to get full list of row subtitles, it is difficult to avoid hard coding without using the approach we are presenting here.

To achieve flexibility, integrity, and easier for assigning the row subtitle values into the %GETFRQ macro, we suggest using a process to create a format and then reverse it. Since categorical reporting variable may have numeric or character values, the created format should be numeric or character format. The left hand side of the format is the designing collected raw data values corresponding to row subtitles and the right hand side is the full category of the row subtitles. If some categories have zero count in the original data set, a designed value (e.g. zero) is assigned on the left hand side based upon table shell. After creating a format, the next steps are to create a data set based upon the format and then conduct an 'outside full join' to merge this data set with the original data set in order to get full reporting row subtitles. In addition, several SAS format procedures can be use to create reverse format to control the order of row subtitle. The format is saved into a data set, the values for the START and LABEL variables can be reversed to a data set and then make the reverse format. This reverse format can then be used to create the intermediate FRQORDER variable. After sorting the FRQORDER variable, it is used to assign final ORDER= _n_. The SECTION and ORDER variables will be used to control the order of multiple sections and orders of row subtitles within each section.

The detail codes are as follows:

```sas
*save format name as a macro variable in order to be used for dynamic process;
data _null_;  
myfmt=compress("&rowtitle_fmt",',.');  
call symput("myfmt",myfmt);  
run;

*use format to create a data set used for creating reverse format;  
proc format lib=work CNTLOUT=tempformat;  
select &myfmt;  
run;

*create reverse format used for control row subtitle order;  
data reversefmt;  
informat label $200.;  
format start end label $200. ;  
set tempformat(keep=start label rename=(start=nstart label=nlabel)) end=last;  
start=nlabel;  
end=nlabel;  
label=nstart;  
fmtname="r"!!left(compress("&myfmt","$."));
fmtname="$r"!!left(compress("&myfmt","$."))!!'.';
type='c';  
if last then call symput('rfmt',fmtname);
run;

*create reverse format to control order;  
proc format library=work cntlin=reversefmt MAXSELEN=200 MAXLABELN=200;  
run;

*use tempformat to create shell file to keep all the frequency categories;  
data tempformat;  
set tempformat(keep=label rename=(label=&frqvar));  
run;

* get analysis data set and change frqvar to char. Var.;  
data tot&indb;  
set &indb(rename=(&frqvar=o&frqvar));  
&db_cond ;  
*change frqvar to char variable;  
&frqvar=put(o&frqvar,&rowtitle_fmt);  
run;  
......  
*get full categories for row subtitle;  
proc sql;
```
create table shell as select distinct
  a.&trt,
  b.&frqvar
from total as a, tempformat as b
;
quit;

GENERATE FULL TREATMENT /PLACEBO COLUMNS
When a treatment or placebo group has zero observation, you still must get full treatment column reporting items. The parameter TRTSCHELL is used for this purpose and also for the purpose to create grand total column if needed. If grand total is requested, each observation needs to be saved again and a '99' is assigned to its treatment group variable. A full category treatment group data set will be created based upon the TRTSCHELL parameter. After merge this data set with the original data set with real treatment group counts, the problem that one treatment /placebo group has zero observation is overcome. The detail codes are included below. The codes also prepare for calculating the incidence by getting the total treatment counts and saving them as macro variables for the dynamic processes.

*create grand total cases;
%if %upcase(&indyn_tot)=Y %then %do;
  data tot&indb;
  set tot&indb;
  &trt=99;
  output;
  run;
%end;

*conservative strategies and take away duplicated cases;
proc sort data=tot&indb nodupkey out=tot;
  by usubjid &trt;
run;

*get real treatment counts;
proc freq data = tot noprint;
  tables &trt / list out = total;
run;

*get full treatment group categories;
%if &trtshell=" %then %do;
  data trtshell;
  format &trt 3.0;
  i=1;
  do while (scan("&trtshell",i," ")='' );
    &trt=input(put(scan("&trtshell",i," "),$3.),3.0);
    count=0;
    output;
    i+1;
  end;
  run;
%end;

*get full treatment categories and counts by merging with treatment group shell file;
data total;
  merge trtshell total;
  by &trt;
run;
%end;

*get each treatment total N counts and save as macro variables;
data _null_
  set total end=last;
  if &trt<=.;
  call symput('N'||compress(left(put(&trt,8.))),
    trim(left(put(count,8.))));
run;

REPORT SELECTED VALUES
Before the final statistics are generated, our macro has the ability to exclude cases by applying SAS’s condition syntax. For example, if one wants to report `evalitt='Y'` cases only, a `DB_COND` parameter can be used and is assigned values using `%str(if evalitt='Y')` to fulfill this task. See the following codes for details.

```sas
* apply data set condition to exclude cases;
data tot&indb;
    set &indb(rename=(&frqvar=o&frqvar));
    &db_cond;
run;
```

**CALCULATE COUNTS (INCIDENCE)**
Regarding calculating the “counts (incidence)” for the treatment/placebo groups, first, the frequency counts by treatment group and reporting categorical variable need to be calculated. Then, the file is merged with the shell file to get full categories for treatment group and reporting variable. This file is then merged with aforementioned file that contains total treatment group counts so that the incidence and final report phrase (counts (incidence)) can be calculated or generated. Finally, the data set is transposed and manipulated to achieve the same reporting layout as the production table.

The detail codes are as follows:

```sas
* get treatment group total counts as denominator to calculate incidence;
data total (keep = &trt total);
    set total;
    rename count = total;
run;

*get fully categories for treatment and frqvar;
proc sql;
    create table shell as select distinct
        a.&trt
    ,b.&frqvar
    from total as a, tempformat as b
    ;
quit;

*get frequency counts for each treatment group and reporting categorical variable;
proc freq data = tot&indb noprint;
    tables &trt*&frqvar / list missing out = &outdb;
run;

* get all categories for reporting categorical data;
proc sql undo_policy=none;
    create table &outdb as select distinct
        a.count
    ,b.&trt
    ,b.&frqvar
    from &outdb as a right join shell as b
    on a.&trt=b.&trt and a.&frqvar=b.&frqvar
    ;
quit;

* sort data in order to conduct merge;
proc sort data=&outdb;
    by &trt;
run;

proc sort data=total;
    by &trt;
run;

* merge with total to get denominator in order to calculate incidence;
data &outdb (keep = &trt &frqvar rpt_col);
    length rpt_col $&len pct &pctfmt;
merge &outdb total;
    by &trt;
    if count=. then count=0;;
    if total>0 then pct = 100 * count / total;
```

else pct=0;
   rpt_col= trim(left(put(count, 4.0))))||' ('||trim(left(pct,&pctfmt))) || ')';
run;

proc sort data = &outdb;
   by &frqvar;
run;

*transpose data structure to reporting layout structure;

proc transpose data = &outdb prefix = trt out = &outdb;
   var rpt_col;
   by &frqvar;
   id &trt;
run;

* get all final variables;
data &outdb(keep=section frqorder column1 trt);
   format column1 $200.;
   set &outdb;
   section=put(&section,4.0);
   column1=&frqvar;
   frqorder=put(column1,&rfmt);
   column1= " "!!left(compbl(column1));
run;

TWO SMALL TIPS
The follow two tips help make the macro more robust. The first is to clean up all intermediate data sets to prevent those data sets from interfering with the subsequent macro calls when there are multiple sections and when a data set in the next macro call has zero observation. Second, a programmer needs to sort the final data set by section and order to control the layout of the row subtitle and section when there are multiple row subtitles and sections.

FINAL STEPS
Upon implementing all of the above steps, a SAS data set, eg., called QC_FINRPT, is generated in the current work directory with the following variables- COLUMN1, TRT (subscribe from 1st number of &trtshell parameter), TRT1 (subscribe from 2nd number of &trtshell parameter), TRTn (subscribe from 3rd number of &trtshell parameter if user wants the grand total for the treatment group), SECTION, ORDER. After print out the final data set with these variables along with the titles and footnotes, a QC programmer can conduct visual comparison of these results against the production table.

The following are the codes for these final steps:
   proc append base=qc_finrpt data=&outdb force;
   run;

   proc datasets lib=work;
      delete &outdb tempformat total tot&indb;
   run;

   proc sort data=qc_finrpt;
      by section order;
   run;

THE MACRO FOR NUMERIC VARIABLES
Similar logic can be used to design and build the %GETSTAT macro. Most of the programming processes are same between the two macros. The biggest difference between them is that %GETSTAT provides numeric statistics, such as, N, Mean, SD, Median, ‘Q1, Q3’, and ‘Min, Max’, which can be fulfilled by the ‘proc univariate’. Minor differences involve the concatenation to create the ‘Q1, Q3’, and ‘Min, Max’ reporting phrase. Using the same final data set and assigning different section numbers will concatenate different sections without messing up the order of the reporting sections and row subtitle.

QC PROCESSES USING THE TWO MACROS: %GETFRQ AND %GETSTAT
The general process of using above two macros is as follows:

- Define the input data set libname, data set name and format library
- Conduct data manipulation if necessary.
- Conduct all macro calls based upon reporting sections (see detail programs below)
- Print out final output and conduct visual checks on titles, table body contents, and footnotes.
- Once the QC output is checked against those in the production tables, if no discrepancy is found, the QC documents will be filled out and the QC task is complete.

The following is a sample program using the %GETFRQ and %GETSTAT macros in conducting QC, which creates outputs with similar layout to that in Table 1.

```sas
options nofmterr ls=110 nocenter;
*run init.sas to assign production data set library, format library;
%inc "init.sas";

*create macro to handle strata conditions;
%macro run_rpt(strata_cond=,filenm=);
*get macro variables used titles and footnotes;
%maptab(filenm=&filenm);

*apply strata condition;
data base_dmg;
  set crt.baseline;
  &strata_cond;
run;

*assign n0, n7, n99 as global macro-which will be create in getfrq and getstat-we will use those big N numbers in the title of qc report;

%global n0 n7 n99;
%let n0=;
%let n7=;
%let n99=;

*get sexcd format;
proc format;
  value $ sex
    'F'="Female"
    'M'="Male"
  ;
```
*call for getfrq macro to get gender reporting section results;
%GETFRQ(indb=base_dmg, frqvar=sexcd, outdb=sexcd, section=1, section_title=%nrbquote(Sex
- n(%)), trtshell=%str(0 7 99), indyn_tot=y, trt=trtcd, pctfmt=3.0, rowtitle_fmt=%str($sex.), len=15);

*get racecd format;
proc format;
  value race
    01="White or Caucasian"
    02="Black or African American"
    03="Hispanic or Latino"
    04="Asian"
    05="Japanese"
    06="American Indian or Alaska Native"
    07="Native Hawaiian or Other Pacific Islander"
    08="Aborigine"
    88="Other"
  run;

*call getfrq again to get race reporting section results;
%GETFRQ(indb=base_dmg, frqvar=racecd, outdb=racecd, section=2, section_title=%nrbquote(Race - n(%)), trtshell=%str(0 7 99), indyn_tot=y, trt=trtcd, pctfmt=3.0, rowtitle_fmt=%str(race.), len=15);

*call for getstat to get age section results;
%GETSTAT(indb =base_dmg, statvar =age, outdb =age, db_cond=, section =3, trt=trtcd, indyn_tot=Y, trtshell=%str(0 7 99), section_title=%nrbquote(Age (year) - n(%)));

*get final qc data set and redefine format;
data qc;
  format c1 $50. c2 c3 c4 $10.;
  set qc_finrpt;
  id=n;
  c1=combl(column1);
  c2=compress(trt0,'');
  c3=compress(trt7,'');
  c4=compress(trt99,'');
run;

*print out final data set;
options missing='-';
proc print data=qc split="*";
  label c1=""
    c2="Placebo*(n=&n0)"
    c3="Treatment*(n=&n7)"
    c4="All Subjs*(n=&n99)"
  ;
  var c1-c4;
  title1 "&title1"
  title2 "&title2"
  title3 "Final listing of QC reporting results for &outname"
  title4 " Placebo=&n0 Treatment=&n7 All subjs=&n99"
  footnote1 "&fot1"
  footnote2 "Program: &sasdir/&program..sas"
  footnote3 "Output: &sasdir/&program..lst (Date Generated: &sysdate &systime)"
  footnote4 "source data: &crt/&source ";
run;
%mend run_rpt;
%run_rpt(filenm=t_base_dmg_full);

The following is the output in v_t_base_dmg.lst file, which can be used to visually compare the QC results with those in the production table.
### Table X. Baseline Demographics

**Full Analysis Set**  
Final listing of QC reporting results for t14-02_001_base_dmg_full  
Placebo=2 Treatment=4 All subjects=6

<table>
<thead>
<tr>
<th>Obs</th>
<th>Sex - n(%)</th>
<th>Placebo (n=2)</th>
<th>Treatment (n=4)</th>
<th>All Subjects (n=6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Female</td>
<td>1(50)</td>
<td>3(75)</td>
<td>4(67)</td>
</tr>
<tr>
<td>2</td>
<td>Male</td>
<td>1(50)</td>
<td>1(25)</td>
<td>2(33)</td>
</tr>
<tr>
<td>3</td>
<td>Race - n(%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>White or Caucasian</td>
<td>2(100)</td>
<td>4(100)</td>
<td>6(100)</td>
</tr>
<tr>
<td>5</td>
<td>Black or African American</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>6</td>
<td>Hispanic or Latino</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>7</td>
<td>Asian</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>8</td>
<td>Japanese</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>9</td>
<td>American Indian or Alaska Native</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>10</td>
<td>Native Hawaiian or Other Pacific Islander</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>11</td>
<td>Aborigine</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>12</td>
<td>Other</td>
<td>0(0)</td>
<td>0(0)</td>
<td>0(0)</td>
</tr>
<tr>
<td>13</td>
<td>Age (year) - n(%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>n</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>Mean</td>
<td>56</td>
<td>54.3</td>
<td>54.8</td>
</tr>
<tr>
<td>16</td>
<td>SD</td>
<td>1.41</td>
<td>7.93</td>
<td>6.24</td>
</tr>
<tr>
<td>17</td>
<td>Median</td>
<td>56</td>
<td>54</td>
<td>55.5</td>
</tr>
<tr>
<td>18</td>
<td>Q1, Q3</td>
<td>55.57</td>
<td>49,60</td>
<td>52.57</td>
</tr>
<tr>
<td>19</td>
<td>Min, Max</td>
<td>55.57</td>
<td>45,64</td>
<td>45,64</td>
</tr>
</tbody>
</table>

Note: Percentages are based on subjects randomized.

Program: /study/analysis/final/tables/validation/program/v_t_base_dmg.sas  
Output: /study/analysis/final/tables/validation/program/v_t_base_dmg.lst (Date Generated: 18NOV06 15:47)

**CONCLUSIONS**

The %GETFRQ and %GETSTAT are small macros. They are easy to maintain and flexible to apply. They are also time-saving and increase accuracy in conducting QC on production tables. As a result, it saves company’s manpower and money.

**ACKNOWLEDGMENTS:**

Many thanks to Sam Wang and Minying Royston for their many inputs for the paper and also thanks for them for mentoring and providing chances to use macros into Amgen productions. Many thanks for my team members, Daping Wang, for her traditional source codes, which gave me some ideas to create above two macros on this paper. Many thanks for my wife, Zhimei Liu, for her help in editing the paper. Many thanks for encouragements of Amgen project team leaders and Quintiles managers for writing this paper.

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.

**CONTACT INFORMATION**

Linfeng Xu  
Quintiles  
325 E. Hillcrest Drive Suite 200  
Thousand Oaks, CA 91360  
Work Phone: 805-557-7727  
E-mail:Linfeng.xu@quintiles.com