```
%macro ma_bign
( in_data = &_default_bign_in_data
, in_where = &_default_where
, out_data = BIGN
, out_var = denom

, trtvar = &_default_trtvar
, trtvar_preloadfmt = &_default_trtvar_preloadfmt
, define_total_groups = &_default_define_total_groups
, trigger_trtvar_num = no

, usubjid = &_default_usubjid

, subgroup =
, subgroup_preloadfmt=

, display_fmt = &_default_display_fmt
, new_display_fmt = __chead

, prcode_split = &_default_prcode_split
, escapechar =
, trigger_bign_in_colhead = YES
, trigger_split_bign = YES
, trigger_parentheses = &_default_trigger_parentheses

, BigNfmt =
, below_bign = &_default_below_bign

, trigger_dummy_for_zero = YES

, help = &_default_help
, debug = &_default_debug
) / store source des='V1.0.0.8' ;
%*************************************************************************
FILENAME:    MA_BIGN.SAS
DEVELOPER:   (b) (6)
PLATFORM:    SAS 9.1.3, 9.2 on PC
MACROS USED: mu_wordscan.sas
ASSUMPTIONS: the dataset going in (IN_DATA) is structured as a single row per
subject
         per treatment group
DESCRIPTION:
This macro:
  1. outputs a dataset called BIGN containing population counts categorized
according to
    SUBGROUPS and TREATMENTS
  2. creates macro variables identified as BIGN(_subgroup)_treatment
  3. creates a new format by adding the N=xxx to the label of an existing format
  4. creates macro variables identified as COLHEAD(_subgroup)_treatment which are
also the labels
```

```
    of the formats described above

USAGE NOTES:

IN_DATA = input data used to calculate BigN.  In the form LIB.DATA, where lib may
be ommitted if WORK
    (default = &_default_bign_in_data) (REQUIRED)
IN_WHERE = subsetting clause
    (default = &_default_bign_where) (OPTIONAL)
OUT_DATA = the output data set produced.
    (default=BIGN) (OPTIONAL)
OUT_VAR = the name of the count variable produced.
    (default=DENOM) (OPTIONAL)


TRTVAR = the treatment group variable. Used in the CLASS statement of the PROC
MEANS.
    (default = blank) (REQUIRED)

DEFINE_TOTAL_GROUPS = used to define (sub)total group(s), this parameter must be
defined as follows:
     list of treatment variable values = subtotal 1 ! list of treatment variable
values = subtotal 2 ! ...
     (default = &_default_define_total_groups) (OPTIONAL)


    For example, the values of TRTVAR are 'A' 'B' and 'C' and the analysis calls
for a subtotal of 'A' and
    'B' and an overall total of all three treatments.  Set the parameter as
    define_total_groups = 'A' 'B' = 'subtotal' ! 'A' 'B' 'C' = 'total'
        The (sub)total value(s) must be of the same type as the exisiting variable
    The (sub)totals must be separated by an exclamation point (!)
    The (sub)total values are then used in the TRTVAR_PRELOADFMT (see below)

TRIGGER_TRTVAR_NUM = a yes/no trigger used to maintain the numeric value of the
treatment variable.
This should only be used is MA_BIGN is run outside of the bigger ArtZ system as the
default behavior
of the rest of the ArtZ macros alway convert the treatment to character.

TRTVAR_PRELOADFMT = the format which assigns the order of the treatments across the
page.  If no
    format is listed, SAS will generate a warn-ing that PRELAODFMT will have no
effect,
    ie, treatment groups which are not already in the data will not be
created/dummied.
    Also, note that any (sub)total groups should be defined here.
    (default = &_default_trtvar_preloadfmt) (OPTIONAL - *but highly recommended)

    For example, as above, the values of TRTVAR are 'A' 'B' and 'C' and the
```

```
analysis calls
    for the columns to be displayed as
        'A', 'B', 'Subtotal of A&B', 'C', 'Total'
    set the format as
      value $_trt <<<--fmtname of user choice
        'A' = '1'
        'B' = '2'
        'subtotal' = '3'
        'C' = '4'
        'total' = '5'

        NOTE - 'subtotal' and 'total' were created with DEFINE_TOTAL_GROUPS.  User
can pick
        whatever makes sense.

USUBJID = the variable(s) used to uniquely identify a subject.
(default=&_default_usubjid) (REQUIRED)

SUBGROUP = subgroups variable(s). (default = blank) (OPTIONAL)
SUBGROUP_PRELOADFMT = if provided, the list of formats will be applied to the
subgroup(s) and a
    Cartesian product of subgroup(s) and treatment will be performed.
    Use with caution as the product may produce illogical combinations if more than
one
    subgroup is used.


DISPLAY_FMT = the character format that will be used as the basis of the new
display format.
    This is always a character format regardless of the type of the TRTVAR that
went
    into the analysis. Program assumes the format is in the WORK library.
    (default=&_default_display_fmt) (OPTIONAL - *but highly recommended)
   For example:
   value $_trtdsp <<<--fmtname of user choice
     '1' = 'Treatment A'
     '2' = 'Treatment B'
     '3' = 'Treatments A & B'
     '4' = 'Placebo'
     '5' = 'Total'

NEW_DISPLAY_FMT = the format that will be created from the DISPLAY_FMT (above) and
which will
    include the optional PRCODE_SPLIT to force the N=xxx to the next line, the
optional
    parenteses around the N=xxx and the optional BELOW_BIGN text.  This is helpful
if using the subgroup(s)/treatment
    as an ACROSS variable in proc report.
    (default = __chead) (OPTIONAL)
```

```
PRCODE_SPLIT = the split character embedded into new display format before the
N=xxx.  This will
    be the same split character used in the proc report or proc print options.
    (default = &_default_prcode_split) (leave blank for no split)

TRIGGER_BIGN_IN_COLHEAD = yes/no trigger to add the bign into the format and
colhead macro variables that are created.

TRIGGER_PARENTHESES = yes/no trigger for the parnetheses around the (N=xx) that
gets added to the format label.
    (Default=NO)

BIGNFMT = the format used on the BigN numbers.  Allows users to specify some other
format, such as comma8.  If missing, will
    default to 8.
    (Default=&_default_BigNfmt)

BELOW_BIGN = The text to display below the N=x in the format/macro variable.  For
example,
    Treatment A
       (N=x)
       n  (%)   <<<---- to add this to the header, set BELOW_BIGN = n  (%)
       (default = blank)

TRIGGER_DUMMY_FOR_ZERO = yes/no trigger to control whether or not to create dummy
headers
    when there is NO data available.  Useful during intial phases of a study when
no paitients
    qualify for a particual population.


HELP = set to Y(es) to output helpful hints to the log
    (Default = &_default_help)

DEBUG = set to Y(es) to save all intermediate datasets and to turn on mprint. Set
to NO to delete
    all intermediate datasets and to not turn on mprint (maintains original option
setting)
    (Default = &_default_debug)


**Versioning**
Version 1.0.0.1 - Release for PGR-09 validation and inclusion in GBL

********************************************************************************
************;
%*------------------------------------------------------------------------------
--*;
  %PUT ------------------------------------------------ ;
  %PUT INFO: (&SYSMACRONAME) ;
  %PUT INFO: Version 1.0 ;
  %PUT -START------------------------------------------- ;


%**** STEP 1 - PARAMETER CHECKS;

  %let abort = no;
  %mu_help_debug


  %*** Initialize global macro variable to contain a return code indicating the
outcome of the BIG macro;
  %** 0 = ran without error and without dummying all results;
  %** 1 = ran without error and with dummying all results;

  %global MA_BIGN_RC;


%***** WORKINFO ****;
%md_workinfo(debug=&debug)

proc sql noprint;
select setting into :option_obs
from dictionary.options
where upcase(optname) = 'OBS'
;
quit;
options obs=max nosyntaxcheck;


%***** set parentheses for format label;
%if %upcase(%substr(&trigger_parentheses, 1, 1)) = Y %then %do;
    %let right_paren = );
    %let left_paren = (;
%end;
%else %do;
    %let right_paren = ;
    %let left_paren = ;
%end;

%******* remove quotes from prcode if they exist;
%if "&prcode_split" = "" %then %do;
    %let prcode_split = '$';
%end;
data _null_;
    call symput('unquoted_prcode_split', compress("&prcode_split", "'"));
run;

%******* remove quotes from escapechar if they exist;
%if "&escapechar" = "" %then %do;
```

```
    proc sql noprint;
    select name from dictionary.macros
    where upcase(name) = '_DEFAULT_ESCAPECHAR'
    ;
    quit;
    %let exist_default_escapechar=&sqlobs;
    %put exist_default_escapechar = &exist_default_escapechar;
    %if &exist_default_escapechar=1 and "&_default_escapechar" ne "" %then %do;
        %let escapechar= &_default_escapechar;
        %put escapechar = "&escapechar";
    %end;
    %else %do;
        %put ALERT_I: ESCAPECHAR not defined.;
        %if &help = Y %then %do;
          %put ALERT_I:    If embedded into the format labels;
          %put ALERT_I:    for the treatment column headers, ;
          %put ALERT_I:    they will not be considered in the ;
          %put ALERT_I:    caluculation of how many lines the ;
          %put ALERT_I:    headers will take up in the report;
        %end;
    %end;

%end;
%if "&escapechar" ne "" %then %do;
    data _null_;
        call symput('unquoted_escapechar', compress("&escapechar", "'"));
    run;
    %put unquoted_escapechar = "&unquoted_escapechar";
%end;
%else %if "&escapechar" eq "" %then %do;
    %let unquoted_escapechar = &escapechar;
%end;


%**** check for missing trigger_bign_in_colhead *****;
%if &trigger_bign_in_colhead ne %str() %then %let trigger_bign_in_colhead =
%upcase(%substr(&trigger_bign_in_colhead,1,1));
%if &display_fmt ne %str() and &trigger_bign_in_colhead ne Y and
&trigger_bign_in_colhead ne N %then %do;
    %put ALERT_I:
----------------------------------------------------------------------------------
--;
    %put ALERT_I: TRIGGER_BIGN_IN_COLHEAD is not valid (&trigger_bign_in_colhead);
    %put ALERT_I:   MA_BIGN will set to YES;
    %put ALERT_I:
----------------------------------------------------------------------------------
--;
        %let trigger_bign_in_colhead = YES;
%end;
```

```
%*-------------------------------------------------------------------------------
--*;
%* DEFINE TOTAL GROUPS;
%if &define_total_groups eq %str( ) or %index(&define_total_groups, =) eq 0 %then
%do;
    %put %str(ALERT_I:
---------------------------------------------------------------------------------
--);
    %put %str(ALERT_I: DEFINE_TOTAL_GROUPS paramenter is missing or does not
specify a total.);
    %put %str(ALERT_I: No (sub)total groups will be calculated.);
    %let define_total_groups=;
    %if &help eq Y %then %do;
        %put %str(HELP: To define subtotal and/or total groups enter the definition
of the group as:);
        %put %str(HELP: list of treatments = subtotal1 ! list of treatments =
subtotal2 ! ....);
        %put %str(HELP: For example: TRTC is in the data with values of 'A' 'B' and
'C' and the analysis);
        %put %str(HELP: calls for a subtotal of 'A' and 'B' along with an overall
total of all three treatments.);
        %put %str(HELP: set DEFINE_TOTAL_GROUPS = 'A' 'B' = 'subtotal' ! 'A' 'B'
'C' = 'total');
        %put %str(HELP: The left side of the equals sign must be in the syntax of
the actual data and will be);
        %put %str(HELP: used in an IN statement, ie, TRTC in ('A' 'B')  (no
commas!).);
        %put %str(HELP: The right side of the equals sign will be the value of the
new group and must be of the same);
        %put %str(HELP: type as the existing treatment group variable.);
        %put %str(HELP: Separate each subtotal/total definition with the
exclamation point (!));
        %put %str(HELP: Make sure to include these new groups in the value
statement of the format that will be );
        %put %str(HELP: assigned to the TRTVAR_PRELOADFMT parameter.);
    %end;
    %put %str(ALERT_I:
---------------------------------------------------------------------------------
--);
%end;
%else %do;
    %mu_wordscan(string=&define_total_groups, root=total, numw=numtotals, delim=!)
    %do i = 1 %to &numtotals;
        %mu_wordscan(string=&&total&i, root=side&i._, numw=numside, delim=%str(=))
    %end;
%end;
```

```
%*-------------------------------------------------------------------------------
--*;
%* check for missing or invalid IN_DATA, TRTVAR and TRTVAR_PRELOADFMT SUBGROUP;

    %mu_check_req_parameters(    parameters_to_check = IN_DATA USUBJID TRTVAR
,help=No)
    %mu_check_data_and_var_exist(
     data_to_check = &in_data
    ,vars_to_check_in_all_data = &trtvar &usubjid &subgroup
    ,vars_to_check_in_respective_data =
    ,abort_if_does_not_exist = YES
    ,help=No
    )

    %put
---------------------------------------------------------------------------------
-----------;
    %put ALERT_I: Using &IN_DATA to calculate BIGN;
    %put
---------------------------------------------------------------------------------
-----------;

    %let dsid_in_data = %sysfunc(open(&in_data));
    %let trtvar_type = %sysfunc(vartype(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &trtvar))));
    %let trtvar_length = %sysfunc(varlen(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &trtvar))));


%*-------------------------------------------------------------------------------
--*;
%** check for TRTVAR_PRELOADFMT, DISPLAY_FMT and NEW_DISPLAY_FMT;

    %if &trtvar_preloadfmt eq %str( ) %then %do;
        %put
---------------------------------------------------------------------------------
-----------;
        %put ALERT_I: No format has been specified in the TRTVAR_PRELOADFMT
parameter.;
        %put ALERT_I: &TRTVAR will not be dummied/preloaded;
        %put
---------------------------------------------------------------------------------
-----------;
    %end;

    %else %do;
      data _null_;
        call symput("trtvar_preloadfmt", compress("&trtvar_preloadfmt", '.'));
      run;
    %end;

    %if &display_fmt ne %str() %then %do;
      data _null_;
        call symput("display_fmt", compress("&display_fmt", '.'));
      run;
    %end;
    %else %if &display_fmt eq %str() and &new_display_fmt ne %str() %then %do;
          %put ALERT_I: DISPLAY_FMT is missing.  COLHEAD macro variables and 
NEW_DISPLAY_FMT will not be generated. ;
          %let new_display_fmt = ;
    %end;

    %if &new_display_fmt ne %str() %then %do;
      data _null_;
        call symput("new_display_fmt", compress("&new_display_fmt",'.'));
      run;
    %end;

    %if &bignfmt = %str( ) %then %do;
      proc sql noprint;
        select name from dictionary.macros
        where upcase(name) = '_DEFAULT_BIGNFMT';
        %let exist_bignfmt =&sqlobs;
      quit;
      %if &exist_bignfmt = 1 %then %do;
        %let bignfmt = &_default_bignfmt;
      %end;
      %else %if &display_fmt ne %str( ) %then %do;
        %put 
----------------------------------------------------------------------------------
-----------;
        %put ALERT_I: No format has been specified in the BigNfmt parameter.;
        %put ALERT_I: Will default to 8.;
        %put 
----------------------------------------------------------------------------------
-----------;
          %let BigNfmt = 8;
      %end;
    %end;

    %if &bignfmt ne %str() %then %do;
      data _null_;
        call symput("bignfmt", compress("&bignfmt",'.'));
      run;
    %end;



%*--------------------------------------------------------------------------------
```

```
--*;
%** Parse subgroup variables;
      %mu_wordscan(string=&subgroup, root=byvar, numw=numbyvar, delim= %str( ))


%*--------------------------------------------------------------------------------
--*;
%** check if SUBGROUP_PRELOADFMT is provided and matches in number to the SUBGROUP
variables(s);
    %if "&subgroup_preloadfmt" ne "" %then %do;
      %mu_wordscan(string=&subgroup_preloadfmt, root=_byvarfmt, numw=numbyvarfmt,
delim= %str( ))


      %if &numbyvar ne &numbyvarfmt %then %do;
        %put
---------------------------------------------------------------------------------
-----------;
        %put ALERT_C: Number of formats listed in SUBGROUP_PRELOADFMT does not
match ;
        %put ALERT_C: number of variables listed in SUBGROUP;
        %put ALERT_C: Formats will not be applied and no preloading will be done;
        %put
---------------------------------------------------------------------------------
-----------;
        %let subgroup_preloadfmt = ;
      %end;

      %else %do i= 1 %to &numbyvarfmt;
        %let bvtype&i = %sysfunc(vartype(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &&byvar&i))));
        %let bvlen&i = %sysfunc(varlen(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &&byvar&i))));
        data _null_;
          call symput("byvarfmt&i", compress("&&_byvarfmt&i", '.'));
        run;

        %if %upcase(%substr(&debug,1,1)) = Y %then %do;
        %put byvar&i = &&byvar&i;
            %put bvtype&i = &&bvtype&i;
        %put bvlen&i = &&bvlen&i;
        %put byvarfmt&i = &&byvarfmt&i;
        %end;
      %end;
    %end;

    %** if subgroup_preloadfmt not specified, then check to see if the variables
are already formated and strip it off;
    %if "&subgroup_preloadfmt" = "" %then %do;
     %do i = 1 %to &numbyvar;
```

```
        %let bvtype&i = %sysfunc(vartype(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &&byvar&i))));
        %let bvfmt&i = %sysfunc(varfmt(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &&byvar&i))));
        %let bvlen&i = %sysfunc(varlen(&dsid_in_data,
%sysfunc(varnum(&dsid_in_data, &&byvar&i))));

        %if /*&&bvfmt&i = %str() and*/ &&bvtype&i = C %then %do;
            %let _byvarfmt&i = $&&bvlen&i..;
        %end;
        %else %if /*&&bvfmt&i = %str() and*/ &&bvtype&i = N %then %do;
            %let _byvarfmt&i = best8.;
        %end;
        %*if "&&bvfmt&i" ne "" %then %let _byvarfmt&i = &&bvfmt&i;

        data _null_;
          call symput("byvarfmt&i", compress("&&_byvarfmt&i", '.'));
        run;


            %if %upcase(%substr(&debug,1,1)) = Y %then %do;
            %put byvar&i = &&byvar&i;
                %put bvtype&i = &&bvtype&i;
            %put bvlen&i = &&bvlen&i;
            %put byvarfmt&i = &&byvarfmt&i;
            %end;

        %let numbyvarfmt = &i;
     %end;
    %end;



%*------------------------------------------------------------------------------------
--*;
%*** check for no obs before subset***;
        %if %sysfunc(attrn(&dsid_in_data, NOBS)) eq 0 %then %do;
          %put trigger_dummy_for_zero = %upcase(%substr(&trigger_dummy_for_zero, 1,
1));
          %if %upcase(%substr(&trigger_dummy_for_zero, 1, 1)) = N %then %do;
            %put
------------------------------------------------------------------------------------
-----------;
            %put ALERT_I: Dataset &in_data contains 0 observations.;
            %put ALERT_I: &SYSMACRONAME will exit;
            %let no_obs = 1;
            %let close_lib_dsin = %sysfunc(close(&dsid_in_data));
            %put
------------------------------------------------------------------------------------
-----------;
```

```
          %goto exit;
        %end;
        %else %if %upcase(%substr(&trigger_dummy_for_zero, 1, 1)) = Y %then %do;
          %put
-----------------------------------------------------------------------------------
----------;
          %put ALERT_I: Dataset &in_data contains 0 observations.;
          %put ALERT_I: &SYSMACRONAME will create dummy head counts;
          %let no_obs = 1;
          %put
-----------------------------------------------------------------------------------
----------;
        %end;
      %end;
      %else %let no_obs = 0;


%*---------------------------------------------------------------------------------
--*;
%*** Close the data for update***;
    %let close_lib_dsin = %sysfunc(close(&dsid_in_data));

%*---------------------------------------------------------------------------------
--*;


%*---------------------------------------------------------------------------------
--*;
%*** STEP 2 - GET THE DATA ***;

%if &no_obs = 0 %then %do;
    %* subset as needed and create a working copy of the data set;
        proc sort data=&in_data out=dsin(keep=&subgroup &usubjid &trtvar);
            by &subgroup &usubjid &trtvar;
            %if %sysevalf(%superq(in_where)=,boolean)=0 %then %do; where &in_where;
%end;
        run;

    %let dsid_dsin = %sysfunc(open(dsin));
        %if %sysfunc(attrn(&dsid_dsin, NOBS)) eq 0 %then %do;
          %if %upcase(%substr(&trigger_dummy_for_zero, 1, 1)) = N %then %do;
            %put
-----------------------------------------------------------------------------------
----------;
            %put ALERT_I: Dataset &in_data contains 0 observations after applying
subset.;
            %put ALERT_I: &SYSMACRONAME will exit;
            %let no_obs = 1;
            %let close_lib_dsin = %sysfunc(close(&dsid_in_data));
            %put
```

```
----------------------------------------------------------------------------------
-----------;
            %goto exit;
          %end;
          %else %if %upcase(%substr(&trigger_dummy_for_zero, 1, 1)) = Y %then %do;
            %put
----------------------------------------------------------------------------------
-----------;
            %put ALERT_I: Dataset &in_data contains 0 observations after applying
subset.;
            %put ALERT_I: &SYSMACRONAME will create dummy head counts;
            %let no_obs = 1;
            %put
----------------------------------------------------------------------------------
-----------;
            %goto no_obs;
          %end;
        %end;

%*--------------------------------------------------------------------------------
--*;
    %let close_dsin = %sysfunc(close(&dsid_dsin));



%*--------------------------------------------------------------------------------
--*;
%*** STEP 3 - CREATE SUBTOTALS AND TOTAL GROUPS AS REQUESTED ***;

    %if &define_total_groups ne %str( ) %then %do;
    ** START SUBTOTAL / TOTAL GROUPS;
        %if &trtvar_type eq C %then %do;
            * may need to redefine the length of the treatment variable - consider
the new treatment group names;
            data _null_;
              call symput('new_trtvar_length',
                trim(left(put(max(%do i = 1 %to &numtotals; length(&&side&i._2),
%end; &trtvar_length), 8.))));
            run;
        %end;

        %* cycle through each definition of subtotal/total and create new
observations for each subject in each group;
        %do i = 1 %to &numtotals;
            * keep one record per subject per group of treatments;
            proc sort data=dsin out=total&i(drop=&trtvar) nodupkey;
                by &subgroup &usubjid;
                where &trtvar in (&&side&i._1);
            run;
            * create the new treatment group;
            data total&i;
                %if &trtvar_type eq C %then %do;
                    length &trtvar $ &new_trtvar_length;
                %end;
                set total&i;
                retain &trtvar &&side&i._2;
            run;
        %end;

        *set the new treatment groups together with the original data;
        data dsin;
        %if &trtvar_type eq C %then %do;
          length &trtvar $ &new_trtvar_length;
        %end;
        set dsin %do i = 1 %to &numtotals; total&i %end; ;
        run;

        proc sort data=dsin;
            by &subgroup &usubjid &trtvar;
        run;
    *END OF SUBTOTAL / TOTAL GROUPS;
    %end;


%*---------------------------------------------------------------------------------------
--*;
% *** STEP 4 - CREATE THE DUMMY COUNTING VARIABLE, APPLY FORMATS, CHECK FOR MISSING
VALUES ***;
   %mu_wordscan(string=&usubjid, root=usubjid, numw=numusubjid, delim=%str() )
        data dsin;
            set dsin;
            by &subgroup &usubjid &trtvar;

        *create a dummy counter variable that PROC MEANS will use in the VAR
statement;
        __counter = 1;

        %if &trtvar_preloadfmt ne %str( ) %then %do;
            format &trtvar &trtvar_preloadfmt..;
        %end;
        %if &subgroup_preloadfmt ne %str() %then %do;
            %do i = 1 %to &numbyvar;
              format &&byvar&i &&byvarfmt&i...;
            %end;
        %end;

        if first.&usubjid1 then do;
        %do i = 1 %to &numbyvar;
            if missing(&&byvar&i) then do;
```

```
              put "ALERT_C: SUBGROUP variable %upcase(&&byvar&i) is missing for "
&usubjid=;
            end;
        %end;

            if missing(&trtvar) then do;
              put "ALERT_C: TRTVAR variable %upcase(&trtvar) is missing for "
&usubjid=;
            end;
        end;
        run;




%*----------------------------------------------------------------------------------
--*;
%*** STEP 5 - COUNT;
        %* Run the data through PROC MEANS - NOTE: if no formats are assigned to
TRTVAR, the CLASS variable,
            then SAS will issue a WAR NING stating that PRELAODFMT will have no
effect;
        %** PROC MEANS OPTIONS ***
        completetypes = Create all possible combinations of class variable values
        nway = Limit the output statistics to the observations with the highest
_TYPE_ value
        preloadfmt = specifies that all formats are preloaded for the class
variables
        mlf = enables PROC MEANS to use the primary and secondary format labels for
a given range or
              overlapping ranges to create subgroup combinations when a multilabel
format is assigned to
              a class variable -- keep this option in place even if not using
multilabels
              in the format definition as it forces the class variable to convert
to character
        ;
        * COUNT!;
        proc means data=dsin noprint missing completetypes nway;
            %if &subgroup ne %str( ) and &subgroup_preloadfmt eq %str() %then %do;
by &subgroup; %end;
            *note the use of MLF in the CLASS statement -
            *  this will ensure that the treatment variable will
            *  be character in the output dataset;
            class &trtvar  / %if &trtvar_preloadfmt ne %str() %then %do; preloadfmt
%end; mlf ;
            %if &subgroup_preloadfmt ne %str() %then %do;
            class &subgroup / %if &subgroup_preloadfmt ne %str() %then %do;
preloadfmt %end; ;
```

```
            %end;

            var __counter;
            output out=BIGN(drop=_type_ _freq_) n=&out_var ;
        run;

%*--------------------------------------------------------------------------------
--*;
%*** STEP 6 - CHECK FOR DUMMIED SUBGROUPS;
        *If preloading formats for subgroups then create a flag to indicate
        *   whether or not a subgroup combination is dummied or not.  This will
        *   allow user to control how the dummied group is to be displayed;
        ***;
        %if &subgroup_preloadfmt ne %str() %then %do;

            proc sort data=dsin out=non_dummied_dsin;
                by &subgroup;
            run;

            proc means data=non_dummied_dsin noprint missing completetypes nway;
              by &subgroup;
              class &trtvar / preloadfmt mlf;
              var __counter;
              output out=non_dummied_BIGN(drop=_type_ _freq_ n) n=n;
            run;

            proc sort data=bign;
              by &subgroup &trtvar;
            run;
            proc contents data=bign out=cont_bign(keep=name type length) noprint;
            run;
            proc sort data=non_dummied_bign;
              by &subgroup &trtvar;
            run;
            proc contents data=non_dummied_bign out=cont_non_dummied_bign(keep=name
type length) noprint;
            run;
            data BIGN;
              merge BIGN(in=a) non_dummied_bign(in=b);
              by &subgroup &trtvar;
              if a and not b and &out_var = 0 then do;
                BIGN_DUMMY_FLAG = 1;
                put "ALERT_I: BIGN_DUMMY_FLAG = 1";
                put "ALERT_I: No data exists in &in_data for this subgroup
(&subgroup) and this subset (&in_where): ";
                put "ALERT_I: " &trtvar= %do i = 1 %to &numbyvar; &&byvar&i= %end;
;
                %if &numbyvar gt 1 %then %do;
                  put "ALERT_I: PROGRAMMER - check that subgroup combinations make
sense and remove illogical ";
```

```
                  put "ALERT_I: PROGRAMMER - combinations from BIGN data set";
                  %if &help eq Y %then %do;
                    put "HELP: Illogical combinations occur when a Cartes-ian
product of the subgroups";
                    put "HELP: are created.  For example, if you are preloading the
formats for REGION";
                    put "HELP: and COUNTRY, you will see every combintion of the
two including those";
                    put "HELP: which do not make sense like North America/China.
";
                    put "HELP: Use SUBGROUP_PRELOADFMT with great caution when
using multiple subgroups";
                  %end;
                %end;
              end;
              else BIGN_DUMMY_FLAG = 0;

            run;

        %end;
        %else %do;
            data bign;
                set bign;
                retain BIGN_DUMMY_FLAG 0;
            run;
        %end;




%*----------------------------------------------------------------------------------
--*;
%* STEP 7 - CREATE BIGN_INDEX;
        %*create bign_index to hold the values of the class variables - this will
be the index of the BigN macros;
        %*translate any potential decimal points in the BY or CLASS variables into
an underscore;
            %*why have decimal points? in case a table calls for subtotal(s) that
appears between
              the existing defined treatments the new classes may be defined in
numeric order by using decimals;
        data BIGN;
            set BIGN;

            length  %if &new_display_fmt ne %str( ) %then %do; start %end;
                    bign_index $ 200
                    ;
            %* prep for new display format;
            %if &new_display_fmt ne %str( ) %then %do;
```

```
                %**start will be used in creating the new_display_fmt;
                start = &trtvar;
            %end;

            %do i = 1 %to &numbyvar;
                *convert or rename SUBGROUP variable(s);
                %*---convert numeric SUBGROUP variables to character to allow for
construction of macro variables;
                %if &&bvtype&i = N %then %do;
                    length __&&byvar&i $ 32;
                    if not missing(&&byvar&i) then do;
                      %if &&byvarfmt&i ne %str() %then %do;
                       __&&byvar&i = trim(left(put(&&byvar&i, &&byvarfmt&i...)));
                      %end;
                      %else %if &&byvarfmt&i eq %str() %then %do;
                       __&&byvar&i = trim(left(put(&&byvar&i, best32.)));
                      %end;
                    end;
                %end;
                %else %do;
                %*---or, if already character, for consistency create a new
character variable ;
                    if not missing(&&byvar&i) then do;
                      %if &&byvarfmt&i ne %str() %then %do;
                        __&&byvar&i = trim(left(put(&&byvar&i, &&byvarfmt&i...)));
                      %end;

                      %if &&byvarfmt&i eq %str() %then %do;
                        __&&byvar&i = &&byvar&i;
                      %end;
                    end;
                %end;

                if index(__&&byvar&i, '.') gt 0 then do;
                  __&&byvar&i = translate(__&&byvar&i, '_', '.');
                end;

            %end;


            ** translate the decimal points into underscores because macro
variables cannot contain decimal points;
                if index(&trtvar, '.') gt 0 then do;
                    &trtvar = translate(&trtvar, '_', '.');
                end;

            %** create the bign_index using the values of the SUBGROUPS (if they
exist)
                    and TRTVAR variables separated by underscores -- this will be
used
```

```
                    in constructing the macro variables;
                bign_index = compress(
                    %if &numbyvar ge 1 %then
                        %do i = 1 %to &numbyvar;
                            "_" ||
                            __&&byvar&i

                            ||
                        %end;
                    "_"||&trtvar );

                if length(compress(bign_index)) gt 24 then do;
                    put "ALERT_C: " bign_index;
                    put "ALERT_C: is too large to use as the name of a macro
variable.";
                    put "ALERT_C: use formats to shorten length of components";
                end;


                run;



%*-------------------------------------------------------------------------------
--*;
%* STEP 8 - CREATE MACRO VARIABLES FOR BIGN ;
     %*create GLOBAL macro variables to hold the BigN counts - since the local
macro variable
        table is not empty, a call symput would create these as LOCAL.  Must use
the GLOBAL
        assignment before using call symput;

        %* the macro variables are not simply a sequence of BigN1-BigNX, the index
is
           descriptive since it contains the actual values of the decoded
           SUBGROUP(S) and TRTVAR ;
        proc sql noprint;
            create table _null_ as
            select * from BIGN;
            %let numBigN = &sqlobs;
            select bign_index into :bni1-:bni&numBigN
            from BIGN
            ;
        quit;

        %* explicit GLOBAL statement ;
        %do i = 1 %to &numBigN;
            %global bign&&bni&i;
        %end;
        * assign counts into GLOBAL macro variables;
        data _null_;
            set BIGN;
            if bign_index ne '' then
              call symput(compress("BIGN" || bign_index) , trim(left(put(&out_var,
8.))));
        run;


%*-------------------------------------------------------------------------------------
--*;
%* STEP 9 - CREATE THE NEW DISPLAY FORMAT AND THE MACRO VARIABLES FROM THE LABELS
OF THE FORMAT;

    %if &display_fmt ne %str( ) %then %do;
    * GENERATE NEW DISPLAY FORMAT;

        %*use existing display format to build a new format to be used as table
column
          headers with bigN counts integrated into the text ;


        %**get the format from the WORK catalog ;
        proc format cntlout=_fmt_ library=WORK ;
            select &display_fmt ;
        run;
        proc sort data=_fmt_;
            by start;
        run;
        proc contents data=_fmt_ noprint out=_fmt_cont;
        run;
        %**length of longest start ;
        proc sql noprint;
            select max(length, 100) into :start_length
            from _fmt_cont
            where upcase(name) = 'START'
            ;
        quit;


        %* check that all values in the data have ben accounted for in the format;
        proc sort data=bign out=check(keep=&trtvar) nodupkey;
         by &trtvar;
        run;
        data _null_;
        length &trtvar $ &start_length;
        merge check(in=a) _fmt_(in=b rename=(start=&trtvar));
        by &trtvar;
        if a and not b then do;
        put "ALER" "T_R: ---- &sysmacroname ----------------------";
        put "ALER" "T_R: %upcase(&trtvar) value [" &trtvar "] unaccounted for in
DISPLAY_FMT [%upcase(&display_fmt.).]";
        put "ALER" "T_R:   Neither NEW_DISPLAY_FMT [$%upcase(&new_display_fmt.).]";
        put "ALER" "T_R:   nor COLHEAD_XXX macro variables will be ";
        put "ALER" "T_R:   properly generated. ";
        end;
        run;


        %**length of longest label ;
        proc sql noprint;
            select max(200, length) into :maxlen
            from _fmt_cont
            where upcase(name) = 'LABEL'
            ;
        quit;

        %**build new format ;
        proc sort data=BIGN;
            by start;
        run;

        data _fmt2_(drop=start rename=(bign_index = start));
            length start end label $ &maxlen;
            merge BIGN(in=a) _fmt_(in=b drop=end);
            by start;
            end = bign_index;
            %if &new_display_fmt ne %str( ) %then %do;
              fmtname = "&new_display_fmt.";
            %end;
            %else %if &new_display_fmt eq %str( ) %then %do;
              fmtname = "$_temp_";
            %end;
            default = &maxlen;
            if not a and b then dummy=1;
            if bign_index ne '';
        run;

        data _fmt2_;
            set _fmt2_;
                if label='' then label = 'not defined';
            if not dummy then
            label = trim(left(label))
            %if %upcase(%substr(&trigger_bign_in_colhead,1,1)) = Y %then %do;
              %if %upcase(%substr(&trigger_split_bign, 1,1)) = Y %then %do; ||
compress("&unquoted_prcode_split") %end;
              || " &left_paren.N=" || trim(left(put(input(symget('BIGN' ||
compress(start)), 8.), &bignfmt..) )) || "&right_paren"
            %end;
            %if "&below_bign" ne "" %then %do; ||
compress("&unquoted_prcode_split") || trim(left("&below_bign")) %end;
            ;
            else if dummy then
            label = trim(left(label))
            %if %upcase(%substr(&trigger_bign_in_colhead,1,1)) = Y %then %do;
              %if %upcase(%substr(&trigger_split_bign, 1,1)) = Y %then %do; ||
compress("&unquoted_prcode_split") %end;
              || " &left_paren.N=?&right_paren"
            %end;
            %if "&below_bign" ne "" %then %do; ||
compress("&unquoted_prcode_split") || trim(left("&below_bign")) %end;
            ;
        run;

        %**add new format to catalog;
        proc format cntlin=_fmt2_ ;
        quit;

        %**create macro vars for the labels;
        %* explicit GLOBAL statement ;
          %do i = 1 %to &numBigN;
            %global colhead&&bni&i;
          %end;

        data _null_;
            length colhead $ &maxlen.;
            set bign;
            %if &new_display_fmt ne %str( ) %then %do;
              colhead = trim(left(put(bign_index, $&new_display_fmt..))) ;
            %end;
            %else %if &new_display_fmt eq %str( ) %then %do;
              colhead = trim(left(put(bign_index, $_temp_.))) ;
            %end;
              call symput("COLHEAD"||compress(bign_index), colhead);

        run;

        data BIGN;
            set BIGN;
            drop
            start
            %do i = 1 %to &numbyvar;
              __&&byvar&i
            %end;
            ;
            colhead = put(bign_index, $&new_display_fmt..);
        run;

        proc sort data=bign;
          by &subgroup &trtvar;
        run;


%end;


%* STEP 9b - if trigger_trtvar_num then convert trtvar to numeric;
%if &trtvar_type = N %then %do;
        %if &trigger_trtvar_num ne %str() %then %do;
          %let trigger_trtvar_num = %upcase(%substr(&trigger_trtvar_num, 1, 1));
          %if &trigger_trtvar_num ne Y and &trigger_trtvar_num ne N %then %do;
             %put ALERT_I: TRIGGER_TRTVAR_NUM is invalid.  Macro will default to
NO;
             %let trigger_trtvar_num = N;
          %end;
        %end;
        %else %if &trigger_trtvar_num eq %str() %then %do;
             %put ALERT_I: TRIGGER_TRTVAR_NUM is missing.  Macro will default to
NO;
             %let trigger_trtvar_num = N;
        %end;

        %if &trigger_trtvar_num = Y %then %do;
                data BIGN;
                  set BIGN(rename=(&trtvar = c_&trtvar));
                  &trtvar = input(c_&trtvar, 8.);
                run;
        %end;


%end;

%* STEP 10 - END OF PROCESS TASKS;
%if %upcase(&out_data) ne BIGN %then %do;
data &out_data;
set BIGN;
run;
%end;
%else %if &out_data = %str() %then %do;
%let out_data = BIGN;
%end;

%md_clean_and_reset(
          debug      = &debug
        ,_workdata   = %str(&WORK_DATASETS_DATA &out_data)
        ,_workview   = %str(&WORK_DATASETS_VIEW)
        ,resetmprint = &mprint_setting
        );
```

```
%* end of processing when there are observations;
%let MA_BIGN_RC = 0;
%end;


%**  if there are no observations in the data, then create macro variable for dummy
column headers;
%no_obs:
%if &no_obs = 1 %then %do;

  proc format cntlout=display;
   select
     %if &display_fmt ne %str( ) %then %do;
    &display_fmt;
     %end;
     %else %if &display_fmt eq %str( ) %then %do;
    %if &trtvar_preloadfmt ne %str( ) %then %do;
    &trtvar_preloadfmt;
    %end;
     %end;
  run;
  data _null_;
    length label $ 200;
    set display(keep=start label) end=eof;
    if eof then call symput("NUMBIGN", trim(left(put(_n_, 8.))));
  run;

  %do i = 1 %to &numbign;
    %global bign&i colhead&i colhead_&i bni&i;
  %end;
  data _null_;
    length label $ 200;
    set display(keep=start label) end=eof;
    label = trim(left(label)) %if "&unquoted_prcode_split" ne "" %then %do; ||
"&unquoted_prcode_split" %end;
        || "&left_paren.N=0&right_paren." ;
    call symput("BIGN" ||trim(left(put(start, 8.))), '0');
    call symput("COLHEAD"||trim(left(put(start, 8.))), label);
    call symput("COLHEAD_"||trim(left(put(start, 8.))), label);
    call symput("BNI"||trim(left(put(start, 8.))), trim(left(put(start, 8.))));

  run;

  %let MA_BIGN_RC = 1;
  %put ALERT_I: data set BIGN not created;

%end;


proc sql noprint;
```

```
select name from dictionary.macros
where upcase(name) like 'COLHEAD%';
%let _colhead_exist = &sqlobs;
quit;

%if &_colhead_exist gt 0 %then %do;
  %global G_TrtColHeadLines;
  data _null_;
    %do i = 1 %to &numbign;
    length x&i $ 200;
    x&i= trim(left("&&&&colhead&&bni&i"));
    lines&i = count(x&i, "&unquoted_prcode_split")+1;
    %if "&unquoted_escapechar" ne "" %then %do;
     * account for all RTF codes which force a line feed;
    lines_ods&i = count(x&i, "&unquoted_escapechar.n") + count(x&i, "\line") +
count(x&i, "\par");
    lines&i = lines&i + lines_ods&i;
    %end;

    %end;

    max= max(%do i = 1 %to &numbign; lines&i, %end; 1);
    call symput("G_TrtColHeadLines",trim(left(put(max,best.))));
  run;
  %put G_TrtColHeadLines = &G_trtcolheadlines;
%end;

options &mprint_setting;
options obs=&option_obs syntaxcheck;


%put
-----------------------------------------------------------------------------------
-----------;
%put >>> BIGN &subgroup &trtvar;
%do i = 1 %to &numBigN;
    %put >>> BIGN&&bni&i = &&&&BIGN&&bni&i;
    %if &_colhead_exist gt 0 %then %do; %put >>> COLHEAD&&bni&i =
&&&&colhead&&bni&i; %end;
%end;
%put
-----------------------------------------------------------------------------------
-----------;


  %PUT --------------------------------------------- ;
  %PUT INFO: MA_BIGN_RC = &MA_BIGN_RC;
  %PUT INFO: (&SYSMACRONAME) ;
  %PUT INFO: Version 1.0 ;
  %PUT -END--------------------------------------------- ;
```

```
%exit:
    proc sql noprint;
    select name from dictionary.macros
            where upcase(name) = 'DSID_IN_DATA'
        ;
        %let exist1=&sqlobs;
    select name from dictionary.macros
        where upcase(name) = 'DSID_DSIN'
        ;
        %let exist2=&sqlobs;
    quit;
    %if &exist1 %then %let close_lib_dsin=%sysfunc(close(&dsid_in_data));
    %if &exist2 %then %let close_dsin=%sysfunc(close(&dsid_dsin));
    %if &abort = yes %then %do;
        data _null_;
            abort;
        run;
    %end;

%mend ma_bign;
```