A SAS® Macro for Simple Statistical Summary
Michael Gunshenan, GlaxoSmithKline, King of Prussia, PA

ABSTRACT
This is a presentation of a macro that generates the statistical summary of a variety of derived data. It requires four keyword parameters: incoming data set, variable(s) upon which to calculate statistics, the numeric variable by which they will be ordered across the page, and a format to convert that variable into the character text to be displayed. It will determine for you which variables are numeric and which are character and apply the appropriate procedure (means / freq) to generate the statistics required.

Optional keywords exist:
- two keyword variables, one numeric and one the format to convert it to character can be used to calculate the statistics by, e.g. visit number and format.
- a keyword variable to calculate by a subgroup that shows output in the form of a fraction with percentages, the denominators are the numbers of distinct patients within the groups displayed across the page.
- a keyword variable to total the columns across the page.
- a keyword variable to controlling the split variable in the proc report

The table was not designed for customization or alteration via the macro parameters because the output has already been designed to match the requirements of the CDISC and ICH.

INTRODUCTION
The purpose of this macro is to standardize report writing, and gain efficiency by using previously prepared and validated code. The macro can be applied to a variety of data: demographic data, study drug exposure data, ECG and inclusion/exclusion data.

Assumptions for this macro are
1. the variable depicted across the page does not have a value of 0, since missing will be set to zero for this variable.
2. the variable depicted across the page has both a numeric and character representation (e.g., a treatment group text and ordering number).
3. other requirements like headers, line and page size options, and minor data preparation prior to the macro call are done.

Keyword parameters for this macro are the following:
1. dsin = incoming dataset
2. acrsvar = numeric variable depicted in the columns across the page
3. avfmt = format - textual representation of &acrsvar
4. varlist = list of variables for statistical calculations
5. byvar = numeric variable by group for statistics
6. byfmt = format - textual representation of &byvar
7. fract = for fraction / statistics on a subset of data
8. total = Y to display a total column
9. spltchr = split value for proc report

SPREADSHEET TO SAS DATASET
MACROVARIABLES FOR COLUMN HEADERS AND DATA HANDLING
The variable depicted in the columns across the page (&acrsvar) will have a header of the variable's value and the count of records for that value. If a total column for all &acrsvar variables is desired, a quick data step is used to duplicate all records and attach a "TOTAL" value to &acrsvar for those duplicates. The next step is getting the total count of records for each column, or &acrsvar value. We create the following macrovariables:
1. &Nx = textual value of each &acrsvar where x is the numeric value of &acrsvar.
2. &NUMn = numeric representation of the value that is sequential for do loop purposes where n is 1 to max.
3. &Cn = count of all records within that value and the total number of distinct values for &acrsvar.
4. &AVMAX = total number of distinct &acrsvar values.

Below is the portion of the macro these variables are created:

```
proc freq data=temp NOPRINT;
  table &acrsvar/ missing;
  out=avcnts(keep=&acrsvar count);
run;

data _null_; set avcnts END=FINAL;
  n+1;
  call symput('N'||left(put(&acrsvar, 8.)), put(&acrsvar, &avfmt..)); * Name *;
  call symput('NUM'||left(put(n, 8.)), trim(&acrsvar));   * Numeric rep *;
  call symput('C'||left(put(n, 8.)), trim(count)); * Count by &acrsvar*;
  if (FINAL) then call symput('AVMAX', put(n, 8.) );
run;
```

We then immediately use these macrovariables in a %DO loop to create macrovariables for the column headers with macrovariable names that match the &acrsvar value. Options missing=0 is used, so all missings are set to zero. Code is added to overwrite the value of 0 with "missing" for the column header:

```
%do e=1 %to &avmax;
  %let t&e = %trim(%nrstr(&n)%eval(&&&num&e));
  %if &&&num&e = 0 %then
    %do;
      %let trt&e = "MISSING~(N=%trim(&&&c&e))";
    %end;
  %else %if &&&num&e ^= 0 %then
    %do;
      %put 't&e' = &&&t&e;
      %let trt&e = "%trim(&&&t&e)~(N=%trim(&&&c&e))";
    %end;
  %end;
```

DETERMINING & SEPARATING TYPES OF VARIABLES FOR STATISTICAL CALCULATIONS
The next step is to take the list of variables for which statistics must be calculated, and subset it into separate lists for character and numeric variables.

Using _character_ and _numeric_ within array statements to grab all variables within those types does this. The macro then writes the variables out as macrovariables with values set to ‘C’ for character and ‘N’ for numeric.
character or 'N' for numeric (e.g., sex becomes a macrovariable where &sex = 'C' because it is a character variable).

```
data _null_;  
set temp;  
array chr {*} _character_;  
array nmr {*} _numeric_;  
length name $ 8;  
do i=1 to dim(chr);  
call vname(chr{i}, name) ;  
call symput(name, 'C');  
end;  
do j=1 to dim(nmr);  
call vname(nmr{j}, name) ;  
call symput(name, 'N');  
end;  
stop;  
run;```

Scan is used to set the first word from the varlist as the macrovariable "&var". If the first variable in &varlist equaled sex, then &&&var would equate to &sex. The macrovariable from the above vname statement is the final output from calling that macrovariable. Using a %if/%%Then statement we set &varc to &var, or as in the case of numeric variables &var is set to &var:

```
%let h = 1;  
%let var  =  %scan(&varlist, &h) ;  
%if &&&var = C %then  
   %let varc= &var ;  
%else  
   %let varc= &varc &var ;  
%end;  
%let varn = &varn &var ;
```

Prior to the code above, &varc and &varn must be created as empty macrovariables to avoid errors. Perform the same check on the subsequent words until, through incrementing &h, scan returns blank. With ever subsequent word, append the new word to the previous value for &varc or &varn, whichever applies:

```
%do %while (%scan(&varlist, &h) ne %str( ) );  
   %let h = %eval(&h+1) ;  
   %if %scan(&varlist, &h) ne %str( ) %then  
      %do;  
         %let var  =  %scan(&varlist, &h) ;  
         %if %eval(&&&var = C) %then  
            %do;  
               %let varc= &varc &var ;  
            %end;  
         %else  
            %do;  
               %let var  =  %scan(&varlist, &h) ;  
               %if %eval(&&&var = N) %then  
                  %do;  
                     %let varn= &varn &var ;  
                  %end;  
            %end;  
      %end;  
%end;  
```

**Statistical Calculations**

At this point, the statistical calculations begin, running character variables through proc freq and the numeric variables through proc means. A similar incrementing scan of &varc and &varn is used to send one variable through at a time. Additionally, a proc sql using sum(count) by &acrsvar is used to get the count of responses for each character variable.

Following those procs, ordering variables are created, data is transposed by &acrsvar and the textual representation of &acrsvar as well as the new ordering variables.

For character variables, percentages are calculated based on the &Cn mentioned above, counts and percentages are converted to text fields, and where counts are missing, percentages are set to "( 0%)".

For numeric variables, statistics are converted to text and the following decimal places are locked (based on incoming data being integers) N = 0, MEAN = 1, STD = 2, MEDIAN = 1, MIN & MAX = 0. Ordering variables are also created at this time in the exact order as above. Data is then stacked, sent through a proc report and output.

At varying stages within the code, the following steps are taken to affix labels of the variables from &varlist as a internally generated format to control the output, determine the maximum length of variables, maximum width of columns, control the format of percentages, calculate totals to be shown in column headers for &acrsvar variables, and attach as a footnote (in a UNIX environment) the user ID, file name and location and its execution date & time.

**Use Existing Labels in Output of Statistics**

Proc contents is used on the incoming dataset keeping only the &varlist variables, and the output retains variable names and labels. These are then concatenated with data lines that form a proc format step which are written out to a SAS file which is then called as a macro.

```
proc contents data=&dsin(keep=&varlist)  
   out=datalist(keep=name label) noprint;  
run;  
data df1 ;  
   length fmtx $ 200;  
   fmtx = ' %macro fmtr ; ' ;  
output;  
   fmtx = '  proc format; ' ;  
output;  
   fmtx = '  value $lblfmt' ;  
output;  
run;  
data df2 ;  
   length fmtx $ 200;  
%do %while (%scan(&varlist, &h) ne %str( ) );  
   data df3 ;  
      length fmtx $ 200;  
      %if %scan(&varlist, &h) ne %str( ) %then  
         set datalist;  
      %end;  
      fmtx = ' ; run ; ' ;  
      output;  
run;  
data _null_ ;  
%end;  
set df1 df2 df3 ;  
file 'fmtr.sas' ;  
put fmtx;  
run;```

**Determine Maximum Length of Variables and Width of Columns in Output**

Use proc sql / into to create a macro variable of the length of the maximum count of patients. If the output is in fraction format, space for the numerator and denominator is needed (so double &lgth below), plus a space for the forward slash of the fraction and two to separate fraction from percentage, and two for parentheses surrounding percent, one for percent symbol and three for the percent value, which totals as 9 additional spaces.
If the output is not in the form of a fraction, only adds the 6 spaces for percentage, and one space to separate count from percentage, which totals as 7 additional spaces.

```plaintext
proc freq data=dstnct NOPRINT;
  table &acrsvar/ missing;
  out=avcnts(keep=&acrsvar count);
run;
proc sql noprint;
  select length(compress(put(max(count), best12.)))+1 into: lgth
    from avcnts;
quit;
%if &fract = Y %then
  %do;
    %let valgth = %eval(9+(&lgth * 2)) ;
  %end;
%if &fract ^= Y %then
  %do;
    %let valgth = %eval(7 + &lgth); ;
  %end;
At this point we use the %sysfunc function to get the linesize and determine if the exact column widths will fit onto the page. If they do not, we will set the first column (category column) to a width of 10 with the flow option. The second column (if appropriate, response column) will have the maximum width available after the length of the first column and all &acrsvar columns, and their appropriate spacings, have been considered.

```plaintext
%if %sysfunc(getoption(linesize)) < %eval((&valgth+&spcng)*&avmax) + (&lresp+&spcng2) + (&lcat+&spcng2) ) %then
  %do;
    %let catwid  = 10;
    %let respwid = %eval(%sysfunc(getoption(linesize)) - ((&valgth+&spcng)*&avmax) - (&catwid) - (2*&spcng2)) ;
  %end;
%if %sysfunc(getoption(linesize)) >= %eval((&valgth+&spcng)*&avmax) + (&lresp+&spcng2) + (&lcat+&spcng2) ) %then
  %do;
    %let catwid = &lcat ;
    %let respwid = &lresp ;
  %end;

```plaintext
FORMAT PERCENTAGES

Our guidance was to present percentages in the following format: If the percent is less than 1%, depict as <1%. If it the percent is greater than 99%, depict as >99%, otherwise show actual percentages. Proc format was used with a picture statement to convert the depiction of percentages calculated:

```plaintext
%do e=1 %to &avmax;
  %let t&e = %trim(%nrstr(&n)%eval(&&&num&e));
  %if &&&num&e = 0 %then
    %do;
      %let trt&e = "MISSING~(N=%trim(&&&c&e))";
    %end;
  %else %if  &&&num&e ^= 0 %then
    %do;
      %put 't&e' = &&&t&e;
      %let trt&e = "%trim(&&&t&e)~(N=%trim(&&&c&e))";
    %end;
  %end;  ;

```plaintext
CREATE TOTAL COUNTS OF &ACRSVAR FOR COLUMN HEADERS

Call symput was used in a data _null_ statement to create many useful and reusable macrovariables: sequential variables for the value, the formatted value (using &avfmt), and count of records for each value of &acrsvar groups (for use in do loops), and also the total number of distinct values of &acrsvar (for use in ending do loops.)

```plaintext
data _null_; set avcnts END=FINAL;
  n+1;
  ** NAME OF ACRSVAR **;
  call symput('N'||left(put(&acrsvar, 8.)), put(&acrsvar, &avfmt..)  );
  ** MACROVAR FOR COMBINING MULTIPLE AMPERSAND IN LET STATEMENTS FOR COLUMN HEADER TOTALS**;
  call symput('NUM'||left(put(n, 8.)), trim(&acrsvar));
  *** MACROVAR FOR COUNT OF ACRSVAR***;
  call symput('C'||left(put(n, 8.)), trim(count));
  *** MACROVAR FOR TOTAL NUMBER OF ACRSVAR ***;
  if (FINAL) then call symput('AVMAX', put(n, 8.));
run;

```plaintext
These macrovariables are used to create one macrovariable with a name directly connected to the variable name of the &acrsvar value and its value equals the formatted value of &acrsvar, using &avfmt, and the concatenation of 'N= ' and the count for that &acrsvar value.

```plaintext
%do e=1 %to &avmax;
  %let t&e = %trim(%nrstr(&n)%eval(&&&num&e));
  %if &&&num&e = 0 %then
    %do;
      %let trt&e = "MISSING~(N=%trim(&&&c&e))";
    %end;
  %else %if  &&&num&e ^= 0 %then
    %do;
      %put 't&e' = &&&t&e;
      %let trt&e = "%trim(&&&t&e)~(N=%trim(&&&c&e))";
    %end;
  %end;  ;

```plaintext
CREATE DESCRIPTIVE FOOTNOTE

Within the UNIX environment, this piece of code will find the User ID of the submittor (from system), the path and filename of the code running this macro (from extfiles in dictionary directory), and the date & time of execution from (system).

```plaintext
proc sql noprint;
  select compress(xpath) into :psource
    from dictionary.extfiles
    where index(upcase(xpath),'.SAS') > 0 ;
QUIT;
%let usr = %sysget(USER) ;
%let foot = USERID: %trim(&usr)
  %trim(&psource) &sysdate &systime    ;
footnote "&foot" ;

```plaintext
ASSUMPTIONS

Several assumptions must be clarified regarding the incoming data and data handling:

- 0 (zero) can not be a value within &acrsvar because 0 will be used for records where &acrsvar is missing.
- Both a numeric and character variable exist for the variable by which macro reports across the page in column form.
- Output will be sorted across the page by &acrsvar and down the page by the order variables are typed into &varlist. Character data will be further sorted by the frequency
(highest to lowest) of the response values.

- Numeric data will use the following order: N, Mean, SD, Median, Min, & Max.
- User must create labels for output for &varlist variables prior to macro call.
- When using &byvar, &byvar is numeric and a format exists and is entered in &byfmt.
- When &total=Y, user must set 99 = 'Total' in &avfmt.
- User has write access to work directory where fmtr.sas code is temporarily written.
- Patient Identifier variable name for incoming data set is 'pid'.

CONCLUSION
The variety of output that this macro can report is only a bonus. The key is that every clinical study requires a demography table with these specific statistics to the decimal places mentioned above. There is no reason this type of output should be programmed more than once, requiring time/effort to write the code as well as validate or check it. It is our intention that this code be used for summaries of demography, ECG findings (Normal/Abnormal/Clinically Significant), Inclusion / Exclusion Criteria Deviations, Exposure to Study Medication, and Adverse Events by demographic subgroups. As we add requested / required tables, this code and the format of its output will be proposed as a possible solution where applicable.

ACKNOWLEDGMENTS
I would like to thank Shi-Tao Yeh and Shu-Mei Huang for their review and feedback in the process of writing this code and presentation.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:

Michael Gunshenan
GlaxoSmithKline
2301 Renaissance Blvd
King of Prussia, PA 19406
Work Phone: (610) 787-3844
Fax: (610) 787 7005
Email: michael.p.gunshenan@gsk.com
Web: www.gsk.com
APPENDIX A

%MACRO dem(dsln=, varlist=, acrsvar=, avfmt=, bvar=, byfmt=, fract=N, total=Y, splitchr=);

*** SET CHARACTER AND NUMERIC GROUPING MACROVARIABLES TO ZERO OR MISSING ***;
%let varc= ;
%let varn= ;
%let maxcntc=0 ;
%let maxcntn=0 ;

*** CREATE PICTURE FORMAT TO SET PERCENTAGES CONSISTENTLY LESS THAN 1 ARE TO BE DISPLAYED AS (< 1%). PERCENTAGES GREATER THAN 99 ARE TO BE DISPLAYED AS (>99%) ***;
proc format;
  picture pp (min=6)
    . = "  " (noedit)
    0 = "  " (noedit)
    0.01 - 0.99 = "(< 1%)" (noedit)
    1.00 - 99.00 = "0009%" (prefix="( ")
    other = "(>99%)" (noedit)
  ;
run;

*** CREATE SET OF TOTAL RECORDS, IF TOTAL COLUMN IS REQUESTED ***;
%if %upcase(&total) = Y %then
  %do;
    data &dsln;
    *** USE DIFFERENT LENGTHS BASED ON DIFFERING OUTPUT FRACTIONS WILL REQUIRE WIDER FIELDS ***;
    %if &fract = Y %then
      %do;
        output;
        &acrsvar = 99 ;
        output;
      %end;
    %end;
    %let valgth = %eval(9+(&lgth * 2)) ;
    %end;
  %end;

*** CREATE A MACRO TO WRITE A PROC FORMAT TO OUTPUT VARIABLES IN PROC REPORT AS A PRE-DEFINED LABEL ***;
proc contents data=&dsln(keep=&varlist) out=datalist(keep=name label) noprint;

*** SET NAMES & COUNTS TO MACROVARIABLES ***;
data _null_;
  set avcnts END=FINAL;
  n+1;
  fmtx = ' %macro fmtr ; ' ;
  output;
  fmtx = '  proc format; ' ;
  output;
  fmtx = '  value $lblfmt' ;
  output;
run;

*** CREATE MACROVARIABLE FOR TOTAL NUMBER OF ACRSVAR ***;
if (FINAL) then call symput('AVMAX', put(n, 8.) );
output;
run;

data _null_;
  set df1 df2 df3 ;
  file 'fmtr.sas' ;
  put fmtx;
run;
%fmtr;

*** GET ACRSVAR COUNT OF DISTINCT PID USE NODUPKEY TO AVOID DOUBLE COUNTING ***;
proc sort data=&dsln out=dstnct nodupkey;
  by &acrsvar pid;
run;
proc freq data=dstnct noprint;
  table &acrsvar/ missing out=avcnts(keep=&acrsvar count);
run;

*** GET MAXIMUM COUNT VALUE TO GENERATE MAXIMUM WIDTH NEEDED TO DISPLAY COUNT OUTPUT ***;
proc sql noprint;
  select length(compress(put(max(count), best12.)))+1 into: lgth
  from avcnts;
quit;
%put '&lgth=' &lgth;

*** USE DIFFERENT LENGTHS BASED ON DIFFERING OUTPUT FRACTIONS WILL REQUIRE WIDER FIELDS ***;
%if &fract = Y %then
  %do;
    %let valgth = %eval(9+(&lgth * 2)) ;
    %end;
%if &fract ^= Y %then
  %do;
    %let valgth = %eval(7 + &lgth); 
    %end;

*** SET NAMES & COUNTS TO MACROVARIABLES ***;
data _null_;
  set avcnts END=FINAL;
  n+1;
  %macro fmtr ;
  %mend fmtr ;
  if (FINAL) then call symput('AVMAX', put(n, 8.) );
  output;
run;

*** CREATE MACROVARIABLE FOR NAME OF ACRSVAR ***;
call symput('N'||left(put(&acrsvar, 8.)), put(&acrsvar, &avfmt..) );

*** CREATE MACROVARIABLE USING SEQUENTIAL NUMBERING FOR COMBINING MULTIPLE AMPERSAND IN LET STATEMENTS FOR COLUMN HEADER TOTALS ***;
call symput('NUM'||left(put(n, 8.)), trim(&acrsvar));

*** CREATE MACROVARIABLE USING SEQUENTIAL NUMBERING FOR COUNT OF ACRSVAR ***;
call symput('C'||left(put(n, 8.)), trim(count));

*** CREATE MACROVARIABLE FOR TOTAL NUMBER OF ACRSVAR ***;
if (FINAL) then call symput('AVMAX', put(n, 8.1 ) );
run;
*** IF NO CHARACTER VARIABLES, THEN SET LENGTH WHEN MERGING FOR FINAL DATASET ***;
%if &varc = %str( ) %then
  %do;
    %let lresp = 10 ;
    %let lcat = 10 ;
  %end;
%if (&varc ne %str( ) ) %then
  %end;
*** CREATE MACROVARIABLE OF MAXIMUM NUMBER OF CHARACTER VARIABLES TO RUN THROUGH DO LOOP STAT CALCULATION ***;
data chars;
  set &dsin;
  "%trim(&&&t&e)~(N=%trim(&&&c&e))";
  %do;
    length name $ 8 ;
    call vname(chr{i}, name) ;
    %let cvar =  %scan(&varc, &l) ;
    call symput(name,'C');
  %end;
*** GET VARIABLE TYPES SET AS DEFINITIONS TO MACROVARIABLES VERSION OF VARIABLE NAMES ***;
run;
data _null_;
  set &dsin;
  %do l=1 %to &maxcntc;
    length name $ 8 ;
    call vname(nmr{j}, name) ;
    call symput(name,'N');
    proc freq data=chars noprint  ;
    table &cvar /
      stop;
    out=lcalc(rename=(&cvar=resp));
  %end;
run;
*** GET LENGTH OF RESP TEXT TO DETERMINE WIDTH OF OUTPUT COLUMN LATER ***;
*** SUBSET &VARLIST TO LISTS OF CHARACTER AND NUMERIC VARIABLES BEFORE DO LOOP, SET MACROVARIABLES FOR FIRST ITERATION CREATE MACROVARIABLE FOR SORTING ORDER OF OUTPUT ***;
%if %eval(&&&var = C) %then %if &&&cvar = C %then %do;
  %do;
    %let varc= &varc  &var ;
  %end;
  *** CREATE CAT(EGORY) VARIABLE FOR ORGANIZING OUTPUT ***;
  proc sort data=chars  out=temp&l  NODUPKEY;
  %if  &var ne %str( ) & &&&var = N %then %do;
    %let varn= &varn  &var ;
  %end;
  %let &var._o = &h ;
  %end;
  %end;
  %end;
%end;
 *** GET TOTAL COUNT FOR ACRSVAR FOR DENOMINATOR WHEN FRACT=Y ***;
  %if &fract = Y %then %do;
    proc freq data=chars NOPRINT ;
      table &acrsvar*id&acrsvar*&cvar / missing out-frcnts (keep=&acrsvar &cvar count rename=(&cvar=resp count=tcnt));
    run;
  %end;
  %end;
*** GET MAXIMUM LENGTH OF CHARACTER RESPONSE TO SET LENGTH WHEN MERGING FOR FINAL DATASET ***;
  proc freq data=chars  NOPRINT ;
    table &cvar / out=calc(rename=(&cvar=resp));
  run;
*** USE $LBLFMT, SO PRE-DEFINED LABELS ARE USED IN OUTPUT ***
proc sql;
  create table chr1 as
  if &byvar eq %str( ) %then
    select *, put(%upcase("&cvar"), $lblfmt.) as cat
  %end;
  if &byvar ne %str( ) %then
    select *, &byvar as cat
  %end;
  if &byvar = C %then
    select *, trim(put(&byvar,&byfmt..)) as cat
  %end;
  if &byvar = N %then
    *** CREATE SORTING ORDER VARIABLES FOR ORGANIZING OUTPUT ***
    select *, trim(put( length(trim(cat)),3.) ) into: lcatc as cat
    from chr1;
  quit;

*** GET LENGTH OF CAT TEXT TO DETERMINE WIDTH OF OUTPUT COLUMN LATER ***
proc sql noprint;
  select max(trim(put (length(trim(cat)),3.))) into: lcatc from chr1
  %put '&&lcatc='&&lcatc;
proc sql;
  create table frcnts as
  if &byvar eq %str( ) %then
    *** GET COUNTS BY RESPONSE ***
    proc freq data=chr1 NOPRINT ;
      table cat*&cvar / missing out=chr1.1 ; *
      (drop=percent) ;
      by &acrsvar id&acrsvar ;
  *** GET COUNT ONLY FOR SUBGROUP IF &FRAC = Y IN_YN VARIABLE MUST BE SET TO 'Y' OUTSIDE THIS MACRO ***
  %if &fract = Y %then
    %do;
      where in_yn = 'Y' ;
    %end;
    run;
  %if &byvar ne %str( ) %then
    %do;
      %if &fract = Y %then
        %do;
          %if &&&byvar = C %then
            %do;
              %do;
                where in_yn = 'Y' ;
              %end;
            %end;
          %end;
        %end;
      %if &&&byvar = N %then
        *** COUNT RESPONSES FOR n IN OUTPUT ***
        proc sql;
          create table n1 as
            select distinct &acrsvar, id&acrsvar, 'n' as resp, 'ANY EVENT' as cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
            from chr1.1
            group by &acrsvar, id&acrsvar
        %end;
        proc sort data=chr1.1;
          by &acrsvar id&acrsvar cat_o cat resp;
        quit;

        %if &fract = Y %then
          %do;
            create table frcnts as
              select distinct &acrsvar, id&acrsvar, 'n' as resp, cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
              from chr1.1
              group by &acrsvar, id&acrsvar
            %end;
            proc sort data=chr1.1;
              by &acrsvar id&acrsvar cat_o cat resp;
          quit;
          %do;
            %if &fract ^= Y, LABEL FOR COUNT MUST BE A LOWER CASE N ***;
            %if &fract ^= Y %then
            %do;
              create table n1 as
                select distinct &acrsvar, id&acrsvar, 'n' as resp, cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
                from chr1.1
                group by &acrsvar, id&acrsvar
              %end;
              proc sort data=chr1.1;
                by &acrsvar id&acrsvar cat_o cat resp;
              quit;
            %end;
            %end;
          %end;
          %if &fract ^= Y %then
            %do;
              create table n1 as
                select distinct &acrsvar, id&acrsvar, 'n' as resp, cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
                from chr1.1
                group by &acrsvar, id&acrsvar
              %end;
              proc sort data=chr1.1;
                by &acrsvar id&acrsvar cat_o cat resp;
              quit;
            %end;
            %end;
          %end;
        %end;
        %if &byvar ne %str( ) %then
          %do;
            %if &&&byvar = C %then
              %do;
                %do;
                  %end;
              %end;
            %end;
          %if &&&byvar = N %then
            *** COUNT RESPONSES FOR n IN OUTPUT ***
            proc sql;
              create table n1 as
                select distinct &acrsvar, id&acrsvar, 'n' as resp, cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
                from chr1.1
                group by &acrsvar, id&acrsvar
              %end;
              from frcnts
              order by &acrsvar, id&acrsvar, cat_o, cat, resp;
            quit;
            %end;
          %end;
        %end;
      %if &byvar ne %str( ) %then
        %do;
          %if &&&byvar = C %then
            %do;
              %do;
                %end;
            %end;
          %end;
        %if &&&byvar = N %then
          *** COUNT RESPONSES FOR n IN OUTPUT ***
          proc sql;
            create table n1 as
              select distinct &acrsvar, id&acrsvar, 'n' as resp, cat, 0 as cat_o, 0 as resp_o, sum(count) as cnt
              from chr1.1
              group by &acrsvar, id&acrsvar
            %end;
            from frcnts
            order by &acrsvar, id&acrsvar, cat_o, cat, resp;
          quit;
        %end;
      %end;
    %end;
  %end;
%end;
id id\&acrsvar 

** CREATING ORDERING VARIABLES USING PERCENT - HIGHEST TO LOWEST **;
proc sql;
    create table ord\&l as
    select cat_o, cat, resp, max(percent) as mpct
    from chr\&l.1
    group by cat_o, cat, resp;
quit;

proc sort data=ord\&l;
    by cat_o cat DESCENDING mpct resp;
run;

data ord\&l;
    set ord\&l;
    by cat_o cat DESCENDING mpct resp;
    resp_o+1;
run;

** SET ORDERING VARIABLES BACK ONTO DATA **;
proc sql;
    create table chr\&l.1 as
    do;
        select a.*, b.resp_o
        if acrs%eval(&&&num&m)p <= 0 then
            from chr\&l.1  a
            left join ord\&l    b
            on a.cat_o= b.cat_o
            and a.cat  = b.cat
            acrs%eval(&&&num&m)p = "  " ;
        if acrs%eval(&&&num&m)p >  0 then
            do;
                acrs%eval(&&&num&m)pct =
                acrs%eval(&&&num&m) / &&&c&m ;
            end;
        end;
    end;
quit;

** TRANSPOSE RESPONSE COUNT DATA BY ACROSS VARIABLE **;
proc sort data=n\&l;
    by cat_o cat resp_o resp ;
run;

proc transpose data=n\&l out=t_n\&l
(drop=_name_ _label_ );
    var cnt ;
    by cat_o cat resp_o resp ;
run;

** TRANSPOSE FREQ DATA BY ACROSS VARIABLE **;
proc sort data=chr\&l.1;
    by cat_o cat resp_o resp ;
run;

proc transpose data=chr\&l.1 out=t_cnt\&l
(drop=_name_ /*_label_ */);
    %if &fract = Y %then
    %do;
        var fcnt ;
    %end;
    %if &fract ^= Y %then
    %do;
        var cnt ;
    %end;
    by cat_o cat resp_o resp ;
run;

** CREATE MACROVARIABLE OF MAXIMUM NUMBER OF NUMERIC VARIABLES TO RUN THROUGH DO LOOP STAT CALCULATION **;

%end;
%end;
%end;

%if (&varn ne %str( )) %then
    %do;
        id\&acrsvar = compress('acrs'||\&acrsvar);
    run;
%end;

*** STACK STATS, CONVERT DATA TO ALIGNED TEXT FOR OUTPUT COLUMNS ***;
data cfin\&l (drop-acs:);
    length resp $ &&lrspc\&l 
cat $ 20
proc sql;
    val: $ &valgth
create table ord\&l as
%do m=1 %to &avmax;
select cat_o, cat, resp, max(percent) as mpct
%end;  ;
from chr\&l.1
set
/* WHEN &FRACT ^= Y, TOTAL COUNT IS NEEDED FOR ALL VARIABLES. */
quit;
%if &fract ^= Y %then
    proc sort data=ord\&l;
    %do;
    by cat_o cat DESCENDING mpct resp;
    t_n\&l(in=a)
    run;
    %end;
t_cnt\&l 
data ord\&l;
%do m=1 %to &avmax;
set ord\&l;
    by cat_o cat DESCENDING mpct resp;
    resp_o+1;
%end;
%end;
%end;
%end;

%if &fract  = Y %then
    proc transpose data=n\&l out=t_n\&l
(drop=_name_ );
    %do;
    val%eval(&&&num&m) =
    var cnt ;
    acrs%eval(&&&num&m) ;
    by cat_o cat resp_o resp ;
    %end;
%end;

proc sort data=chr\&l.1;
    by cat_o cat resp_o resp ;
run;

proc transpose data=chr\&l.1 out=t_cnt\&l
(drop=_name_ /* _label_ */);
data nums ;
    set &dsin;
    array num {*} &varn;
    do i=1 to dim(num);
        call symput('MAXCNTN', i);
    end;
    id\&acrsvar = compress('acrs'||\&acrsvar);
    run;

%do w=1 %to &maxcntn;
  %let nvar = %scan(&varn, &w);
%if &&nvar = N %then
  %do;
    proc sort data=nums out=temp&w NODUPKEY;
    by pid &acrsvar id&acrsvar &varlist;
  run;

  *** CREATE CAT(EGORY) & CAT_O VARIABLE FOR ORGANIZING OUTPUT ***;
  proc sql;
  create table num&w as
    %if &byvar eq %str( ) %then
      %do;
        select *, put(%upcase("&nvar"), &lblfmt.) as cat, &&&nvar._o as cat_o
      %end;
    %if &byvar ne %str( ) %then
      %if &&&byvar = C %then
        select *, &byvar as cat, &&&nvar._o as cat_o
      %else %if &&&byvar = N %then
        select *, trim(put(&byvar,&byfmt..)) as cat, &&&nvar._o as cat_o
      %end;
    %end;
  from temp&w
  order by &acrsvar, id&acrsvar, cat_o, cat;
  quit;

  proc sql noprint;
  select max(trim(put(length(trim(cat)),3.)))
  into: lcatn&w
  from num&w;
  %put '&&lcatn&w='&&lcatn&w;
  %end;
  %end;
  %end;

  *** GET MAXIMUM LENGTH OF RESP VARIABLE FOR COLUMN WIDTH DETERMINATION ***;
  %if (&varc ne %str( ) ) %then
    %do;
      %let lresp = &lrspc1;
      %do n=1 %to &maxcntc;
        %if &&lrspc&n > &lresp %then
          %do;
            %let lresp = &&lrspc&n;
          %end;
      %end;
      %put '&lresp='&lresp;
    %end;
    run;
  %end;

proc means data=num&w noprint;
  var &nvar;
  output out=num&w.2(drop=_freq_ _type_)
    n=n mean=mean std=sd median=median
    min=min max=max;
  by &acrsvar id&acrsvar cat_o cat;
%if &fract = Y %then
  %do;
    where in_yn = 'Y';
  %end;
run;

*** TRANSPOSE FREQ DATA BY ACROSS VARIABLE ***;
proc transpose data=num&w.2 out=t_cnt&w(drop=_label rename=(_name_=resp))
  by cat_o cat;
  id id&acrsvar;
  idlabel &acrsvar;
run;

*** CONVERT DATA TO ALIGNED TEXT FOR OUTPUT

COLUMNS CREATE RESP_O FOR SORTING VARIABLES FOR OUTPUT ***;
  data nfin&w (drop=acrs:);
    length val: %eval(&valgth).2;
  set t_cnt&w;
%do m=1 %to &avmax;
  select(upcase(resp));
  when('N') do;
    resp = lowcase(resp);
  %end;
  when('MEAN') do;
    resp_o=2;
  val%eval(&&&num&m) =
    put(acrs%eval(&&&num&m),%eval(&lgth+2).1)||'
    ';
  end;
  when('SD') do;
    resp_o=3;
  val%eval(&&&num&m) =
    put(acrs%eval(&&&num&m),%eval(&lgth+3).2)||'
    ';
  end;
  when('MEDIAN') do;
    resp_o=4;
  val%eval(&&&num&m) =
    put(acrs%eval(&&&num&m),%eval(&lgth+2).1)||'
    ';
  end;
  when('MIN') do;
    resp_o=5;
  val%eval(&&&num&m) =
    put(acrs%eval(&&&num&m),%eval(&lgth+2).1)||'
    ';
  end;
  when('MAX') do;
    resp_o=6;
  val%eval(&&&num&m) =
    put(acrs%eval(&&&num&m),%eval(&lgth+2.0)||'
    ';
  end;
%end;
%end;
%end;
%end;
%end;
%end;
%end;
*** GET MAXIMUM LENGTH OF CAT VARIABLE FOR COLUMN WIDTH DETERMINATION ***;
%if (&varc ne %str( ) ) %then
  %do;
    %let lcat = &lcatc1;
    %do n=1 %to &maxcntc;
      %if &&lcatc&n > &lcat %then
        %do;
          %let lcat = &&lcatc&n;
        %end;
    %put '&lcat=' &lcat ;
  %end;
%end;
%if (&varn ne %str( ) ) %then
  %do;
    %if (&varc eq %str( ) ) %then %let lcat = &lcatn1;
    %do n=1 %to &maxcntn;
      %if &&lcatn&n > &lcat %then
        %do;
          %let lcat = &&lcatn&n;
        %end;
    %put '&lcat=' &lcat ;
  %end;
%end;

*** SET ALL PREPARED DATA FOR REPORTING ***;
data final (drop= acrs:);
  length  resp $ &lresp   cat $ &lcat ;
  set %do n=1 %to &maxcntc;
cfin&n
  %end;
  %do n=1 %to &maxcntn;
nfin&n
  %end;
/* WHEN &FRACT = Y, TOTAL COUNT (ANY EVENT) IS CREATED FOR ALL CHARACTER VARIABLES. ONLY ONE IS NEEDED. TAKE 1ST */
%if &fRACT = Y %then
  %do;
    t_n1(in=a)
  %end;
  if a then
    %do m=1 %to &avmax;
      val%eval(&&&num&m)=acrs%eval(&&&num&m) ;
    %end;
  %end;
run;

*** DETERMINE COLUMN WIDTHS BASED UPON PRE- SET LINESIZE, ACSVAR COLUMNS USE SMALLER SPACING FOR APSVAR COLUMNS THAN FOR CATEGORY / RESPONSE COLUMNS ***;
%let spcng = 1;
%let spcng2= %eval(&spcng+2);

*** SET FOOTNOTE OF USER ID, FILE NAME / LOCATION DATE AND TIME OF FILE EXECUTION ***;
%let usr = %sysget(USER) ;
%let foot = USERID: %trim(&usr) %trim(&psource) &sysdate &systime ;
footnote "&foot" ;

*** SET FOOTNOTE OF USER ID, FILE NAME / LOCATION DATE AND TIME OF FILE EXECUTION ***;
%let usr = %sysget(USER) ;
%let foot = USERID: %trim(&usr) %trim(&psource) &sysdate &systime ;
footnote "&foot" ;

*** DETERMINE COLUMN WIDTHS BASED UPON PRE- SET LINESIZE, ACSVAR COLUMNS USE SMALLER SPACING FOR APSVAR COLUMNS THAN FOR CATEGORY / RESPONSE COLUMNS ***;
%let spcng = 1;
%let spcng2= %eval(&spcng+2);

*** IF CAT/RESP COLUMNS ARE TOO WIDE, RESET CAT TO 10 (LENGTH OF 'PARAMETER' TITLE + 1, AND RESP TO SPACE LEFT OVER ***;
%if %sysfunc(getoption(linesize)) < %eval( ((&valgth+&spcng)*&avmax) + (&lresp+&spcng2) + (&lcat+&spcng2) ) %then
  %do;
    %let catwid  = 10;
    %let respwid = %eval( %sysfunc( getoption(linesize) ) - ( (&valgth+&spcng)*&avmax ) - (catwid) - (2*&spcng2) ) ;
  %end;
%if %sysfunc(getoption(linesize)) >= %eval( ((&valgth+&spcng)*&avmax) + (&lresp+&spcng2) + (&lcat+&spcng2) ) %then
  %do;
    %let catwid = &lcat ;
    %let respwid = &lresp ;
  %end;

proc report data=final headline headskip
  split="&spltchr" missing spacing=&spcng;
  column cat_o cat resp_o resp
  %do o=1 %to &avmax;
    val%eval(&&&num&o)
  %end;
  define cat_o / order order=internal noprint ;
  define cat / order order=internal " 
    width=&catwid  flow spacing=&spcng2; 
  define resp_o / order order=internal noprint ;
  define resp / order order=internal " 
    width=&respwid flow spacing=&spcng2 ;
  %do p=1 %to &avmax;
    define val%eval(&&&num&p) / display 
      %trim(&&&trt&p) width=&valgth center ;
  %end;
  break after cat/skip;
run;
%MEND dem;

*** REMOVE EXCESS BAGGAGE FROM MEMORY *** ;
proc datasets;
  delete c: n: t_: temp: frcnts d: ;
run;

*** GET FILE NAME, INCLUDING PATH, FROM DICTIONARY.EXTFILES ***;
proc sql noprint;
  select compress(xpath) into :psource
    from dictionary.extfiles
    where index(upcase(xpath),'.SAS') > 0
  ;
quit;