Paper 236-28
An Automated Reporting Macro to Create Cell Index –
An Enhanced Revisit
Shi-Tao Yeh, GlaxoSmithKline, King of Prussia, PA

ABSTRACT

When generating tables from SAS® PROC TABULATE or PROC REPORT to summarize data, sometimes it is necessary to produce a cell index as part of the data display. A cell index is simply a data listing that reports the summarized count of each cell shown in the table.

This topic was presented and discussed at NESUG'01 Conference [7]. Several enhancements have since been made resulting in a major modification of the original code. It provides an automated SAS reporting macro to generate a cell index with several new features. The design goal for this program is to provide an efficient, flexible and easy-to-use macro.

The SAS product used in this paper is SAS BASE, with no limitation of operating systems.

INTRODUCTION

SAS System provides several useful tools for the production of tables and tabulations that summarize data. The procedures, PROC REPORT, and PROC TABULATE are very popular.

It is sometimes required to generate a cell index as part of the data display after producing a summarized table. The clinical study reviewer or SAS programmer uses a cell index for program validation purposes. The difference between a data listing and a cell index is that a cell index displays only categorical variables that appear in the table. It displays data observations to support the summary count shown in each cell of the table. The last column of a cell index usually displays the subject variable. It is also a preferred design to have the column before the last as a numeric count of the sub-categorical grouping of the subject variable.

The features and macro arguments of this macro are:

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsin</td>
<td>a dataset name from which the cell index is to be produced</td>
</tr>
<tr>
<td>sortby</td>
<td>a categorical variable list that appeared in the table</td>
</tr>
<tr>
<td>idvar</td>
<td>a subject variable name</td>
</tr>
<tr>
<td>num_var</td>
<td>a variable that provides additional information for the cell index</td>
</tr>
<tr>
<td>freqsort</td>
<td>a variable name that is used for descending sort order</td>
</tr>
<tr>
<td>nprintv</td>
<td>a dummy variable that provides the correct sort order but not showing on cell index output</td>
</tr>
<tr>
<td>pageby</td>
<td>a variable list for output separated by page by variable list</td>
</tr>
</tbody>
</table>

The new and enhanced features are: num_var, freqsort, nprintv, and pageby.

This paper provides a step-by-step approach to developing a SAS reporting module for cell index production.

This paper is comprised of five parts. Part 1 includes the abstract and the introduction. Part 2 describes the sample data used throughout the paper. Part 3 is devoted to the program design and SAS code. Part 4 involves the SAS macro invocation and application. Part 5 concludes the paper.

SAS MACRO DESIGN SPECIFICATION FOR CELL INDEX

The program requirements for this macro are efficient, flexible and easy-to-use. This design goal can be achieved by minimizing the macro arguments. The framework of the program flow is described as follows:

The main code consists of six steps:

STEP 1:
This step accepts the user's input macro argument information and prepares the dataset for the cell index output.

There are seven macro arguments, namely, dsin, sortby, num_var, freqsort, nprintv, and idvar.

STEP 2:
This step performs three counts: one for number of variables in the sortby list, the other for number of subject observations in each sub-category grouping, the last one for the space width available for listing of all variables in the sortby list.

STEP 3:
This step performs specific features for variables freqsort, nprintv, pageby and num_var.

STEP 4:
This step creates all SAS statements related to the procedure PROC REPORT. Then the SAS macro statements create a temporary SAS macro file named ctemp.sas.

STEP 5:
SAS file ctemp.sas is run to create a cell index.
STEP 6:

All data files and temporary SAS macro files are deleted after the execution of the SAS macro `get_cell.sas`.

**Note:** The X statement used in this program is for the UNIX platform. You need to issue your operating system command to delete the temporary file `ctemp.sas`.

**Note:** The statement

```
%let dwidth=%eval(%eval(94/%eval(&j-1)));
```

is for reserved space width for all sortby variables. You can make further adjustments to fit your needs.

SAS CODE FOR CELL INDEX

The whole SAS macro code are shown as follows:

```sas
%macro get_cell(dsin=, sortby= , idvar=, num_var=, freqsort=N, noprintv=, pageby=);
  ***********************************************;
  * read in dataset ;
  ***********************************************;
  data dsin;
    set &dsin;
    count=1;
  proc sort;
    by &sortby;
    %if &pageby ne %then %do;
      proc sort data=dsin nodupkey;
        by &pageby &sortby &idvar;
    %end;
  run;
  ***********************************************;
  * using &sortby for grouping to count number of * objects in each cell
  ***********************************************;
  proc summary data= dsin;
    by &pageby &sortby;
    var count;
    output out=pxyz1 sum=n;
  data dsin;
    merge dsin pxyz1;
    by &pageby &sortby;
  run;
  %if %upcase(&freqsort) eq Y %then %do;
    proc sort data=dsin;by descending n ;
    data dsinn(keep=orderv n);
      set dsin;
      output out=pxyz1 sum=n;
    data dsin;
      merge dsin pxyz1;
    by &pageby &sortby &idvar;
    run;
  %end;
  %global sortbyn dwidth;
  %let sortbyn=%eval(&j-1);
  %if &num_var ne %then %do;
    %let dwidth=%eval(%eval(81/%eval(&j-2)));%end;
  %if &noprintv ne %then %do;
    %let dwidth=%eval(%eval(81/%eval(&j-2)));%end;
  %end;
  %mend varn;
  %varn;
  ***********************************************;
  * covert &sortby list to a column that contains * each variable in &sortby list* as a record with one &sortby variable
  ***********************************************;
  data f1;
    length &sortby 4.;
    %do i=1 %to &sortbyn;
      %scan(&sortby,&i)=1;
    %end;
  run;
  proc transpose data=f1 out=f2(keep=_name_);
    var &sortby;
  data f2;
    set f2;
    if _name_ = "%upcase(&noprintv)" then delete;
  run;
  proc sort data=dsin;by &sortby &idvar;
  %end;
  run;
  proc transpose data=f1 out=f2(keep=_name_);
    var &sortby;
  data f2;
    set f2;
    if _name_ = "%upcase(&noprintv)" then delete;
  run;
  proc sort data=dsin;by &sortby &idvar;
  %end;
  run;
  %if %upcase(&freqsort) eq Y %then %do;
    proc sort data=dsin;by descending n ;
    data dsinn(keep=orderv n);
      set dsin;
      output out=pxyz1 sum=n;
    data dsin;
      merge dsin pxyz1;
      by &pageby &sortby &idvar;
    run;
  %end;
  ***********************************************;
  * construct a dataset with define statement for &sortby argument
  ***********************************************;
  %macro varn;
    %local j;
    %let j=1;
    %let varn=%scan(&sortby,&j);
    %do %while ("%varn" ne "");
      %let j=%eval(&j+1);
      %let varn=%scan(&sortby,&j);
    %end;
    %global sortbyn dwidth;
    %let sortbyn=%eval(&j-1);
    %if &num_var ne %then %do;
      %let dwidth=%eval(%eval(81/%eval(&j-2)));%end;
      %let dwidth=%eval(%eval(81/%eval(&j-2)));%end;
    %end;
  %mend varn;
  %varn;
  ***********************************************;
  * covert &sortby list to a column that contains * each variable in &sortby list* as a record with one &sortby variable
  ***********************************************;
  data f1;
    length &sortby 4.;
    %do i=1 %to &sortbyn;
      %scan(&sortby,&i)=1;
    %end;
  run;
  proc transpose data=f1 out=f2(keep=_name_);
    var &sortby;
  data f2;
    set f2;
    if _name_ = "%upcase(&noprintv)" then delete;
  run;
  proc sort data=dsin;by &sortby &idvar;
  %end;
  run;
```

*SUGI 28 Posters*
SAS MACRO INVOCATION AND APPLICATION

Before this macro invocation, it is important that you use the DATA STEP to prepare the SAS data set and label variables to appear in the cell index. In this section, individual an invocation example with code and a figure showing the relationship between table and cell index are provided. The sample macro invocation example 1 is as follows:

```sas
%get_cell(dsin = final, sortby = trt agec sex, idvar = patient);
```

Note: This macro accepts categorical variables in the sortby argument. If you have numeric variables in the sortby list, you need to convert them to character variables before invoking the macro.

A summary table displays the treatment variable as the across variable with categorical variables of age group and sex. The output of summary table and cell index from this example 1 is shown in Figure 1.

![Figure 1 Output of Summary Table and Cell Index from Example 1](image)

Example 2 uses the feature `num_var`, to add additional variable to the cell index, either numeric or character variable, and specifies it as age. The output from this example is shown in Figure 2.

![Figure 2 Output of Summary Table and Cell Index from Example 2](image)
%get_cell(dsin = final, sortby = trt agec sex, num_var = age, idvar = patient);

Figure 2  Cell Index with Additional Variable num_var Specified

Sometimes the output table contains a certain sort order for the key variable. It is desired that the cell index contains the sort order for the same variable. You can create a ‘no print’ variable that controls the desired sort order without printing it on the output. Example 3 is a sample of this type of output.

%get_cell(dsin = final, sortby = trt_o trt agec sex, num_var = age, noprintv = trt_o, idvar = patient);

Sometimes the output table program contains a by statement that output is breaking by this 'pageby' variable. It is desired that the cell index contains the same feature to break the output by 'pageby' variable. Example 4 is a sample of this type of output.

%get_cell(dsin = lab, sortby = visit range, num_var = labhl, pageby = labtest trt, idvar = pid);

Figure 3 Cell Index with noprintv Specified

Figure 4. Cell Index with pageby Specified
Sometimes the output table contains a certain descending sort order for the key variable. It is desired that the cell index contains the descending sort order for the frequency count \( N \). Example 5 is a sample of this type of output.

\[
%\text{get\_cell}(\text{dsin = aesub, sortby = pref trt, freqsort = Y, idvar = pid});
\]

**CONCLUSION**

This paper takes a comprehensive approach in designing a reporting module for creation of a cell index. SAS code and the examples of how to invoke the SAS macro are provided in this paper. This paper achieves the following goals:

* Provides an easy-to-use macro with only seven macro arguments
* Provides enhanced features to fit the different needs

**ACKNOWLEDGMENTS**

The author would like to thank Ellen Forman, Shu-Mei Huang, Biostatistics & Data Management, GlaxoSmithKline, and Ju Zhang from EDP Contract Services, for their review, validation and enhancement suggestions of this macro.

**REFERENCES**


SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA
© indicates USA registration.

Author
Shi-Tao Yeh, Ph. D.
(610)917-5883(W)
E-mail: shi-tao_yeh-1@gsk.com