PharmaSUG China 2015 – Paper 02

Fifty Shades of Sorting

Haibin Shu, AccuClin Global Services LLC, Wayne, PA
John He, AccuClin Global Services LLC, Wayne, PA
Elena Rojco, DentaQuest, Salem, NH

ABSTRACT

The title of 'Fifty Shades of Grey' means a lot of facets of the main character Grey's personality, likewise, the tricks and tips of sorting variables in SAS go way beyond the syntax of PROC SORT. In many circumstances a good order does matter therefore customized sorting variables are to be adopted for achieving a desirable sorting effect. A variety of examples will be presented to illustrate in details why customized sorting variables are needed in these specific situations and how they are derived accordingly. Further, the ideas can be generalized to prototype/establish a systematic approach. For the consideration of the effect of reporting and presenting information that often depends on whether a good order is in place SAS programmers may use PROC PRINT, PROC REPORT, or even a DATA SET with sorting techniques to consistently solve certain challenging issues.

INTRODUCTION

The authors explore several programming methods to enhance what PROC SORT can do by applying PROC SORT in an innovative way. We'll explain through examples why a straight-forward use of PROC SORT may not work in these circumstances and our approaches employ customized variables for sorting purpose. The paper has been prepared with the assumption of familiarity with PROC SORT and basic sorting options such as NODUPS and NODUPKEYS etc. Our data sets used throughout the examples are based on CDISC compatible structures. We'd also make a note that the methods presented in this paper don't mean to be exclusive because it's generally true that there're always a different way to solve a problem in SAS!

USING NUMERIC VARIABLES

The sorted variables are commonly used for ordinal data in statistical analysis. A simple case of creating a customized sorting variable is to assign a sequential numeric values to a character variable. The Adverse Events (AE) Reports are usually summarized by severity levels such as "Mild", "Moderate", "Severe", and "Life Threatening". In creation of AE reports, a numeric variable is often created to signify the severity level, such as 1 -> "Mild", 2 -> "Moderate", 3 -> "Severe", and 4 -> "Life Threatening". Not only this numerical variable is used in PROC REPORT as a sorting variable (most time non-displayable) but also it is used to count the most severe case in incidence reporting. The order of this data is necessary to group the summary statistics at severity levels. Here is the expected result of the lay out – included both incidence rate and events count:

System Organ Class **Preferred Term**	**Treatment 1**	**Treatment 2**	**Overall**
SOC Term 1	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy
AE Term 1	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy
Mild	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy
Moderate	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy
Severe	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy
Life Threatening	xx (xx.x) yy	xx (xx.x) yy	xx (xx.x) yy

SORTING BY DICTIONARY TERMS - AE, CM, MH, etc.

Sorting event variable in dictionary terms is a typical situation to create customized variables. Here we use a common requirement of descendent order in AE reporting for illustration. There are a few variables to be considered for the sorting orders. The first level is SOC, and the second is PT, then the event incident rates of SOC and PT. We may first sort the SOC terms according to the descendant order of incidence rate such as:

```
proc sort data=AE out=AE_SOC1(keep=AEBODSYS) nodupkeys;
     by descendent _incidence_rate_soc AEBODSYS;
quit;
```
Then a numeric variable is generated to represent the order of the above SOC terms:
```
data AE_SOC2;
     set AE_SOC1;
     _soc_order+1;
run;
```

Similarly a separate order variable _pt_order for AEDECOD is created to represent the descending order of AE preferred terms within SOC terms. These two customized variables together adequately define the descending order of AE reports by System Organ Class and Preferred Term. The reporting data set looks like this:

_soc_order	_pt_order	System Organ Class	_incidence_rate_soc	Preferred Term	_incidence_rate_pt
1	1	SOC Term1	0.4	AE Term11	0.15
1	2	SOC Term1	0.4	AE Term12	0.10
1	3	SOC Term1	0.4	AE Term13	0.05
1	4	SOC Term1	0.4	AE Term14	0.01
2	1	SOC Term2	0.3	AE Term21	0.20
2	2	SOC Term2	0.3	AE Term22	0.10
2	3	SOC Term2	0.3	AE Term23	0.05

Note: in the above example, the custom variables _soc_order and _pt_order can be substituted directly by the incidence variables _incidence_rate_soc and _incidence_rate_pt respectively.

We may also use PROC SQL to sort and merge the data (ready to order the rates of SOC and PT):

```
proc sql _AE;
     create table adsl_ae(label='ADSL and AE dataset') as
     select  * from admin(sortedby=usubjid ) T1, ae(sortedby=usubjid soc pt) T2
     where T1.usubjid=T2.usubjid;
quit;
```

From the above steps you can combine sorting and merging data together in one procedure. The idea is the same: for each level of sorting a numeric variable associated with the term variable will be created; then you can decide the superiority of these sorted variables.

The following data step is another way of creating the numeric sorting variables _soc_order and _pt_order:

```
data…
    by …
     if first.pt then do;
        _pt_order+1;
        if first.soc then do;
           _soc_order+1;
           _pt_order=1;
        end;
     end;
run;
```

ORDERING WITH MULTIPLE LEVELS

After discussion of ordering the terms, we can go deeper into multiple levels of sorting. The method in the previous section can be generalized for sorting with multiple-level hierarchies. For each level a sorting variable is created according to specified requirement. When there're only two levels it can reduce to just one as the SOC Term/AE Term sorting combination in the previous section.

We use symbols L_1, L_2, …, L_n to represent a reporting with n levels, each succeeding level L_{j+1} should be imbedded within the previous level L_j. This is just like the relationship between AE preferred term and AE body system term.

Suppose R_1 to represent the reporting variable at the highest level $L_{1,}$ such as frequency rates and/or events counts in the example of AE incidence table, we use O_1 to indicate the sorting order. Depending on the requirement, O_1 can be derived numerically by a number of different ways: ascending or descending order of $L_{1,}$ ascending or descending order of $R_{1,}$ or other specific orders. The combined data components for level L_1 are: $O_{1,}$ $L_{1,}$ $R_{1.}$

When constructing the next level L_2 components, the relationship with the previous level L_1 should be taken into consideration. Since L_2 is imbedded within L_1, we use L_1*L_2 to represent the nested levels, and R_{12} and O_{12} to indicate the reporting variable and order variable respectively. Therefore the combined data components for level L_2 are: O_{12}, L_1*L_2, R_{12}.

To report R_1 and R_{12} according to the hierarchy structure by the order variables O_1 and O_{12} as previously defined, we define a new text variable C over L_1 U L_1*L_2 such as $C(L_1) = L_1$, $C(L_1*L_2)= L_2$. The corresponding reporting variable should be $R(L_1) = R_1$ and $R(L_1*L_2)= R_{12}$. Hence the combined data components for levels L_1 and L_2 are as follows:

Order		Non-display	C	R
O_1		L_1	L_1	R_1
O_1	O_{12}	L_1*L_2	L_2	R_{12}

Similarly the combined data components for levels L_1, L_2, and L_3 are as follows:

Order			Non-display	C	R
O_1			L_1	L_1	R_1
O_1	O_{12}		L_1*L_2	L_2	R_{12}
O_1	O_{12}	O_{123}	L_1*L_2*L_3	L_3	R_{123}

We can expand 3-level components to any n-level data as represented below:

Order			Non-display	C	R
O_1			L_1	L_1	R_1
O_1	O_{12}		L_1*L_2	L_2	R_{12}
			…		
O_1	O_{12}	$O_{12\ldots n}$	L_1*L_2…L_n	L_n	$R_{12\ldots n}$

We now apply the process to the AE incidence table by severity (n=3) as presented previously as the first table of this paper. Obviously we have L1=AEBODSYS, L2=AEDECOD, and L3=AESEV. R1 can be obtained as:

```
/*each AEBODSYS only counts once*/
proc sort data=ADAE out=AEB nodupkeys;
     by USUBJID AEBODSYS;
quit;
proc freq data=AEB;
     tables AEBODSYS*TRT01AN/out=AEB2(keep=AEBODSYS TRT01AN count);
quit;
```

Calculate the incidence rate R_1 after adding the denominators DTRT01AN:

```
data AEB3;
     set AEB2x;
     R1=count/DTRT01AN*100;
run;
```

Assuming O_1 is the ascending order of R_1 on TRT01AN=9:

```
proc sort data=AEB3 out=AEB_O;
     by descending R1 AEBODSYS;
     where TRT01AN=9;
quit;
```

```
data AEB_O2;
     set AEB_O;
     by descending R1 AEBODSYS;
     O1+1;
     keep AEBODSYS O1;
run;
```

Adding O_1 back into AEB2 via merging by AEBODSYS:

```
data AEB4;
     merge AEB3 AEB_O2;
     By AEBODSYS;
run;
```

Repeat the calculation and derivation for variables R_{12} and O_{12}:

```
proc sort data=ADAE out=AEP nodupkeys;
     by USUBJID AEBODSYS AEDECOD;
proc freq data=AEP;
     tables AEBODSYS*AEDECOD*TRT01AN/out=AEP2(keep=AEBODSYS AEDECOD TRT01AN count);
quit;
```

Calculate the incidence rate R_{12} after adding the denominators DTRT01AN:

```
data AEP3;
     set AEP2x;
     R12=count/DTRT01AN*100;
run;
```

Assuming O_{12} is the ascending order of R_{12} on TRT01AN=9:

```
proc sort data=AEP3 out=AEP_O;
     by AEBODSYS descending R12 AEDECOD;
     where TRT01AN=9;
quit;

data AEP_O2;
     set AEP_O;
     by AEBODSYS descending R1 AEDECOD;
     O12+1;
     keep AEBODSYS AEDECOD O12;
run;
```

Adding O_{12} back into AEP3 via merging by AEBODSYS AEDECOD:

```
data AEP4;
     merge AEP3 AEP_O2;
     By AEBODSYS AEDECOD;
run;
```

Similarly calculate R_{123} and derive O_{123}:

```
proc sort data=ADAE out=AES nodupkeys;
     by USUBJID AEBODSYS AEDECOD AESEV;
     where AOCCPIFL=’Y’; /*pick up the highest severity*/
quit;
proc freq data=AES;
     tables AEBODSYS*AEDECOD*AESEV*TRT01AN/out=AES2(keep=AEBODSYS AEDECOD AESEV
                                                          TRT01AN count);
quit;
```

```
data AES3;
     set AES2x;
     R123=count/DTRT01AN*100;
run;
```

Assuming O_{123} is the pre-specified order of R_{123} based on the severity code: Mild, Moderate, Severe, Life Threating:

```
data AES4;
     set AES3;
     select(AESEV);
       when('Mild') O123=1;
       when('Moderate') O123=2;
       when('Severe') O123=3;
       when('Life Threating') O123=4;
       otherwise;
     end;
run;
```

After the ordering components captured, we now can combine the 3 levels together: AEBODSYS, AEDECOD, and AESEV, and derive the overall text variable C and reporting variable R:

```
data _rpt;
     set AEB4(in=in1) AEP4(in=in2) AES4(in=in3);
     if in1 then do;
        C=AEBODSYS;
        R=R1;
        O12=0;
        O123=0;
     end;
        else if in2 then do;
             C=AEDECOD;
             R=R12;
             O1=input(AEBODSYS,L1f.);
             O123=0;
        end;
           else do;
                C=AESEV;
                R=R123;
                O1= input(AEBODSYS,L1f.);
                O2= input(AEBODSYS||AEDECOD,L2f.);
           end;
run;
```

In the above statement, L1f. and L2f. are the formats that map to the order variables for specific AEBODSYS and AEBODSYS/AEDECOD combinations. Sort the final reporting data set:

```
proc sort data=_rpt;
     by O1 AEBODSYS O12 AEDECOD O123 AESEV;
quit;
```

It's not difficult to generalize the above case of n=3 to a general case with more than 3 levels. Since each step is well defined and the progress from one step to the next is also well defined in a consistent way, a macro routine can be further developed to streamline the programming. The authors plan to share the macro approach in a separate paper.

ORDERING WITHOUT SACRIFICING GROUP VARIABLES

In order to allow the customized sorting variables to work expectedly, the group variables in PROC REPORT should be paired with the sorting variables in such a way that each individual value of the group variables should be associated with the same values of sorting variables. In the previous section the two formats L1f. and L2f. are used to assign the values of the corresponding sorting variables so that this requirement is satisfied.

DISCUSSION AND CONCLUSION

Sorting is a common and important technique for data displaying and summarization. However, sorting has many unlisted facets therefore there are many approaches in sorting to make the programming more effective in terms of achieving the goal of informative data reporting. Not only an informative reporting requires a cohesive order in displaying values, the imbedded hierarchical structure should also be considered when constructing such customized ordering variables.

Multilevel reporting has both well-defined calculations at each level as well as uniform approach of progressing from one level to the next one. This dynamic process makes it desirable for SAS programmers to develop the standard routine or macro module. Note the overall text variable C as described previously in the paper should consider utilization of indention or in-text formatting etc. to visually reflect the hierarchical structure among different levels.

Using standard conventions such as CDISC data structure (ADAE in our illustrations) facilitates communication of ideas. A standard data structure provides a common platform to understand, research and discuss. CDISC has evolved over the years with advanced features that provide root of thought for solution in certain circumstances or bridge over to more effective approaches in other situations.

REFERENCES

SAS Institute Inc. SAS Online Doc 9.4

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Haibin Shu
AccuClin Global Services
P.O. Box 1491
1000 W. Valley Rd
Southeastern, PA 19399-1491
haibin@accuclinglobal.com