**PhUSE 2008**

**Paper CS05**

# Creating adverse event tables using PROC SQL

Berber Snoeijer, OCS Biometric Support, the Netherlands

**ABSTRACT**

The creation of adverse event tables by body system, preferred term, severity and/or intensity can be difficult. This is because next to the number of events, the number and percentage of subjects experiencing an event has to be calculated per treatment and overall.
At OCS Biometric Support we have created a standard for creating these tables by using simple SQL statements. This diminishes the time required for creating the tables and executing the QC of these tables considerably. In this paper we will present the method used.

## INTRODUCTION

The creation of adverse event tables is regular practice in data analysis of clinical studies. In many cases more than three summary tables are dedicated to this subject alone. Usually counts have to be determined by body system, by preferred term, in total, by intensity, by relationship and by treatment or treatment day. Normally next to the number of events, the number of subjects experiencing an event and the percentage of subjects of the total population is requested as well. As one subject can experience more than one event with different intensities and with different relationships to the study drug counting can be very difficult. In our office we have experienced that using standard SQL procedures in a standard macro is very efficient and produces the right tables without erroneous counting.

## CREATING ADVERSE EVENT TABLES BY BODY SYSTEM AND PREFERRED TERM

### DATASET USED

As we have standardised the adverse event counting procedure, we should have a standard input dataset. Therefore the variables for body system (SOC), preferred term, intensity, relationship and treatment (day) should be standardised as well.
We use the following standard:

```
DATA aes;
   SET aes;
   Soc=soc;
   Prefterm=prefterm;
   Sever=severity;
   Relat=relationship;
   Cntby=treatment;
RUN;
```

### COUNTING

We have created one standard macro using six SQL statements, in which the number of events or the number of subjects are counted including totals.

The SQL statements perform the following:

Statement 1 counts the total number of events/subjects;
Statement 2 counts the total number of events/subjects by treatment;
Statement 3 counts the total number of events/subjects by body system;
Statement 4 counts the total number of events/subjects by body system and treatment;
Statement 5 counts the total number of events/ subjects by body system and preferred term;
Statement 6 counts the number of subjects by body system, preferred term and treatment.

NB The variable cntby will be filled with 1000 to represent all groups combined.

```
%MACRO cntae(ds=,prefix=,var=);
    PROC SQL;
       /* Count all var with at least one AE */
       CREATE TABLE &prefix.0 AS
          SELECT COUNT( &var ) AS tot, 1000 AS cntby, "TOTAL" AS soc_name
          FROM &ds;
       /* Count var by treatment and treatment day */
       CREATE TABLE &prefix.1 AS
          SELECT COUNT( &var) AS tot, cntby, "TOTAL" AS soc_name
          FROM &ds
          GROUP BY cntby;
       /* Count var by SOC */
       CREATE TABLE &prefix.2 AS
          SELECT COUNT( &var ) AS tot, soc_name, 1000 AS cntby
          FROM &ds
          GROUP BY soc_name;
       /* Count var by SOC and treatment */
       CREATE TABLE &prefix.3 AS
          SELECT COUNT( &var ) AS tot, soc_name, cntby
          FROM &ds
          GROUP BY soc_name, cntby;
       /* Count var by SOC and PT */
       CREATE TABLE &prefix.4 AS
          SELECT COUNT( &var ) AS tot, soc_name, pt_name, 1000 AS cntby
          FROM &ds
          GROUP BY soc_name, pt_name;
       /* Count var by SOC and treatment */
       CREATE TABLE &prefix.5 AS
          SELECT COUNT( &var ) AS tot, soc_name, pt_name, cntby
          FROM &ds
          GROUP BY soc_name, pt_name, cntby;
   QUIT;
%MEND cntae;
```

This macro is called twice to get counts of the number of events and counts of the number of subjects like this:

```
/* count subjects */
%cntae(ds=ae,prefix=sub,var=DISTINCT pt);
/* count events */
%cntae(ds=ae,prefix=ev,var=pt);
```

We use DISTINCT to count each subject only once in each category. If we do no use DISTINCT the total number of events is counted.
Running the above code will result in 12 intermediate datasets which are combined like this:

```
/* PUT all data together */
DATA sub;
   LENGTH soc_name $200;
   SET sub0 sub1 sub2 sub3 sub4 sub5;
RUN;

PROC SORT DATA=sub;
   BY cntby soc_name pt_name;
RUN;

DATA ev;
   LENGTH soc_name $200;
   SET ev0 ev1 ev2 ev3 ev4 ev5;
RUN;

PROC SORT DATA=ev;
   BY cntby soc_name pt_name;
RUN;
```

```
/* merge subjects and events */
DATA tot;
   MERGE sub (RENAME=(tot=sub)) ev (RENAME=(tot=ev));
   BY cntby soc_name pt_name;
RUN;
```

The dataset 'tot' now contains all counts including the number of events and the number of subjects by preferred term, body system, treatment and overall. The next step is to get the total number of subjects per treatment (i.e. cntby). In our example we will assume that these totals are stored in macro variables. But it may also be possible that they are stored in a second dataset which is merged with our dataset 'tot'.
In our example we counted the total number of subjects for four treatments (as defined in cvntby) and we have an overall count stored in macro variables num1 to num4 and nrsub respectively. We calculate the percentages and we create a new text variable concatenating all data that we want to present. In this case this is the number of events, the number of subjects and, between brackets, the corresponding percentage.

```
DATA tot;
   SET tot;
      IF cntby=1 THEN perc=sub/&num1;          /* trt 1 */
            ELSE IF cntby=2 THEN perc=sub/&num2;
            ELSE IF cntby=3 THEN perc=sub/&num3;
            ELSE IF cntby=4 THEN perc=sub/&num4;
      IF cntby=1000 THEN perc=sub/&nrsub;
      perc=ROUND(perc*100,.1);
      txtval=PUT(ev,3.)||' '||PUT(sub,2.)||' ('||PUT(perc,5.1)||')';
RUN;
```

This dataset can be transposed by cntby for presentation purposes.
If you would like to create this table for related events only, you can simply create an input dataset containing related events only.

## CREATING ADVERSE EVENT TABLES BY SEVERITY OR RELATIONSHIP

### COUNTING

The steps as presented above can be performed for counting by intensity or relationship as well. For this purpose, we have created another standard macro which is similar to the one presented above. However, one variable is added to this standard counting macro. Our next example will describe counting by severity, but the same principle applies to counting by relationship. The macro used includes six SQL statements as well.

The SQL statements perform the following:

Statement 1 counts the total number of events/ subjects by treatment;
Statement 2 counts the total number of events/subjects by treatment and severity;
Statement 3 counts the total number of events/subjects by body system and treatment;
Statement 4 counts the total number of events/subjects by body system, severity and treatment;
Statement 5 counts the total number of events/ subjects by body system, preferred term and treatment;
Statement 6 counts the number of subjects by body system, preferred term, treatment and severity.

```
%MACRO cntae(ds=,prefix=,var=);
   PROC SQL;
      /* Count var by treatment: cntby */
       CREATE TABLE &prefix.0 AS
          SELECT COUNT( &var) AS tot, cntby, "TOTAL" AS soc_name, "99" AS sever
          FROM &ds
          GROUP BY cntby;
      /* count by treatment, treatment day and severity */
      CREATE TABLE &prefix.1 AS
         SELECT COUNT( &var) AS tot, cntby, "TOTAL" AS soc_name, sever
         FROM &ds
         GROUP BY cntby,sever;
      /* Count by SOC and treatment */
      CREATE TABLE &prefix.2 AS
         SELECT COUNT( &var ) AS tot, soc_name, cntby, "99" AS sever
         FROM &ds
         GROUP BY soc_name, cntby;
      /* Count by SOC treatment and severity*/
      CREATE TABLE &prefix.3 AS
         SELECT COUNT( &var ) AS tot, soc_name, cntby, sever
         FROM &ds
         GROUP BY soc_name, cntby, sever;
      /* Count var by SOC and PT */
      CREATE TABLE &prefix.4 AS
          SELECT COUNT( &var ) AS tot, soc_name, pt_name, cntby, "99" AS sever
          FROM &ds
          GROUP BY soc_name, pt_name, cntby;
      CREATE TABLE &prefix.5 AS
           SELECT COUNT( &var ) AS tot, soc_name, pt_name, cntby, sever
           FROM &ds
          GROUP BY soc_name, pt_name, cntby, sever;
   QUIT;
%MEND cntae;
```

Combining the intermediate output datasets and to process them for output is done in a similar way as shown in the example of creating adverse event tables by body system and preferred term alone. Be aware that you should take severity into account as well when combining the datasets.

**CONCLUSION**

In this paper a simple and efficient method is presented to create adverse event tables. With these standard macros creating such tables is much more efficient and less time consuming.

**CONTACT INFORMATION**

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


| | | |
|---|---|---|
| Author Name | : | Berber Snoeijer |
| Company | : | OCS Biometric Support |
| Address | : | Schipholweg 78<br>2316 XD Leiden<br>The Netherlands |
| Work Phone | : | +31 (0)71 5721828 |
| Facsimile | : | +31 (0)71 5765040 |
| E-mail | : | info@ocs-biometricsupport.com |
| Web | : | www.ocs-biometricsupport.com |