Integrating Information – from Character Strings to SAS Datasets

Kamila Duniec, Oxford Pharmaceutical Sciences, Warsaw, Poland

What do you do when every bit of subject information is in a separate variable, record or dataset? Of course you concatenate, merge, set, append etc. Simple as it sounds, may become troublesome when the study design or data structure is quite complex. Below are a number of scenarios typical when handling pharmaceutical data.

CHARACTER STRING CONCATENATION

SEVERAL SUBJECTS – SEVERAL VISITS – ONE VARIABLE – SEVERAL RECORDS

The usual way to concatenate character strings is to bring together the values of several variables from the same record. For example, COMMENT1 and COMMENT2 can be joined into ALL_COMMENT like this:

```sas
data dsset2;
  set dsset;
  length all_comm $100;
  all_comm=trim(comment1)!!', '!!trim(comment2);
run;
```

ONE SUBJECT – ONE VARIABLE – SEVERAL RECORDS

What if you have the necessary data in subsequent records? There is a nice PROC SQL query to deal with such situations.

```sql
proc sql;
  select adverse into :list separated by ', ' from dsset;
quit;
```

SEVERAL SUBJECTS – SEVERAL VISITS – ONE VARIABLE – SEVERAL RECORDS

The problem is however that we usually deal with datasets containing records for more than one subject. PROC SQL becomes then useless, as it lists all records for the given variable. We can, however, manage with two simple data steps.

```sas
proc sort data=dsset out=dsset;
  by sid visit date ascending adverse;
run;

data list1;
  set dsset;
  length list $200;
  list=trim('list');
  n=0;
  if first.visit then do;
    list=trim(adverse)!!', '!!list;
    n=1;
    end;
  else do;
    list=compress(adverse)!!', '!!list;
  end;
run;

data list2;
  set dsset;
  length list $200;
  list=trim('list');
  n=0;
  if first.visit then do;
    list=trim(adverse)!!', '!!list;
    n=1;
    end;
  else do;
    list=compress(adverse)!!', '!!list;
  end;
run;
```

COMBINING SAS DATA SETS

SIMPLE MERGING

The below datasets (RESDRG and TREAT) are an example of the most common situation where merging is needed. In these datasets there are two variables in common (SID and DOSE), and therefore both have to be used as ‘by’ variables.

```sas
proc sort data=restrr; by sid dose;
run;

data restrr; merge treat resdrg;
  by sid dose;
run;
```

WHERE PROC SQL WILL NOT DO

Protocol compliance dataset (COMPL) contains patient information recorded at subsequent visits (VI and V2). Treatment dataset (TREAT) gives data connected with dosing events (D1 and D2). If we want to integrate information from the corresponding visits and dosing events (VI with information from D1 and not with D2), we cannot use PROC SQL joining, only data step merging.

```sas
proc sort data=restrr; by sid dose;
run;

data list1;
  set dsset;
  length list $200;
  list=trim('list');
run;

data list2;
  set dsset;
  length list $200;
  list=trim('list');
run;

data list3;
  set dsset;
  length list $200;
  list=trim('list');
run;

data list4;
  set dsset;
  length list $200;
  list=trim('list');
run;
```

PROC SQL

```sql
proc sql;
  create table trtcomp as
    select t.sid, t.dose,
    from treat a full join resdrg b
    on a.sid = b.sid and
    a.dose = b.dose;
quit;
```

PROC FORMAT

In case of mapping treatment dataset TRT for a study with a complicated design of treatments, a table of drug treatments, codes and names can be created and used as a base for drug formats. Below a simplified example is given, in which formats grp and dname are created to define variables DRGGRP and DRGNAME basing on variable GROUP – a concatenated information about treatment combination (TRT=TRT+TRT+TRT).

```sas
data grp_fnt (keep=fntname start label type);
  set fnt_table;
  type = "t";
  fntname = "grp";
  start = trt_code;
  label = dgroup;
run;

data dname (keep=fntname start label type);
  set dname;
  type = "t";
  fntname = "dname";
  start = trt_code;
  label = dgroup;
run;
```

CONCLUSION

SAS® offers a wide range of tools to integrate information at various levels. Beside the classical concatenating and merging cased this paper also presents solutions to possible scenarios in the context of clinical data where character concatenation and data set merging is required. Emphasis is put on where DATA STEP and PROC SQL cannot be used interchangeably. Finally, format tables and PROC FORMAT is shown as a convenient way to cope with situations where numerous combinations of variables have to be concatenated and used to code new variables.