Integrating Information - from Character Strings to SAS Datasets

Kamila Duniec, Oxford Pharmaceutical Sciences, Warsaw, Poland
kamila.duniec@ops-web.com

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 VARIABLES - ONE RECORD

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_COMM like this:

```
data dset2;
   set dset;
   length all_comm $100;
   all_comm=trim(comment1)!!', '!!trim(comment2);
run;
```

SID	VISIT	COMMENT1	COMMENT2
1001	V1	Patient missed 1 tablet	Patient's wife not supportive
1001	V2	Patient came to visit 2 days later	Patient forgot to take study drug during holidays

SID	VISIT	COMMENT1	COMMENT2	ALL_COMM
1001	V1	Patient missed 1 tablet	Patient's wife not supportive	Patient missed 1 tablet, Patient's wife not supportive
1001	V2	Patient came to visit 2 days later	Patient forgot to take study drug during holidays	Patient came to visit 2 days later, Patient forgot to take study drug during holidays

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. It makes a macro variable LIST and in addition counts how many records have been concatenated (SQLOBS). Note that there is a length limit to the created character string of 32767 signs!

```
proc sql ;
   select  adverse
     into :LIST separated by ', '
     from  dset ;
quit;


data dset2;
   set dset;
   length clist $200;
   list=symget('list');
   num=&SQLOBS;
run;
```

SID	ADVERSE
1001	leg pain
1001	constipation
1001	tremor
1001	hot flushes
1001	cold

SID	ADVERSE	LIST	NUM
1001	constipation	constipation, tremor, headache, laryngitis, cold	5
1001	tremor	constipation, tremor, headache, laryngitis, cold	5
1001	headache	constipation, tremor, headache, laryngitis, cold	5
1001	laryngitis	constipation, tremor, headache, laryngitis, cold	5
1001	cold	constipation, tremor, headache, laryngitis, cold	5

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.

SID	VISDATE	ADVERSE
1001	**10052005**	**constipation**
1001	**10052005**	**tremor**
1001	21052005	headache
1002	16042005	laryngitis
1003	14032005	cold
1003	**25042005**	**headache**
1003	**25042005**	**insomnia**
1004	11052005	hot flushes
1005	05042005	leg pain

SID	VISDATE	ADVERSE	LIST
1001	**10052005**	constipation	constipation, tremor
1001	**10052005**	tremor	constipation, tremor
1001	21052005	headache	headache
1002	16042005	laryngitis	laryngitis
1003	14032005	cold	cold
1003	**25042005**	headache	headache, insomnia
1003	**25042005**	insomnia	headache, insomnia
1004	11052005	hot flushes	hot flushes
1005	05042005	leg pain	leg pain

```
proc sort data=dset out=dset;
   by sid visdate descending adverse;
run;

data list1;
   set dset;
   length clist $200;
   by sid visdate;
   retain clist '';
   m=0;

   if first.visdate then do;
      clist=adverse;
   end;
   else do;
      clist=compress(adverse)!!', '!!clist;
      m+1;
   end;
run;

proc sort data=list1 out=list2 ;
   by sid visdate descending m;
run;

data list3;
   set list2;
   by sid visdate  descending m;

   if first.visdate then do;
      retain list;
      list=clist;
   end;

   drop clist m;
run;
```

CONCLUSION

SAS® offers a wide range of tools to integrate information at various levels. Beside the classical concatenating and merging codes this paper also presents solutions to possible scenarios in the context of clinical data where character concatenation or 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.

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.

DATA STEP:

```
proc sort data=restrt;
     by sid dose;
run;

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

SID	DOSE	DRGTAK	TIMETAK
1001	D1	1	3
1001	D2	2	
1002	D1	2	
1002	D2	1	2
1002	D2	1	4
1003	D1	2	
1003	D2	2	
1004	D1	1	2.5
1004	D2	2	

SID	DOSE	DOSEDAT	DOSEGRP
1001	D1	20032005	1
1001	D2	27032005	1
1002	D1	10022005	2
1002	D2	18022005	2
1003	D1	12032005	2
1003	D2	19032005	2
1004	D1	30012005	1
1004	D2	6022005	1

PROC SQL:

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

... WHERE PROC SQL WILL NOT DO

Protocol compliance dataset (COMPL) contains patient information recorded at subsequent visits (V1 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 (V1 with information from D1 and not with D2), we cannot use PROC SQL joining, only data step merging.

SID	VISIT	COMPL
1001	V1	2
1001	V2	2
1002	V1	1
1002	V2	1
1003	V1	1
1003	V2	1
1004	V1	1
1004	V2	2

SID	DOSE	DOSEDAT	DOSEGRP
1001	D1	20032005	1
1001	D2	27032005	1
1002	D1	10022005	2
1002	D2	18022005	2
1003	D1	12032005	2
1003	D2	19032005	2
1004	D1	30012005	1
1004	D2	6022005	1

PROC SQL:

```
proc sql;
   create table trtcomp as
   select t.sid, t.dose,
t.dosegrp,
c.visit, c.compl
   from treat t, compl c
   where t.sid = c.sid;
quit;
```

SID	DOSE	DOSEGRP	VISIT	COMPL
1001	D2	1	V1	2
1001	D2	1	V2	2
1001	D1	1	V1	2
1001	D1	1	V2	2
1002	D1	2	V1	1
1002	D1	2	V2	1
1002	D2	2	V1	1
1002	D2	2	V2	1
1003	D2	2	V1	1
1003	D2	2	V2	1
1003	D1	2	V1	1
1003	D1	2	V2	1
1004	D1	1	V1	1
1004	D1	1	V2	2
1004	D2	1	V1	1
1004	D2	1	V2	2

DATA STEP:

```
%macro sort(set, sort);
proc sort data=data.&set out=&set;
   by sid &sort;
run;
%mend;
%sort(compl, visit);
%sort(treat, dose);

data trtcomp2;
   merge treat (keep=sid dose dosegrp)
         compl;
   by sid;
run;
```

SID	DOSE	DOSEGRP	VISIT	COMPL
1001	D1	1	V1	2
1001	D2	1	V2	2
1002	D1	2	V1	1
1002	D2	2	V2	1
1003	D1	2	V1	1
1003	D2	2	V2	1
1004	D1	1	V1	1
1004	D2	1	V2	2

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 (trt1+trt2+trt3).

```
data fmt_table;
input trt_code $ drg_group $  drg_name
$15.;
CARDS;
111 AA ACTIVE 1
112 AB ACTIVE 2
121 BA ACTIVE 3
211 CA Placebo 1
212 CB Placebo 2
221 DA Placebo 3
;
run;
```

TRT_CODE	DRG_GROUP	DRG_NAME
111	AA	ACTIVE 1
112	AB	ACTIVE 2
121	BA	ACTIVE 3
122	BB	ACTIVE 4
211	CA	Placebo 1
212	CB	Placebo 2
221	DA	Placebo 3
222	DB	Placebo 4

TRT1	TRT2	TRT3	GROUP	DRGGRP	DRGNAME
1	1	2	112	AB	ACTIVE 2

```
data grp_fmt (keep=fmtname start label type);
   set fmt_table;
   type ='c';
   fmtname ='grp';
   start=trt_code;
   label=drg_group;
run;

data dname (keep=fmtname start label type);
   set fmt_table;
   type ='c';
   fmtname ='dname';
   start=trt_code;
   label=drg_name;
run;

proc format cntlin=grp_fmt;
run;

proc format cntlin=dname;
run;

data trt;
   trt1='1';
   trt2='1';
   trt3='2';

   group=left(compress(trt1))!!(compress(trt2))
!!(compress(trt3));

   drggrp = put(group, $grp. );
   drgname = put(group, $dname.);
run;
```