Paper 174-2012

SAS® Macros to Transpose SDTM Data Sets Automatically

Chunmao Wang, National Institutes of Health, Bethesda, MD, United States

ABSTRACT

In pharmaceutical industry, when producing analysis datasets from clinical data, usually it is necessary to transpose variables for lots of domains. Individually manipulating each domain data would be time-consuming and error-prone. However, thanks to CDISC, the data structures are standardized so that it is possible to use macros to handle SDTM data sets automatically. Here we present a set of macros to transpose multiple columns for some domains. It not only saves programmers time but also improves data quality.

INTRODUCTION

The purpose of CDISC/SDTM is to use a standardized structure to store source data collected during a clinical trial. However, in the pharmaceutical industry, the clinical trials are much diverse. The trade-off of SDTM is to move non-standardized data into several standardized columns, while listing the specific properties as rows. From tabulation viewpoint, the data are pretty much standardized; however, the structure of the data is not an easy one to be analyzed straightforward, and is not ready for tabulating and listing. Transposing the columns to rows is usually required when generating analysis datasets, creating tables and listings.

In SDTM, all “Findings” domains, such as EG, IE, LB, PE, QS, SC, VS, DA, MB, MC, PC, PP, and FA, have similar variables: --TEST, --TESTCD, and –STRESC, which can be used for transposition to produce analysis datasets, tables and listings. Other domains, such as SU, have --trtcd and --trt or similar columns, could also be used for transposition.

EXAMPLE

Here we use an example throughout the paper for illustration. We used NIDA provided public shared data (Clinical Trials Network (CTN) Data Share), in which the subject’s information has been de-identified. Subject Characteristics (SC) domain is one record per reported characteristic per subject. A subject can have several characteristics collected during the course of a trial. The value is a verbatim name of the test or examination used to obtain the measurement or finding. At each row is a variable with character result/finding in standard format for the corresponding characteristic. See Table 1. for an example.

USUBJID	SCTESTCD	SCTEST	SCSTRESC
Unique Subject Identifier	*Subject Characteristic Short Name*	*Subject Characteristic*	*Character Result/Finding in Std Format*
10_001_001	CELIG	CONTINUES TO MEET ELIGIBILITY CRITERIA	YES
10_001_001	EDUCYRS	EDUCATION COMPLETED	8
10_001_001	INCAGE	INCLUSION/EXCLUSION AGE	18-21
10_001_001	INCSEX	INCLUSION/EXCLUSION GENDER	FEMALE
10_001_001	MARITAL	MARITAL STATUS	NEVER MARRIED
10_001_002	CELIG	CONTINUES TO MEET ELIGIBILITY CRITERIA	NO
10_001_002	EDUCYRS	EDUCATION COMPLETED	10
10_001_002	INCAGE	INCLUSION/EXCLUSION AGE	18-21
10_001_002	INCSEX	INCLUSION/EXCLUSION GENDER	MALE
10_001_002	MARITAL	MARITAL STATUS	NEVER MARRIED

Table 1. An example of Subject Characteristics (SC) domain extracted from clinical study

To feed this domain data for statistical analysis and listing, it is necessary to transpose the data appropriately. The transposed data would be something like Table 2.

USUBJID	CELIG	EDUCYRS	INCAGE	INCSEX	MARITAL
Unique Subject Identifier	*Continues To Meet Eligibility Criteria*	*Education Completed*	*Inclusion/ Exclusion Age*	*Inclusion/ Exclusion Gender*	*Marital Status*
10_001_001	YES	8	18-21	FEMALE	NEVER MARRIED

10_001_002	NO	10	18-21	MALE	NEVER MARRIED

Table 2. Transposed table

Simply, we could use PROC TRANSPOSE:

```
proc transpose data=sc out=sc1;
    by usubjid;
    id sctest;
    var scstresc;
run;
```

This procedure is easy to use; however, it is not flexible and may even produce incorrect results if there are any missing values (see Shostak (2005) for a detailed discussion). Take a look at the first row of Table 1 and Table 2: the bold uppercase words are for variable names, and the italic phases are for the corresponding labels. PROC TRANSPOSE cannot pass SCTEST into labels for transposed data.

On contrary, DATA step is much more flexible. In the following code, the first data step adds a variable “n” to the dataset, which is used in the second data step. The second data step adds the labels and transposes the data.

```
data sc2;
    set sc;
    by usubjid;
    if first.usubjid then n=0;
    n+1;
run;

data sc3;
    set sc2;
    by usubjid;

    length CELIG EDUCYRS INCAGE INCSEX MARITAL $ 200;
    retain CELIG EDUCYRS INCAGE INCSEX MARITAL "";
    array byvars {5} CELIG EDUCYRS INCAGE INCSEX MARITAL;

    label CELIG="Continues To Meet Eligibility Criteria";
    label EDUCYRS="Education Completed";
    label INCAGE="Inclusion/Exclusion Age";
    label INCSEX="Inclusion/Exclusion Gender";
    label MARITAL="Marital Status";

    if first.usubjid then
        do i = 1 to 5;
            byvars{i} = "";
        end;

    byvars{n} = scstresc;

    if last.usubjid;
    drop n i SCSTRESC SCTEST SCTESTCD;
run;
```

Though it works fine, you may notice that it looks too complicated and involves a lot of typing, which in turn may introduce more errors.

However, with careful programming, macros could be written to handle this purpose nicely and make the transpose work easily. In this case, a single line could work:

```
%sdtmtran(indata=sc, outdata=sc4, byvar=sctestcd, bylabel=sctest, byvalue=scstresc);
```

How the macros work

Let us look at the input parameters. We need to define the following parameters: input and output datasets, the variable to be transposed, the variable which holds values for labels, and the corresponding variable for values. SCTEST and SCTESTCD list characteristics; while the latter is used for short name, which is suitable for variable names, the former is longer and better for variable labels.

```
%macro sdtmtran(indata=,outdata=,byvar=,bylabel=,byvalue=);
```

First, we need to get unique byvar values, using PROC SORT:

```
    proc sort data=&indata nodupkey out=invar;
        by &byvar;
    run;
```

We save variable names and labels to macro variables:

```
    data _null_;
        set invar end=eof;
        call symput("byvar"||put(_n_,1.), trim(&byvar));
        call symput("bylabel"||put(_n_,1.), trim(&bylabel));
        if eof then call symput("byvar"||"N", put(_n_,1.));
     run;
```

and use PROC SQL to save all variable names to one variable, which will be used for array:

```
    proc sql noprint;
        select &byvar into :tranname
            separated by ' '
            from invar;
    quit;
```

We add a tag, to the input dataset, for each observation, matching with byvar macro variables:

```
    data temp;
        set &indata;
        %do i=1 %to &byvarN.;
            if &byvar="&&byvar&i" then n=&i;
        %end;
        drop _id;
    run;
```

We also need to save variables' length to macro variables.

To achieve that, we first use PROC CONTENTS to output all variables' information, and then use DATA _NULL_ to save the length to each macro variable:

```
    proc contents data=temp out=outtmp noprint;
    run;
    data _null_;
        set outtmp;
        call symput(trim(name)||'len',trim(length));
    run;
```

Here comes the core part: we use a nested macro to do the transpose work:

```
%macro trans(out=&outdata,in=temp);
    data &out;
        set &in;
        by usubjid;

        length &tranname $ &&&byvalue.len.;
        retain &tranname "";
        array byvars {&byvarN.} &tranname;

        %* Add label for each transposed variable;
        %do j=1 %to &byvarN.;
            label &&byvar&j="&&bylabel&j";
        %end;
```

```
        %* Initial the transposed variable;
        if first.usubjid then
            do i = 1 to &byvarN.;
                byvars{i} = "";
            end;

        %* Assign the transposed variable;
        byvars{n} = &byvalue;

        %* Only output all transposed variables when reach the last row of a subject;
        if last.usubjid;
    run;
%mend trans;
```

How to use the macros

The macros work for several domains.

For example: IE domain:

```
%sdtmtran(indata=ie,outdata=adie,byvar=ietestcd,bylabel=ietest,byvalue=iestresc);
```

For Example: SU domain (though a little preparation and a macro are necessary):

First, we use a macro to split the data into three parts:

```
%macro splitdata(outdata=,cond=,prefix=,suffix=);
    data &outdata;
        set su;
        where sucat="DRUG/ALCOHOL USE" and suevlint="&cond";
        length sutrtcd $32 sutrt $200;
        sutrtcd="&prefix"||compress(translate(trim(sutrt),"", "(", "",")",
                "","/"));
        sutrt="&prefix "||trim(sutrt)||" &suffix";
        drop sucat suevlint SUSEQ;
    run;
%mend splitdata;
%splitdata(outdata=tempp30d, cond=-P30D,prefix=P30D,suffix=Duration);
%splitdata(outdata=templife, cond=,     prefix=LIFE,suffix=Duration);
%splitdata(outdata=temproute,cond=-P30D,prefix=,    suffix=Route);
```

Then we use the macro to transpose the data:

```
%sdtmtran(indata=tempp30d,outdata=su_use_p30d,byvar=sutrtcd,bylabel=sutrt,byvalue=sudu
r);
%sdtmtran(indata=templife,outdata=su_use_life,byvar=sutrtcd,bylabel=sutrt,byvalue=sudu
r);
%sdtmtran(indata=temproute,outdata=su_use_route,byvar=sutrtcd,bylabel=sutrt,byvalue=su
route);
```

Finally we merge them:

```
proc sort data=su_use_p30d;
    by usubjid;
run;
proc sort data=su_use_life;
    by usubjid;
run;
proc sort data=su_use_route;
    by usubjid;
run;
data adsu(label="DRUG/ALCOHOL USE");
    merge su_use_p30d su_use_life su_use_route;
    by usubjid;
run;
```

CONCLUSION

This paper presents a set of macros to transpose SDTM datasets automatically with reliability and accuracy. The macros not only provide an efficient and solid way to transpose SDTM datasets, but can also be used for other datasets with minimal modification.

REFERENCES

Clinical Data Interchange Standards Consortium, *http://www.cdisc.org/sdtm*

Clinical Trials Network (CTN) Data Share, *https://secure.emmes.com/ctnweb/*

Jack Shostak (2005). SAS Programming in the Pharmaceutical Industry. *SAS Publishing*.

ACKNOWLEDGMENTS

The author thanks Shruti Deshmukh, Gang Chen, Jiang Xu , and Weiya Zhang for their kind assistance.

CONTACT INFORMATION

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

Name:	Chunmao Wang
Enterprise:	Synteract, Inc.
Address:	2801 Slater Road, Suite 100
City, State ZIP:	Morrisville, NC 27560
Work Phone:	919.462.4809
E-mail:	wangcm4@hotmail.com

APPENDIX

The whole codes of the transpose macros:

```
/*3456789C123456789H123456789U123456789N123456789M123456789A12345678901234567 8*/
/* Program: sdtmtran.sas                                                      */
/* Purpose: Transpose SDTM datasets                                           */
/* Author : Chunmao Wang                                                      */
/* Date   : 10-Nov-2011                                                       */
/* Contact: wangcm4@hotmail.com                                               */
/*3456789C123456789H123456789U123456789N123456789M123456789A12345678901234567 8*/

%macro sdtmtran(indata=,outdata=,byvar=,bylabel=,byvalue=);

    %* add index _id to keep its original order;
    data &indata;
        set &indata;
        _id=_n_;
    run;

    %* get unique byvar values in its original order;
    proc sort data=&indata nodupkey out=invar;
        by &byvar;
    run;
    proc sort data=invar out=invar(keep=&byvar &bylabel);
        by _id;
    run;
    %* and save var names and labels to macro variables;
    data _null_;
        set invar end=eof;
        if _n_<10 then do;
```

```
            call symput("byvar"||put(_n_,1.), trim(&byvar));
            call symput("bylabel"||put(_n_,1.), trim(&bylabel));
        end;
        else do;
            call symput("byvar"||put(_n_,2.), trim(&byvar));
            call symput("bylabel"||put(_n_,2.), trim(&bylabel));
        end;
        if eof then call symput("byvar"||"N", put(_n_,z2.));
    run;
    %* and save all var name to one variable which will be used for array;
    proc sql noprint;
        select &byvar into :tranname
            separated by ' '
            from invar;
    quit;

    %* give each obs a tag matching with byvar macro variables;
    data temp;
        set &indata;
        %do i=1 %to &byvarN.;
            if &byvar="&&byvar&i" then n=&i;
        %end;
        drop _id;
    run;

    %* save variables length and label to macro variables;
    proc contents data=temp out=outtmp noprint;
    run;
    data _null_;
        set outtmp;
        call symput(trim(name)||'len',trim(length));
        call symput(trim(name)||'label',trim(label));
    run;

    %* transpose for each variable and append to temp1;
    %trans(out=temp1,in=temp);

    %* return outdata and clean up temp data;
    data &outdata(label="&prefix &&&bylabel.label");
        set temp1;
        drop &byvar &bylabel &byvalue n i;
    run;
    proc datasets lib=work;
        delete temp temp1 invar outtmp;
        quit;
    run;
%mend sdtmtran;

%macro trans(out=,in=);
    data &out;
        set &in;
        by usubjid;

        length &tranname $ &&&byvalue.len.;
        retain &tranname "";
        array byvars {&byvarN.} &tranname;

        %do j=1 %to &byvarN.;
            label &&byvar&j="&&bylabel&j";
        %end;

        if first.usubjid then
            do i = 1 to &byvarN.;
```

```
                byvars{i} = "";
            end;

        byvars{n} = &byvalue;

        if last.usubjid;
    run;
%mend trans;
```