PhUSE 2014

PP07

FLIPFLOP the data: To check sustained time-point Occurrences

Rucha Landge, inventiv International Pharma Services Pvt. Ltd., Pune, India

ABSTRACT

Frequently in dataset creation as well as in reporting there is always a need to restructure data from columns into rows and rows to column. What other method than the conventional PROC TRANSPOSE will be used to flip-flop the data. PROC TRANSPOSE is a simple method and very precise to obtain the desired results.

Keeping aside the conventional usage of PROC TRANSPOSE to obtain single observation from multiple observations, let us know some more benefits of this procedure.

We frequently come across instances in CLINICAL REPORTING to check consecutive occurrences of events across time-points like days, visits. These checks are very significant while creating summary reports. This paper will elaborate on different perspective to use PROC TRANSPOSE.

INTRODUCTION

While working on clinical data there are lot of instances where in we have to check to report the summary for consecutive time points for e.g. consecutive missed visits, consecutive abnormalities, etc. Usual approach of any programmer is to use data step to obtain the desired results. PROC TRANSPOSE procedure is flexible enough to generate an appropriate structure of data to identify such instances.

Consider we have to create a summary report to show sustained elevations in VITAL SIGNS like systolic blood pressure/diastolic blood pressure depending on the definition of the elevation. First thing that will click our mind is to code data step. This paper will provide an alternative and a different approach to such a scenario.

The purpose of this paper is to provide a code which will skip the data steps required for examining consecutive behavior of specified events with respect to time points using PROC TRANSPOSE. This approach will not only be precise but also will allow us to check data horizontally rather than checking vertically in data step. This approach will be useful in checking time-points for VITAL SIGNS / LABORATORY ANALYSES and many more datasets and their safety reporting.

Industry starters are not very familiar or may find working with RETAIN/ FIRST. / LAST. concepts. PROC TRANSPOSE on the other hand is simple procedure to code and allows to check the data horizontally. Another advantage of using PROC TRANSPOSE is that the horizontal structure of data provides visual checks to be performed on data.

Using PROC TRANSPOSE will not only reduce the number of coding steps for generating this output but also will reduce on the execution time for the same.

PROC TRANSPOSE

Data transposition is the process of changing the orientation of the data from a normalized structure to a non-normalized structure or vice versa. Normalized data may also be described as “stacked,” “vertical,” or “tall and skinny,” while non-normalized data are often called “flat,” “wide,” or “short and fat.” Transposition of data is also called as Flip-flopping of data.

De-normalization of data is needed when a statistical procedure requires that the information to be analyzed must be on the same observation. Let us see the code for PROC TRANSPOSE and an example of horizontal data transposing to vertical data.

```
**** INPUT SAMPLE NORMALIZED SYSTOLIC BLOOD PRESSURE VALUES.
**** SUBJECT = PATIENT NUMBER, VISIT = VISIT NUMBER,
**** SBP = SYSTOLIC BLOOD PRESSURE.;


data sbp;
input subject $ visit sbp;
datalines;
101 1 160
101 3 140
101 4 130
101 5 120
202 1 141
202 3 161
202 4 171
202 5 181
;
run;
```

	subject	visit	sbp
1	101	1	160
2	101	3	140
3	101	4	130
4	101	5	120
5	202	1	141
6	202	3	161
7	202	4	171
8	202	5	181

Fig. 1

The above data is a snapshot of the datalines created.

```
**** TRANSPOSE THE NORMALIZED SBP VALUES TO A FLAT STRUCTURE.;

PROC TRANSPOSE data = sbp
               out = sbpflat
            prefix = VISIT;
   by subject;
   id visit;
   var sbp;
run;
```

	subject	_NAME_	VISIT1	VISIT3	VISIT4	VISIT5
1	101	sbp	160	140	130	120
2	202	sbp	141	161	171	181

Fig. 2

The systolic blood pressures value of each subject has been transposed from row-based data to column-based data. Wide data is easy to visualize and check, than checking data horizontally record by record using RETAIN/LAG statements. In this paper we will have a look into wide transposed data and few examples where this kind of data is more often used in clinical industry.

EXAMPLE- 1 :

Laboratory analyses data has been used in the following example. The data is collected at planned visits for the LAB tests. Consider we have to identify the patients who have missed two or more than two consecutive planned visits for LABS . This type of analyses is useful to report the summary of patients having consecutive missed visits, also we can report these patients for Protocol Violation .

Code to obtain the results is as given below :

```
**** DUMMY VISIT DATA CREATED TO OBTAIN ALL THE PLANNED VISITS;

proc sort data =visit out = lab_visit(keep = patid visitnum);
       by patid visitnum;
       where visitnum in  (1,3,4,6,10,14,19,26,32);
run;
```

There can be a scenario where a patient does not have planned visits collected in LABS other than having missing numeric result value. Hence we merge dummy visits data to get all those missing planned visits not present in the dataset.

```
**** LABS DATA;

proc sort data = labs(keep = patid visitnum parameter numeric_result character_result)
          out = lab1 nodupkey;
       by patid visitnum  ;
  where visitnum <801 ;
run;
```

Subset the LABS data for particular study interval data visits.

```
**** LABS AND VISIT DATA MERGED TO OBTAIB ALL PLANNED VISITS;

data lab2(keep = patid visitnum parameter numeric_result);
       merge lab_visit(in = a) lab1(in = b);
       by patid visitnum;
       if a;
run;
```

#	PATID	VISID	LAB_RESULT_TYPE	LAB_NUM_RESULT	MAXVIS
1	2303	1.00	CN	37	22.00
2	2303	3.00	CN	41	22.00
3	2303	4.00	CN	39	22.00
4	2303	6.00	CN	38	22.00
5	2303	10.00	CN	48	22.00
6	2303	14.00		.	22.00
7	2303	19.00		.	22.00

Fig. 3

LABS and VISIT data are merged to get all the planned visits in a single dataset. We can observe that V14 and V19 have missing lab_num_result.

```
**** MISSFLG VARIABLE CREATED FOR MISSING LAB NUMERIC RESULTS;

data lab5;
       set lab2;
       length missflg $1;
       retain missflg;
       by patid visitnum;
       if first.visitnum then missflg = '0';
       if lbrn = . then missflg = '1';
run;
```

#	PATID	VISID	LAB_RESULT_TYPE	_AB_NUM_RESULT	MAXVIS	DS_VISID	MISSFLG
1	2303	1.00	CN	37	22.00	24	0
2	2303	3.00	CN	41	22.00	24	0
3	2303	4.00	CN	39	22.00	24	0
4	2303	6.00	CN	38	22.00	24	0
5	2303	10.00	CN	48	22.00	24	0
6	2303	14.00		.	22.00	24	1
7	2303	19.00		.	22.00	24	1

Fig. 4

We flag the missing numeric_result to 1 (variable = missflg). In the snapshot DS_VISID is the disposition visid and MAXVIS is the maximum visit in the dataset. Hence any planned visit missed before V22 / v24 and having missing lab_num_result is flagged as 1.

```
**** DATA TRANSPOSED TO HAVE FLAT DATA;

PROC TRANSPOSE data = lab5 out= lab6 prefix = v;
  by patid ;
  id visitnum ;
  var missflg ;
run;
```

#	PATID	_NAME_	V1	V3	V4	V6	V10	V14	V19
1	2303	missflg	0	0	0	0	0	1	1

Fig. 4

Transpose data using visit as ID variable and missflg variable as the value to be transposed. Now we have the data in horizontal format and it easy to check whether a patients has consecutive missed visits. Horizontal data is visually easy to validate with the reports generated.

```
**** CONCATENATE DATA USING CATT FUNCTION ;

data lab7;
       set lab6;
       cat1 = catt(v1,v3,v4,v6,v10,v14,v19,v26,v32);
       if index(cat1,'11') > 0;
run;
```

#	PATID	_NAME_	V1	V3	V4	V6	V10	V14	V19	CAT1
1	2303	missflg	0	0	0	0	0	1	1	0000011..

Fig. 5

Concatenate the visit-numbers into a variable “cat”. Once the cat variable is formed we check using the index function if concatenated variable as consecutive “1” in the value present. Consecutive 1 present in the value of cat symbolizes consecutive missed time-points.

Same result can also be obtained using retain statement or few other methods. But this is the easiest way and a fewer lines of code is required to perform the same activity.

Let us have a look at another example where we have to check if the Pulse rate of a patient is treatment emergent and flag the treatment emergent record .

Example - 2 :

Consider Electrocardiogram data of Pulse Rate (PR) that is collected at PLANNED VISIT. We have to conclude if the PR results are treatment emergent and then report the number of patients having elevated PR and further derive the summary statistics.

A simpler and a minimalistic lines of code using PROC TRANSPOSE is a way to obtain the subjects having elevated PR and also deriving descriptive statistics for reporting procedure is preferable than conventional data step.

A dummy ECG data for ECG assessment PR is created using datalines.

```
**********DUMMY ECG DATA IS CREATED USING DATALINES AND FILTERED;

data ecg (keep = patid parameter tempvar2 ecg_normality visitnum seqid
numeric_result);
       set ecg1 ;
       tempvar2 = "v" ||strip(put(visitnum,best.)) || "s" || strip(put(seqid,best.));
       if ecg_normality = " " then ecg_normality = "M" ;
run;
```

#	PATID	VISIT_NUM	ECG_ASSESSMENT	EGNORM	TEMPVAR2
1	725	1.00	PR	N	v1s1
2	725	4.00	PR	N	v4s1
3	725	7.00	PR	H	v7s1
4	725	10.00	PR	N	v10s1
5	2255	1.00	PR	N	v1s1
6	2255	4.00	PR	N	v4s1
7	2255	7.00	PR	N	v7s1

Fig. 6

ECG data is created using datalines and tempvar2 variable is created to check the normality of the Pulse Rate at the first record at that particular visit.

```
*************DATA FILTERED FOR PULSE-RATE ECG RESULT AND BASELINE VISITS;

proc sort data = ecg
             out = basenormau;
       by   patid parameter;
       where parameter in ('PR') and 1 <= visitnum <= 3;
run;
```

#	PATID	VISIT_NUM	ECG_ASSESSMENT	EGNORM	TEMPVAR2
1	725	1.00	PR	N	v1s1
2	2255	1.00	PR	N	v1s1

Fig. 7

Basenormau dataset is filtered for the PULSE RATE ECG assessment and the respective baseline visits.

```
**** DATA TRANSPOSED;

PROC TRANSPOSE data = basenormau
                out = tbasenormau;
       by   patid parameter;
       var ecg_normality;
       id visitnum;
run;
```

#	PATID	ECG_ASSESSMENT	V1S1
1	725	PR	N
2	2255	PR	N

Fig. 8

The data is transposed to check the ECG normality vertically and checked igf the ecg_normality = “H” at baseline.

```
**** DATA PROCESSING;

data tbaseau;
set tbasenormau;
       val = catt( _1d00 , _2d00) ;
       if val in ('N','L','') then bshinormau = 0;
       else if val = "H" then bshinormau = 1;
keep   patid parameter val bshinormau;
run;
```

#	PATID	ECG_ASS...	BSHINORMAU	BSLONORMAU
1	725	PR	0	0
2	2255	PR	0	0

Fig. 9

SUBJID 725 and SUBJID 2255 both are “N” = Normal at the baseline .

```
proc sort data = ecg ;
       by   patid parameter;
run;

**** BASELINE DATA MERGED TO ORIGINAL DATA TO CHECK AT ALL CORRESPONDING VISITS;

data add_chgprvy;
   merge ecg  tbaseau;
   by   patid parameter;
   if parameter ='PR' then
   do;
    if ecg_normality = 'M' or  visitnum <= 3 then hiflag = .;
    else if 3 < visitnum <= 9 and bshinormau = 0 and ecg_normality = 'H' then hiflag = 1;
    else if 3 < visitnum <= 9 then hiflag = 0;
   end;
run;
```

#	PATID	VISIT_NUM	ECG_ASSESSMENT	EGNORM	TEMPVAR2	BSHINORMAU	BSLONORMAU	HIFLAG	LOFLAG
1	725	1.00	PR	N	v1s1	0	0	.	.
2	725	4.00	PR	N	v4s1	0	0	0	0
3	725	7.00	PR	H	v7s1	0	0	1	0
4	725	10.00	PR	N	v10s1	0	0	.	.
5	2255	1.00	PR	N	v1s1	0	0	.	.
6	2255	4.00	PR	N	v4s1	0	0	0	0
7	2255	7.00	PR	N	v7s1	0	0	0	0

Fig. 10

SUBJID 725 is normal at baseline and hence has BSHINORMAU = 0 present at all VISITS but has EGNORM = “H” at post-baseline visits and hence HIFLAG is flagged as 1. While SUBJID 2255 has BSHINORMAU = 0 but does not have any EGNORM = “H” value at post-baseline and is flagged as HIFLAG = 0 .

Very common method to derive TRT-emergent flag is in a data step using retain statement or LAG function. Most industry starters are not very familiar with these options but would prefer to use procedures in comparison to them in data step as this

requires fewer lines of code and is easy to debug. Sometimes using PROC TRANSPOSE may require initial or later processing but it is easy to debug .

COMPARISON OF AN EXAMPLE CODED BOTH IN DATA STEP AND USING PROC TRANSPOSE

Below line of code is to summarize the data and report the count of patients having vitals 95th percentile missed at 3 or more consecutive visits. We also have to check the difference of baseline visit result value to the corresponding visit result value .

Two types of code one in DATA step and the other using PROC TRANSPOSE is written to compare the easier and easy to debug method.

```
***************ALTERNATE METHOD TO PROC TRANSPOSE - DATA STEP;

data anadata1_1;
  set anadata1;
  by patid visid;
  retain baseval;
  if vsmaxblflg&phase = 1 then baseval = vsrn;
  if baseval ne . then chg = vsrn - baseval;
run;

data anadata2_1;
  set anadata1_1(where = (visid = int(visid)));
  by patid visid;
  previs = lag(visid);
  if chg > 5 and vspct >= 95 then tempflg = 1;
  if first.patid then previs = .;
  if first.subjid or visid - previs ne 1 then countvis = 1;
  else if visid - previs = 1  then countvis + 1;
  lag_countvis  = lag(countvis);
  if countvis = 1 or chg <= 5 or vspct < 95 then finalcounter = 0;
  if chg > 5 and vspct >= 95 and countvis-lag_countvis = 1 then finalcounter + 1;
  if finalcounter = 3 then flg = 1;
  keep patid visid previs countvis baseval chg vspct lag_countvis finalcounter flg tempflg;
run;
```

As we can see LAG / FIRST. has been used to obtain result. If we have to debug the code or handle particular condition the above line of code is time consuming and difficult for an industry starter to debug.

```
************************PROC TRANSPOSE METHOD;

proc transpose data = vitals out = vstrans prefix = v;
  by patid ;
  id visid ;
  var vsrn ;
  where 1<= visid <= 2 and vspct < 95 ;
run;

proc sort data = vitals out = basevs (keep = patid vsrn);
  by patid visid ;
  where vsmaxblflg&phase = 1  ;
run;

data one (keep = patid vsrn visid base_vsrn v95);
  merge vitals(in = a) basevs(rename =(vsrn = base_vsrn)) ;
  by patid;
  if a ;
  if (vsrn - base_vsrn >= 5 ) and vspct >= 95 then v95 = '1' ;
  else v95 = "0";
  format visid ;
run;
```

```
proc transpose data = one out = trans prefix = v;
  by patid;
  id visid ;
  var v95 ;
run;

data trans1 ;
  set trans ;
  cat1 = catt(of v1 - v33);
  if index(cat1,'111') > 0;
run;
```

Two PROC TRANSPOSE are written one for checking 95th percentile <95 at baseline v1 and V2 and second PROC TRANSPOSE is written to the final condition of having numeric result >= 5 and vspct >= 95 at post baseline visits.

ADVANTAGES :

1. Easy to code
2. Simple to debug
3. Lines of code can be sometimes less in comparison to other method. (depending on the required analysis).
4. The output can we manually crosschecked as the data is flat and the number of observations are few to check .

CONCLUSION

PROC TRANSPOSE is a very useful and a handy procedure that should be widely used in day-to day **SAS®** programming. It not only makes data wide to narrow or narrow to wide but also can be smartly used for other purposes taking into account the wide options the procedure provides.

REFERENCES

PROC TRANSPOSE
See the **SAS®**documentation for other procedure options.

SAS® Programming in the Pharmaceuticals industry.

ACKNOWLEDGEMENT

I would like to express my greatest gratitude to my family and colleagues for encouraging me and always supporting me.

CONTACT INFORMATION

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

Rucha Sanjay Landge
inventiv international pharma services pvt. Ltd.
Commerzone bldg. – 4 floor – 6 Yerwada
Pune - 411006
9096160321
rucha.landge@inventivhealth.com