

Flip-flop the data: To check the sustained time-point occurrences

Introduction

- Flip-flopping is the process of changing the orientation from “stacked,” “vertical,” or “tall and skinny to “flat,” “wide,” or “short and fat data.
- There are a lot of instances in clinical industry where we have to check for consecutive time points for e.g. consecutive missed visits, consecutive abnormalities, consecutive 95th percentile results etc. PROC TRANSPOSE procedure is flexible enough to generate an appropriate structure of data to identify such instances.
- Industry starters are not very familiar or may find difficulty working with RETAIN/ FIRST. / LAST. concepts. PROC TRANSPOSE on the other hand is simple procedure to code and also allows to check the data horizontally.

Data to check consecutive time-points

Create a flag variable which reflects Tally marks

Transpose the data using PROC TRANSPOSE

Collate the tally marks

Use CATT function and check for consecutive time-points

ILLUSTRATION

Let us have a look at an example where PROC TRANSPOSE will be used to obtain results than just transposing the data. Consider we have to check consecutive missed time-points (numeric lab results) in the LABORATORY ANALYSES data.



#	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

Figure- 1

- **Figure1 :**
- LAB data has scheduled visits as V1,3,4,6,10,14,19.
- The LAB_NUM_RESULT is missing at two consecutive scheduled visits V14,V19.
- We have to check and obtain patients that have such missed consecutive numeric results at 2 or more than two visits.
- To analyze data vertically not only will we require more lines of code but it will also be difficult to debug.
- Instead of checking the data vertically it is easy to analyze the data horizontally i.e. one record per subject and it will be visually easy to double check.

We will be in the same dilemma as Flintstone is. Then the very ancient method of Tally marks Clicks!!



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

Figure- 2

- The same concept of conventional Tally marks is applied in deriving the results. Vertical data will require lot of processing.
- Alternatively PROC TRANSPOSE will convert the data in to flat structure and we can obtain the desired results by checking the data at just one record.
- LAB data is transposed using PROC TRANSPOSE and the missed results are flagged as ‘1’.
- All the consecutive ‘1’ are then collated and checked

Advantages

- Easy to code
- Simple to debug
- The output can be manually crosschecked as the data is flat and the number of observations are few to check.
- Few lines of code.

Conclusion

- PROC TRANSPOSE is a very useful and 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.

Comparison between PROC TRANSPOSE vs. DATA step

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

proc sort data = vitals out = baseval (keep = patid vsrn);
  by patid vsrn;
  where vsmaxbllgphase = 1;
run;

data one (keep = patid vsrn visid baseval v95);
  merge vitals(in = a) baseval(rename = (vsrn = base_vsrn));
  by a;
  if a;
  if (vsrn - base_vsrn >= 5) and (v95 >= 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;

data anadata1;
  set anadata1;
  set anadata1;
  by patid visid;
  retain baseval;
  if vsmaxbllgphase = 1 then baseval = vsrn;
  if baseval ne . then chg = vsrn - baseval;
run;

data anadata2;
  set anadata1_1(where = (visid = int(visid)));
  by patid visid;
  prevvis = lag(visid);
  if chg > 5 and vsrpt >= 95 then tempflg = 1;
  if first.patid then prevvis = .;
  if first.subjid or visid - prevvis ne 1 then countvis = 1;
  else if visid - prevvis = 1 then countvis + 1;
  lag_countvis = lag(countvis);
  if countvis = 1 or chg <= 5 or vsrpt < 95 then finalcounter = 0;
  if chg > 5 and vsrpt >= 95 and countvis-lag_countvis = 1 then finalcounter + 1;
  if finalcounter = 3 then flg = 1;
  keep patid visid prevvis countvis baseval chg vsrpt lag_countvis finalcounter flg tempflg;
run;
```