

Transposing a Relational Database : The Q-System for Pharmaceutical Safety Summarization, Graphics and Analysis

Mary Cowmeadow, Parke-Davis Pharmaceutical Research, Division of Warner Lambert Company

Abstract

Transposing relational data to one record per subject can solve all those "many to one" problems that make analyzing and summarizing this type of data so difficult. The Q System transposes pharmaceutical safety data, i.e. adverse event (AE) data, from a relational database structure to a "data mart" type of structure with one record per patient. Since AE data are generally summarized and analyzed at the patient level, the data should be structured in this same way. Data can be entered directly into appropriate statistical or graphical procedures with meaningful results.

Transposing a relational database is a straight forward process. The major challenge is sorting, organizing, renaming and relabeling the hundreds, sometimes thousands, of AEs collected during a clinical trial. But once this was done, writing programs for standard summary and analysis is greatly simplified.

Introduction

Transposing a relational database can greatly simplify subsequent counting. For example, automotive fatal accident data may be organized at the passenger level. However counting fatalities at the passenger level would seem to make mini-vans, which carry many passengers, far more dangerous than small sport cars, which carry only a few. Instead of passenger counts, re-organizing the data to the car level, i.e., car counts, would provide a more common sense comparison.

Similarly pharmaceutical safety data is generally collected as relational data with a variable number of Adverse Event (AE) records collected per patient. Adverse Events are medical events such as rashes, headaches, strokes, etc., that are reported during a clinical trial. The purpose of collecting this data is to determine whether or not any of them are *real*

side effects of the drug. This determination can be done comparing the incidence and prevalence of various AEs between treatment groups..

Relational data are structured so that there is one record for each event, and a single patient may have many such events. However the analysis and summarization of this data is usually done the patient level rather than the AE level.

This paper describes an easy method of transposing relational data with SAS® to one record per patient datasets. In addition useful tools for re-organizing this type of data are described, including macros and the SAS *min* and *sum* functions.

Choose the worst case.

When there is more than one record of the same type, which one should be counted? For example, a single patient may have 3 records for headaches. If we are looking at AE severity, choose the most severe headache and use that for the single patient record. If we are looking across a body system such as the digestive system, choose the AE that is the worst case to represent that system.

Transposing AE data.

While transposing data is a simple idea, the simplest part of matrix algebra, the words in a book do not necessarily correspond well to prior concepts in the mind. A couple of potential semantic problems that can arise with *proc transpose* will be discussed here, along with the general difficulty of handling a very wide dataset.

Adverse Event data is abundant. One patient may have 10 AEs or more. Across an entire study, there may be hundreds, even thousands of AEs. It is coded data, with codes like "0287A". It is also labeled data, but the first 40 characters are not always unique. How do we specify a *proc transpose*?

The first question is what are we transposing. You say we are transposing AE's? Loosely speaking we are; in fact we are not.

Figure 1: An example of the original data:

The AE		
	<u>Code</u>	<u>The label</u>
Patient 1	0023m	headache
Patient 1	0008x	rash
Patient 2	0068b	diarrhea

We do not actually transpose the AE code itself, we instead transpose the "body of the table", the numeric data inside the array. The AE code itself becomes nothing more than the column header variable.

The "body of the table" is a variable that does not yet exist: it is the abstract concept that the AE is "present", because there is one record "present" in the relational database for each AE that a patient has. So we have to create a new variable and set it to "1" for "present" throughout the relational file. This new variable is what we actually transpose.

The next question is how to specify where to put these "1"s for "present". How do we specify the new column header? Is there a COLHEAD statement? No. We specify it with the ID statement. This can be confusing because many people think ID is something like patient ID. But in this context it is the new column header for the transposed data.

Figure 2: The data now ready for *proc transpose*:

The ID The data to be			
	<u>The AE</u>	<u>variable:</u>	<u>Transposed:</u>
	<u>Code</u>	<u>Label</u>	<u>AE "Present"</u>
Patient 1	0023m	headache	1
Patient 1	0008x	rash	1
Patient 2	0068b	diarrhea	1

The transpose finally looks like this:

```
proc transpose data=ae out=ae;
var present;
id ae;
idlabel aelabel;
by prot trial rxgrp ptno;
```

where "present" is coded "1", "ae" is the variable that contains the actual AE code, and "aelabel" contains the label for the AE code. (You can also just as easily transpose other AE variables such as Start Time, Relationship to Drug, etc. In these cases you do not need a new variable like "Present.")

Figure 3: The data after transpose:

	<u>_0023m</u>	<u>_0008x</u>	<u>_0068b</u>
	<u>(headache)</u>	<u>(rash)</u>	<u>(itch)</u>
Patient 1	1	1	
Patient 2			1

But this transposed data, while correct, is essentially useless, because no one would ever want to type a list of the 100s of AEs, when the list looks like the following:

_0002b _0019z _0010m _0025g

Why were the variables not given better names during the *proc transpose*? For 2 reasons: 1st, the labels were not unique in the first 40 characters, and, 2nd, the variables needed to be re-organized anyway. Nevertheless, this was a major problem to overcome if the transposed data was ever going to be useful. Fortunately the programming tools to do the job were available.

Many of our tables organized AEs under body system, and, within body system the AE are sorted by descending frequency of study drug treatment. This structure could be used for re-organization, after which it would be possible to rename them to a simple AE1-AEn and re-label them.

Useful Tools for Re-Organization:

We already had a macro in our library which, with a little modification, helped make all this reorganization possible. This macro initially put a string with the names of every variable in a dataset into a single macro variable. It used *proc contents* to do this. But for this problem a sorted *proc freq* was sometimes needed rather than a *proc contents*, hence the modification. The resulting macro looks like this:

```
%macro qgetvars(dsname, inname);
  global vars varnum;
  %let varnum=0;
  data _null_;
    set &dsname end=eof;
    count+1;
    call symput('hold' ||
      left(put(count,3)),&inname);
    if eof then call
      symput('varnum',put(count,3));
  run;
  %let vars=;
  %do i=1 %to &varnum;
    %let vars=&vars &&hold&I;
  %end;
%mend qgetvars;
```

where *dsname* is the dataset name (e.g. the output dataset from a *proc freq*) and *inname* is the variable name from the dataset which contains the list of variable names you want to get. (In *proc contents* *inname* is *_NAME_*.) The list is output to a global macro variable called *&vars*.

Once those impossible AE variable names were put into a macro variable, they could be re-organized with *%do* loops and *%scans*. An example follows which re-codes the AE variables (such as AE severity) to "1" temporarily so that frequencies can be calculated.

```
%getvars(contlist,name);
```

```
data varlist;
  set ae;
  %do i=1 %to &varnum;
    if %scan(&vars,&i)>. then
      %scan(&vars,&i)=1;
  %end;
```

Other useful tools were the SAS *min* and *sum* functions. Not everyone knows that these functions work even when some of the data is missing, minimizing the need for special coding.

The result of all of this was a decent AE transpose macro, *%qAEtrans*, the core of a good system. But much more needed to be done.

Another macro, *%qAEinit* was written as a study initialization guide so that data from a variety of protocols and from changing AE forms could also use the transpose macro. It included everything experience indicated might change with a new study, so that the user would have to make changes in only one place. *%qFormat* stored the formats for the variables used in *%qAEinit*.

The next macro was *%qSwitch*, which gives the user the option of running the data straight from the relational database or from derived datasets created from *%qAEtrans* and then stored to save execution time.

These derived datasets essentially make up a "data mart" as in "data warehouse". Much can be done with them.

Now, pharmaceutical tables have special look that can be called the "side by side" look. For example if you look at Table 2 carefully, you will see that each row of the table is really a pruned 2 by 2 table interleaved with a 2 by 1 table. The totals are in the column header. This is a typical example. The format saves a great deal of paper and comparisons across rows are easier to make. Consequently the next macro was called *%qSide*. It both summarizes the data and rearranges it 'side by side' so that it has that pharmaceutical 'look' to it.

The last step was to create a simple comma-delimited file so that SAS output could be put into Excel®.

At Parke-Davis the emphasis has always been to get electronic output into either Microsoft Word® or HTML. No-one had really looked at taking advantage of Excel's capabilities with the power of SAS behind it. Yet for this kind of data Excel had advantages that the others did not. Particularly attractive were the cell notes. These might be called electronic "sticky notes". They could go along with the final output and describe such things as how the calculations were done, the formats and data files that were used. The Excel chart wizard was also very attractive and very easy to use. The ribbon chart (Table 1) is

ideal for ordinal data such as the "mild -moderate -severe" categories of the AE severity variable.

All this made it possible to produce both attractive tables, very nice graphs, and a system with flexible input, flexible output, and standardized data in the middle.

SAS is a registered trademark of the SAS Institute, Inc.
Excel and Word are a registered trademarks of the Microsoft Corporation.

For more information, contact: Mary Cowmeadow,
e-mail: cowmeam@aa.wl.com
Parke-Davis Pharmaceutical Research
2800 Plymouth Rd.
Ann Arbor, MI 48105

Table 1. From SAS to Excel: an Adverse Event Ribbon Chart

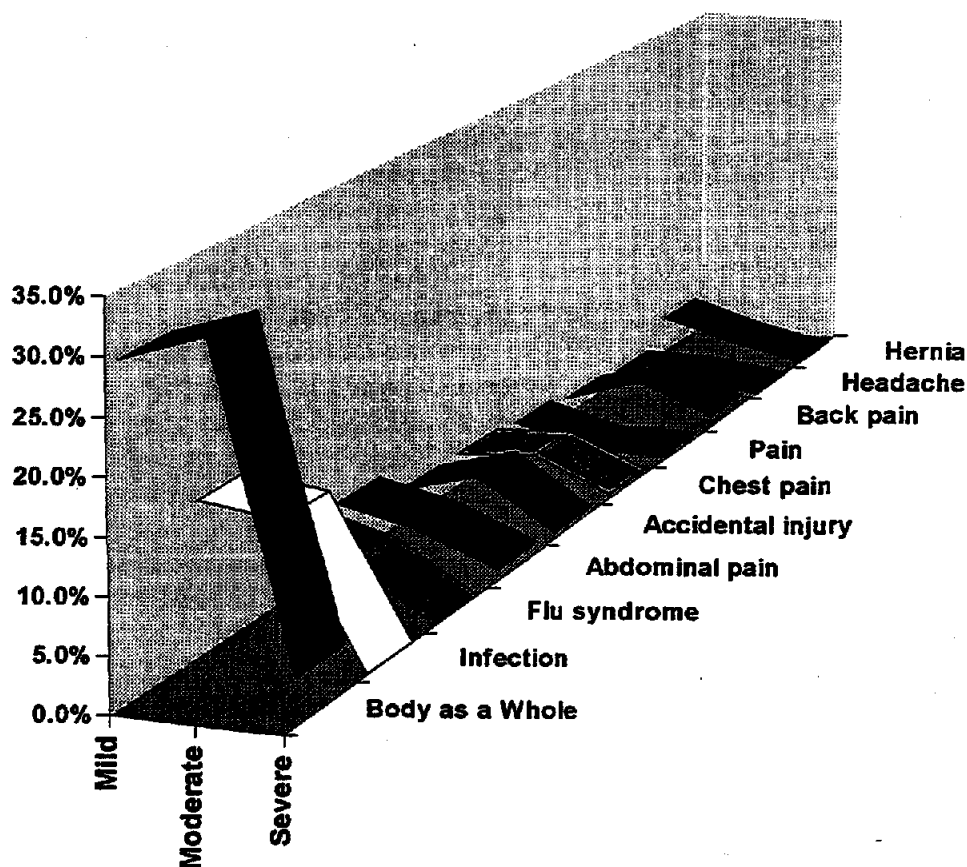


Table 2: A sample SAS output in Excel showing A data after transposing it, re-organizing it and summarizing it.

SUMMARY OF ADVERSE EVENTS

Adverse Events	Placebo			Drug A		
	Number of Patients (N=198)	Percent	Total Reports	Number of Patients (N=197)	Percent	Total Reports
Body as a Whole	129	65.20%	238	137	69.50%	278
Infection	52	26.30%	74	61	31.00%	85
Flu syndrome	22	11.10%	39	31	15.70%	51
Abdominal pain	21	10.60%	23	25	12.70%	32
Accidental injury	17	8.60%	18	21	10.70%	22
Chest pain	14	7.10%	15	17	8.60%	19
Pain	7	3.50%	9	13	6.60%	13
Back pain	13	6.60%	15	12	6.10%	12
Headache	4	2.00%	5	8	4.10%	8
Hernia	10	5.10%	12	8	4.10%	8
Abdomen enlarged	7	3.50%	8	5	2.50%	7
Cellulitis	4	2.00%	4	5	2.50%	5
Asthenia	1	0.50%	1	4	2.00%	4
Fever	3	1.50%	3	4	2.00%	4
Cyst	2	1.00%	2	2	1.00%	2
Neoplasm	0		0	2	1.00%	2
Abscess	2	1.00%	2	1	0.50%	1
Carcinoma	0		0	1	0.50%	1
Pelvic pain	0		0	1	0.50%	1
Sepsis	1	0.50%	1	1	0.50%	1
Adenoma	1	0.50%	1	0		0
Allergic reaction	1	0.50%	1	0		0
Anaphylactoid reaction	1	0.50%	1	0		0
Neck pain	2	1.00%	2	0		0
Shock	1	0.50%	2	0		0
Cardiovascular System	85	42.90%	131	72	36.50%	128
Angina pectoris	33	16.70%	40	37	18.80%	45
Atrial fibrillation	8	4.00%	14	9	4.60%	10
Cerebral ischemia	2	1.00%	2	8	4.10%	8
Syncope	7	3.50%	9	8	4.10%	10
Hypertension	6	3.00%	6	7	3.60%	7
Hemorrhage	3	1.50%	3	5	2.50%	7
Peripheral vascular disorder	6	3.00%	6	5	2.50%	5
Arterial anomaly	11	5.60%	13	4	2.00%	4
Vascular anomaly	1	0.50%	1	4	2.00%	4
Arrhythmia	1	0.50%	1	3	1.50%	3
Cardiovascular disorder	0		0	3	1.50%	3
Congestive heart failure	3	1.50%	5	3	1.50%	3
Myocardial infarct	4	2.00%	5	3	1.50%	3
Palpitation	2	1.00%	2	3	1.50%	4
Cerebral infarct	3	1.50%	4	2	1.00%	2
Ventricular tachycardia	0		0	2	1.00%	2