SAS Tutorial: Hierarchial Data

Peter L. Rikard

Introduction

Hierarchy - a body of entities arranged in a graded series. Hierarchical data appears everywhere. We are just able to ignore its problems because of relative size. A conventional hierarchical set of data is the census data. There are records at many descending levels of size of the geographical areas in the country. Each lower level belongs to a level above it. In its own, a companies weekly payroll records and its main master payroll dataset form a hierarchical database. The sum of all an individuals weekly records are related to the single record for the individual in the master database. Clinical studies are often hierarchical, with individual test reports, monthly observations and a master demographic dataset.

A raw data file of clinic data may appear as mixed types of records; the first record(s) in a group contains the demographic information for an individual. The next record(s) contains the report for a monthly observation followed by record(s) containing individual tests.

```
1 John Doe 2288 male white 34 12/10/80
2 2288 1/3/81 112345666
3 2288 1/3/81 111
3 2288 1/6/81 918
```

Or there may be separate files for each record type. In any case, a normal first organization of the data is in different SAS datasets, one for each type of record, each sorted by the ID and other variables that can be used to merge the datasets together. All the considerations below are concerned with the SAS datasets, not the original "raw" data files.

Coping with the Problems

Given either of the two collections of data above, for efficiency sake (yours and the machine), you should analyze your needs BEFORE the data is stored.

(1) What is the most frequent analysis to be performed.
(2) What other analysis will be performed.(you will miss a bunch)
(3) Will any analysis be done interactively.
(4) What turnaround will you need. (minutes, hours, days?)
(5) Will data continue to be added to the database.
(6) How much data will be collected.

There is an order to the list and to some degree the amount of data is the last consideration. It involves all of the ones before, but it is last. Both datasets above have three levels to their structure and can therefore be organized in many different ways. All three levels merged together, no permanent merging (only merge for an analysis), reorganized data merged......

If analysis is most often done with data from only one level, then leaving the data as three separate datasets is potentially an efficient method. If data is continually added to each level then separate datasets may be required.

If analysis is across levels (demographic against test data) and queries must be made interactively, with quick response, then a totally merged dataset may be required, no matter the cost.

If analysis is shared between each level and the totally merged file then maintaining all 4 datasets might be the answer.

The problem is size. If a clinic study involved 30 individuals for 12 months, with 3 tests per month then the data can be organized in many ways. If there are 30 bytes of demographic data, the monthly report has 20 bytes of data, and each test has 5 bytes of data, then each test can be merged with the appropriate month and then merged with the demographic data. There will be 1090 records of 55 bytes each, a relatively small amount of data, that can be used to easily compare test results with demographic information over time (59,950 bytes of data, about 6 3330 disk tracks).

Consider another study, not necessarily large. There were 500 individuals studied over 4 years, with an individual in the study up to 30 months, there were as many as 30 tests for one individual in a month but an average of 5 per month. For each individual there were about 40 demographic items (about 200 bytes of data), monthly reports took about 50 bytes of data, and a test took about 20. If this was combined in a similar fashion like the one above, problems occur.
There would be over 50,000 records with each record having 270 bytes of data (13,500,000 bytes of data, over 1000 tracks). The whole SAS package takes less than that.

Sheer size is not just can you get the storage for it, but also the amount of time it takes to do any analysis. If the majority of analysis is with demographic information and the monthly reports then you must throw away 75% of the records before an analysis can be done.

To the items above:

1) Analysis may force you to maintain combined datasets.

2) Multiple datasets for each kind of analysis to be performed may be justified.

3) Interactive analysis may limit the size of disk files you can access and this may mean most storage is done with unmerged files.

4) Answers may be required instantaneously (or as fast as possible), and you must maintain merged files.

5) Constantly updating data may lead you to separate files. Updating merged files can become expensive, and if you maintain several levels of merged files, you have to update them all. Keeping them as unmerged files means you only update the smallest number.

6) If the total amount of data is small (a relative term), then your organization can be totally dependent upon your other requirements. If your data is medium sized, but response is important, then organization must be considered but does not rule. If your data is huge, then your only choice may be to keep unmerged files on disk and the merged file on tape. Or just tell your systems manager to give you a whole disk pack.

The definitive statement is that there is no single method.

Now that you are discouraged that there are no answers here are some methods to cope with the problems.

**Methods**

**Reducing Size**

The smallest amount of storage is used when the least redundant data is stored.

In the examples, no data element found at another level is repeated in another level, and it is stored at the highest level possible. The only element absolutely repeated is the ID variable necessary to merge the levels together. A person's name would be in the demographic level only. Date would have to be in both the monthly and test datasets in order to merge the files but if the person giving a test is the same person making the monthly report then their ID is contained at the higher (monthly) level only. Non-redundancy aids the process of fixing bad data, but redundancy can provide for internal checks.

It may be possible to reduce the number of observations or even the levels of data. In the first dataset, the test records had ID, date, test type and a result. There were never more than three tests per month, so test records could be merged into a single record per month, with ID, month, TYPE1-TYPE3, RESULT1-RESULT3, and this record merged with the monthly record.

**DATA FEW:**

```sas
DATA FEW;
SET BIG.TEST;
BY ID DATE;
ARRAY T(J) TEST1-TEST3;
ARRAY R(J) RESULT1-RESULT3;
IF FIRST.DATE THEN DO;
   DO OVER T;
      T=0; R=0; END;
   J=0;
   END;
J=J+1;
T=TEST;
R=RESULT;
RETAIN TEST1--RESULT3,
IF LAST.DATE;
```

This reduces the number of levels, the redundancy of date and ID, reduces total storage in the example by 15%. The second example could not be handled this way. There would have to be 30 test variables and 30 result variables and in most cases they would be missing. As SAS data records are NOT variable within a dataset, the amount of storage would increase rather than decrease. This method may also be precluded by analysis to be performed. If comparisons are to be made across tests then the data may have to be uncompacted every time you wish to use it.

Virtually all categorical variables should be stored as characters rather than as numeric variables. SEX 1=male 2=female, takes a minimum of 2 bytes to store as a numeric (8 bytes if you do not use length statements), but only 1 byte to store as character. Responses to questions can often be stored as categorical (character) variables since they are rarely used in computations, but can still be used in analysis.
Storing long character data may be unnecessary. If there are 10 experimenters in the study, storing the experimenters name Dr. Alfred T. Bonkwithers, on the monthly record consumes space, storing an ID ("ATB") instead serves the purpose and can be expanded using PROC FORMAT for reporting. This may lead to problems in coding the data, but may reduce problems from data entry (Dr. Alfred Q. Bonkwithers would be considered a distinct person by SAS, so would Dr. A.T. Bonkwithers...).

Selecting observations

In analysis, it is often necessary to know some summary conditions BEFORE selecting an observation for the analysis. Is there enough data for this person to be included, is the final state of the individual acceptable for this analysis (not dead, cured...), as well as the "correct" demographic group (male, black, over 35...). Even if all levels are merged together, this may take several passes through the data, and several sorts.

DATA SELECT;
   SET COMBINED;
   BY ID DATE;
   IF SEX='M';
   IF FIRST.ID THEN N=0;
   N+1;
   PROC SORT; BY ID DESCENDING DATE;
DATA FINAL;
   SET SELECT;
   BY ID DESCENDING DATE;
   IF FIRST.ID THEN DO;
   IF N<15 OR COND='DEAD'
      THEN DELETE='YES';
   RETAIN DELETE;
   END;
   IF DELETE='YES' THEN DELETE;
   PROC SORT.......;

If this has to be repeated for other types of analysis and the data is STATIC, then performing counts of this type and getting final conditions may be done once and stored at the highest level. Observations may now be selected from the information available in the demographic file, without sorting, and observations selected from lower files with matching ID.

DATA FIRST;
   SET DEMOGRAP;
   IF SEX='M' AND MONTHS>10 AND
      FIN_COND='DEAD';
DATA FINAL;
   MERGE FIRST(IN=IN1) TESTS;
   BY ID;
   IF IN1;

Note that this is generally efficient only with STATIC files. If data is continually added then then summary variables in the highest level must be continually updated. This adds some data redundancy, but may be more efficient than a long selection process.

If the data is static then selection may be done from SAS datasets using the POINTER= option and collecting the beginning and ending observation numbers into the demographic dataset (see unconventional merges in the Tutorials).

DROP and KEEP

A technique that could greatly speed up most analysis and the one most often ignored is the judicious selection of the variables needed for the current analysis. By KEEPing only variables needed (or DROPping extraneous ones), the amount of data that is read is limited. The less data read, the less time wasted in any procedure. There are two methods available. KEEP or DROP statements may be included in a DATA step when selecting observations, use the one that is easiest for you to put in, the action is the same. The alternative, often ignored, is the use of the KEEP=/DROP= parameter available when ever you reference a SAS dataset.

PROC SORT DATA=BIG(KEEP=(...));

There are other techniques that apply to any use of procedures in SAS. Use the least costly procedure that produces the output you need (use SYSREG instead of GLM). Limit procedures to producing only the statistics you need, rather than ALL of the default ones. (see SAS Applications Guide)

Conclusion

The general idea is planning and the use of efficient techniques. The larger the file the more planning will pay off, but thinking through the process, is required no matter the size of the data.