Human errors in machine readable data are rarely random errors; usually, they are very systematic. Because they are also not independent nor identically distributed from some notion, there is rarely easy to cancel out. Errors in data tend to follow distinctive patterns, but finding these patterns is rarely easy.

SAS software provides many good tools to maintain quality control over input data in information processing. Although used more often for error detection than for error avoidance or correction, SAS is useful for all three purposes. At the same time, many features of the SAS system present easy opportunities to introduce error into a database. All SAS programmers should be aware of these pitfalls.

Error Avoidance

Almost all data processing projects still require, at some stage, the transfer of data from a form that can be read by people to a machine readable form. Inevitably, this causes transcription errors, three-fourths of which are substitution errors of one character for another (1). It appears that these errors do not insidiously introduce error into a dataset without the use of the "BY keyfield" statement where one is intended. This makes it almost insidiously easy to introduce keyfield errors into a dataset. When the BY statement is inadvertently forgotten, as it often is, Sally gets Joe's data (or his paycheck or his criminal record) and no one is the wiser.

Error Detection

It's easy for programmers to focus their attention on using SAS proc's to find errors in a dataset, and several proc's are useful for error detection. But such an orientation can divert attention from obvious clues that are sometimes present in the SASLOG. Few programmers ignore highlighted "ERROR"
messages; this is not so true of the "NOTES." For example, subsetting IF's frequently surprise by producing an unexpected number of observations (zero !). A MERGE will result in fewer than the expected number of variables in the output dataset if the programmer has forgotten that the input datasets have more than just the BY-group variable(s) in common. Numeric-to-character conversions, "missing" value results from arithmetic operations, and "more than one input dataset has multiple BY-groups" in a merge are often overlooked indications in the SASLOG that something, somewhere has already gone drastically wrong.

Looking for errors in the data should be the first step in all exploratory data analyses. Human eye examination of long lists, for example, the output of PROC PRINT) should not be solely relied upon to do this. PROC UNIVARIATE, PROC CHART, or PROC DATASHOWS are much better for finding "wild" codes. If there are more than one hundred or so different analyses. Human eye examination of long fewer than the expected number of variables in observations (zero !). A MERGE will result in generate reams of output unless carefully For example, subsetting IF's frequently with horizontal bars is a good first step even numeric values for a variable, using PROC CHART "missing" and "discrete" options. This will generate reams of output unless carefully applied, but it shows exactly what's in the dataset in a much more visually interesting way than PROC PRINT. Unfortunately, it does not tell where to find the observations that seem wrong. Many programmers use PROC FORMAT with a "FORMAT statement in the datastep to validate data through a table lookup. This is of the form, "1 = male, 2 = female, 9 = missing, other = miscoded." Again, this method doesn't tell where the errors are. When using PROC UNIVARIATE for numeric variables, be sure to add the "ID keyfield" statement to show not only what the outliers are but also where to find them. Do not neglect to check the keyfield itself. In particular, if the study design calls for a unique keyfield in each observation, always check for duplicates.

The tails of distributions are always a good place to focus suspicion. This is true not only because the tails are the most likely places to harbor errors but also because the tails will often determine the results of an analysis, and it is especially true if the statistics computed depend on sums of squared deviations from a mean. More robust measures of central tendency exist than those that depend on the variance of a distribution, the interquartile range in particular (5). The main ingredient of such a statistic, QRANGE, is readily supplied by PROC UNIVARIATE. When using the REG procedure, the INFLUENCE option is very useful in finding the data points that have the most consequence for the results. These observations should be carefully examined.

Quality control means more than just checking variables one-by-one in a file. Making sure that they are logically consistent is just as important but not as easy in SAS. The most exacting method is to use a long series of "IF" statements in a datastep. To borrow a famous error in the 1970 Census special subject tabulations on transportation.

"If residence equals California and workplace equals Hawaii and trip-to-the-tally bus than output the case ID and check this out." This is also a very tedious method. An easier technique in some situations is to use a SORT followed by a PROC PRINT. For example, a sort of height by age and sex followed by a PROC PRINT of the first and last 2 percent of all cases in each combination of age and sex might turn up good candidates for close examination. The chief virtue of such a system is that it identifies exactly what the outlier is in the tails as well as the values of every other variable in each observation listed. Knowing the values of these other variables will help in deciding the validity of observations in the tails. It also aids in finding out what might have gone wrong if an observations is in error.

The following is an easy macro that can produce a series of single variable sorts and PROC PRINTS for use in examining distributional tails. The programmer only needs to pass the macro parameters for the number of all observations in the dataset (ALLOBS), the number of observations to print out (PRINTOBS), the dataset name (SASDSN), and the list of numeric variables to check (of the form, VARLIST = WEIGHT):
records through data entry, error detection, and updating. SAS programmers have to improve audit trails on their own (6). Because errors can be discovered at any stage (especially after data analysis is well underway), "checkpoint and restart" oriented record keeping is essential.

Whenever an error is discovered and corrected, do not neglect the possibility that a new error will be created by the correction transaction itself. This is not so much because the corrected variable could be invalid as because the new value might be in logical conflict with the value of another existing variable. At a minimum, this makes it necessary to keep a record of changes by keyfield and item changed, the time and date of the change, old and new values, or a notation of addition/deletion for an entire case. These requirements lead to the use of transaction files.

Many SAS programmers do simple updates in the data step of the form: "If ID equals ABC then VARIABLE equals newvalue." A less cumbersome approach is to create a transaction file for use with the UPDATE statement. Using SET with the POINT option is even more direct and updating. SAS programmers

FSEDIT is entirely suitable for error correction or data modification of any kind because the corrected variable could be invalid when used to update files in a manner that leaves no permanent record of the changes made. If the updates are in sorted order, even the old fashioned PROC EDITOR is more useful if only because a SASLOG is behind when done in batch. Of course, PROC FSEDIT presents a sensible means to create a transaction file of changes as long as these changes are permanently recorded and the old dataset as well as the transaction are archived (8). Never throw these away: today's updates frequently require tomorrow's corrections.

Conclusion

Successful quality assurance depends as much on the position and personality of the SAS programmer as it does on programming skills. Quality control in a data processing department is, first and foremost, a responsibility of management: The SAS programmer responsible for quality control must function as a manager in a very real sense. A manager is demanded in the quality assurance role by the "n plus and minus one" rule (9). This rule says that the review of work done in phase n of a project must be done by those directly responsible for the work in phase n minus one as well as those responsible for the next phase, n plus one. The n minus one reviewers are best suited to catch errors in transcription, or transmission. These errors cause the most intractable of all quality control problems and they occur whenever data are transferred. The n plus one reviewers should also have a turn. These reviewers have the best motivation to perform quality assurance tasks because they have the most to lose if phase n isn't done correctly. Although keypunchers, coders, editors, and programmers all have important roles in maintaining quality control, the ultimate responsibility must never be left to them. Management must assume the responsibility for quality assurance because it is management who designates the goals a data processing department should achieve, and because the information generated should ultimately be useful to management - no matter who else is involved in the intervening stages.

For managers in phases n plus and minus one, SAS is a fine tool in providing quality assurance for any dataset. In phase n, or in other hands, it will be much less helpful.

References


433