

Using SAS to Process TIGER Line Files

John Blodgett, Urban Information Center
University of Missouri - St. Louis

Introduction

In our paper presented at SUGI 15 in Nashville we described software developed by the Urban Information Center related to using SAS to access the Census Bureau's TIGER line files. This software converted the series of TIGER files as distributed by the Bureau and converted them into a SAS data library (SDL) with an enhanced structure to allow better linkage between the various TIGER subfiles (data sets.) We then described software that accessed this TIGER SDL in order to generate polygon boundary data sets suitable for use with the GMAP procedure. In this paper we'll look at how the Urban Information Center has utilized this software in recent months as we attempt to use TIGER as a tool for helping us to understand and analyze the new 1990 census data products. We'll look at three basic kinds of applications: mapping, 1980/1990 geographic correlation, and linking local address files to census geography (geocoding.)

Code References

Much of the code described in this paper, including SAS macros and job streams (most with MVS JCL, a few suitable for PC-DOS) is available on diskette from the author, or via the SASPAC-L file server (BITNET) maintained by the author. Rather than attempt to reproduce that code in this paper we are providing references to the names of the distributed modules. For example, a reference to the source code for %GENLABS macro will be "see GENLABS.SAS". Hard copy packets of the modules referenced in this paper will be available at the MWSUG conference and can be requested from the author.

The Conversion-to-SAS Step

The Census Bureau has added some additional files to the standard collection of 6 which our convert-to-SAS software deals with. None of these new files is essential to the kinds of applications we were interested in and so we have not modified our software to deal with them. In fact, the only significant change to our conversion software has been the addition of an option to process multi-county input files. The Bureau distributes TIGER line files only for counties but they can, of course, be concatenated for processing. As it turns out, they have to be both concatenated and then re-sorted in some instances as we found out in the St. Louis metro area. This glitch caused us to change our mind about creating a single TIGER SDL for the metro area and instead to go with separate SDL's for each county. To date, we have limited our TIGER processing to the 3 largest counties in the metro area: the city of St. Louis (an independent city or county equivalent), St. Louis county and St. Charles county. While we do our TIGER line file processing for the 3 counties separately, we do much of our mapping and analysis for the tri-county area as a whole. We run this software using SAS6.06 in an MVS/ESA environment. We store the TIGER SDL's as V5-format SDL's to save space. The conversion macros used to convert the TIGER files are TIGERLN.SAS and TIGRLNK.SAS. A sample run to invoke them is TIGER183.MVS.

Mapping Applications: Chaining

The TIGRGPOL application described in our SUGI 15 paper accessed a TIGER SDL and generated GMAP mapping data sets. This application was developed and run in a 6.03 environment on a PC and came with a SAS/AF front-end to assist the user in specifying the parameters and choosing the steps they wanted to run. We did this (back in 1989-90) because we wanted to explore the capabilities of the PC-SAS platform and because we wanted to take advantage of some of the features of Version 6 which were not then available in MVS. Neither of these considerations is relevant today, however, and it is much more convenient to do all our work in MVS. This eliminates the time and disk-consuming download-to-PC step and makes the chaining execute in a fraction of the time required on a PC.

We have actually only had to run the chaining application a handful of times. We ran it once for each of our 3 counties to chain at the census block level, and once for each county to chain the 1980 census tracts. We were surprised and pleased to discover that these codes from the auxiliary geocodes file (TIGER3) were filled in for every observation (in the St. Louis area files at least) and that we were able to chain around them without a break. There were also no unchained census blocks. We are not using an AF front-end for our chaining since the setup to handle it is straightforward.

A sample setup to chain at the census block level is shown in CHAIN183.MVS, and a sample run to chain 1980 tracts is in CHAINTR8.MVS. Each setup references the SAS modules PARMEDIT, GENLINKS, CHNLINKS, CHAINRPT and GENGMAP.

Using PROC GREMOVE

Although our chaining software is capable of creating polygons at specified levels such as census tract, county subdivision, block group, etc. in practice we have found it necessary only to chain at the 1990 block and 1980 tract levels. This is due to the fact that the 1990 blocks serve as "atomic units" for all other 1990 geographies. Thus, we can use PROC GREMOVE (in conjunction with some sort and match-merge steps - see GREMOVES.MVS) to generate virtually any of the polygon types we want. In addition, it is a simple matter to define new geographies (polygon types) in terms of unions of census defined geographies. In our case, we defined a unit called "tract group" consisting of unions of contiguous census tracts. We use a "format table" (PROC FORMAT-generated format code that relates each census tract to its corresponding tract group code and is invoked using a PUT function in a DATA step) to easily define these new geographic summary units. Such custom units are useful not only for aggregating and re-summarizing the census data, but also for assigning useful locally-recognized area names which help users relate to the data.

We used the Public Law 94-171 census data file ("PL94") to get a SAS data set defining the correlation between the 1990 census

blocks and all the other geocodes that are unique to each block. We could also have built this correspondence by examining the correspondences on our TIGER1 data set, but it was somewhat more convenient and efficient to use the PL94 data set.

Annotating Census Maps

Generating maps depicting census data for units such as census tracts, block groups or blocks presents some special problems that you don't have when using geographies like states and counties, where the user of the map has some knowledge of what they are looking at (they can probably recognize Iowa without a label.) There are very few users who are going to be able to look at a census tract map for a county and recognize the tracts. It becomes very important, when mapping data at such levels, to help the map user relate to the geographic units of the map through the use of carefully chosen polygon labels or other annotation. In many cases (such as a census tract map for St. Louis) the sheer number of units (over 100 is a typical tract count for a metropolitan county) presents a problem. Not only is there considerable work involved in generating the labels, but there is the problem of how to make them readable on the map and how to prevent them from "running over" each other. There are no easy solutions to these problems, but we have developed some tools that can help provide some partial solutions.

To handle the problem of generating hundreds of polygon labels we have developed a SAS macro, GENLABS (see GENLABS.SAS). This macro reads a SAS GMAP data set (and, optionally, a "response data set" with values to be mapped for the polygons) and generates a data set suitable for use with the SAS/GRAPH annotate feature. Each observation on the output data set generates a label for a GMAP polygon. The default label is just the ID variable(s) for the polygons, but other options include names assigned using format codes and/or the value of the response variable associated with the polygon. The routine calculates the centroid of the polygon to use as the x, y coordinate values.

Because census geography is a complex hierarchy (counties are divided into tracts which are divided into block groups which are divided into blocks) it is often necessary or desirable to show more than one level of geography on a census data map. For example, a map displaying data for block groups will be much clearer if the outlines and the labels of the census tracts are shown as well as those of the block groups. The GMAP procedure will draw the BG outlines, but not the tract boundaries. The best way to display these "higher level" geographic boundaries is to use the annotate feature to generate a series of MOVE, DRAW's to generate the additional outlines. A second macro, GOUTLINE, has been developed (see GOUTLINE.SAS). You invoke the macro, pointing to a polygons data set and specifying the parameters to control the line type, width, color, etc. that can be used to draw the outlines.

TRMAP1.MVS is a complete job stream that employs the GENLABS/GOUTLINE tools to create the sample map shown in Fig. 1. Actually, this job created a graphics stream file which was downloaded to a PC and the map was then plotted on an HP Laserjet II for the sake of creating the figure, but the setup was originally designed for a CALCOMP 1051 4-pen platter and utilizes color.

This is a data map displaying 113 polygons (census tracts) with automatically generated labels. In addition to the tract labels

(which are borderline-readable at this scale) the map contains bold outlines and labels for the tract groups in the city. These commonly used neighborhood name labels significantly enhance the ability of the local user to understand where they are on the map.

This map pushes the limits of annotation. Another approach would be to plot a "base map" of this area at a somewhat larger scale with all the annotation shown plus some key streets. Then when plotting actual data maps, you could omit all annotation except for the tract group outlines and labels which could serve to refer the user to the base map for details about specific tracts. Another option that could then be used would be to have GENLABS plot actual response values as labels within the tracts.

Relating Different Geographies

One of the most troublesome problems facing users of geographic summary data is how to deal with data summarized at different geographic units. You want to do a demographic analysis of the population in St. Charles county, studying trends in key variables between 1980 and 1990. If all you want to do is look at county level data you have no problem, but any smaller units such as census tracts or below present problems because the units are not consistent from decade to decade. In a fast-growing county like St. Charles over half the census tracts used in 1980 were not used in 1990; instead most were split into several new tracts for 1990. At the block group (BG) level it is even more perplexing, since in some of the rural portions of the county there were no block groups defined in 1980 (they used areas called enumeration districts, or ED's, which went away with the '90 census.) At the UIC we have a standard data product called CAPS (Circular Area Profiling System) which uses data at the BG/ED level (the smallest geographic units for which the Bureau reports data such as income, education and occupation). The system uses the geographic centroid of each BG/ED to determine if it within n miles of a site, then aggregates a standard demographic summary observation for each unit found to be in the circular area. This worked fine for 1980, with the help of a Census file called MARF2 which contained latitude-longitude coordinates of the "population centroids" of each BG/ED. With the 1990 data being released we were very interested in being able to take all this 1980 data at the BG/ED level and convert it to the new BG units being used for 1990. When/if we can do this we can alter our CAPS system to provide both 1980 and 1990 data for the same circular areas. The problem then becomes one of assigning each 1980 geocode (BG or ED) to its 1990 equivalent. In some cases a 1980 ED area which was mostly farms in 1980 is now mostly subdivisions and hence consists of a dozen BG's, so assigning it to a single 1990 geocode may not be totally accurate. A scheme which could allocate the ED to a series of 1990 BG's would be better in such a case.

The TIGER files have figured in two different approaches to this problem. The first involves relating the 1980 geocodes in the TIGER3 data set with the 1990 codes found mostly in TIGER1. By simply counting TIGER line segments where the codes correspond you can get a good measure of how a 1980 code relates to 1990 geography. For example, in the city of St. Louis, block groups were left virtually unchanged over the decade, except that some were simply renamed. Because of this, and because the 1980 block group information is almost always present in the city portion of the TIGER files (because most of the features come from the 1980 DIME files) it is a relatively simple matter of running a MERGE step and a FREQ to see the 1980/1990 BG

[illegible]

 +3 TO +10

Census Data and TIGER Files 167

correspondences in the city. In the more rural St. Charles county file, however, we were up against a different situation. Only a small fraction (less than 25%) of the TIGER3 records for St. Charles came from the DIME files and thus had block (group) codes for 1980 filled in. So this easy method did not work. We tried a different approach, which as this paper is being written has still not been judged as a success or failure.

Point-in-polygon Encoding

A fundamental feature of GIS software packages is the ability to relate a point to a closed polygon using a "point-in-polygon" routine. Such a routine requires as input the coordinates of the points on the boundary of a polygon and the coordinates of the point to be tested; the routine typically returns a 3-way flag saying whether the point is inside, outside or on the perimeter of the polygon. Such routines are often hidden from direct use but are widely used in other routines, such as those that take a point and a "coverage" (a set of non-overlapping polygons that cover the geographic area), and return the id of the polygon(s) within the coverage where the point is located. This is the kind of capability we were interested in for trying to solve our geographic correspondence problem. So we decided to see if we could do it in SAS.

We started with a clumsy routine (macro) called PTINPOL (it turned out that we ended with a clumsy routine as well; this sort of thing is really not a SAS strong point due to the lack of a true subroutine capability - it was much easier to implement in FORTRAN, the language in which the algorithm we borrowed was written.) Save for possible problems dealing with multi-segment polygons and polygons with holes in them, we got our PTINPOL routine working on trivial test cases in a few days. But when we went to write the more useful routine, the "driver" that would apply this macro to a whole set of polygons and a given point to see which, if any, of the set of polygons contained the point, we found it to be a very tedious proposition. We got sidetracked and several months passed between writing PTINPOL and actually trying to use it to solve our problem. (In the meantime we kept hoping that our real GIS would arrive, and then the hardware needed to run our real GIS would arrive and then we wouldn't have to do this in SAS.)

Our conversion strategy called for going to the 1980 geography file, MARF2, with geographic codes and lat-long coordinates for all the areas we wanted to relate to 1990 geography. We used BG/ED areas, split by townships and places as our smallest 1980 unit and attempted to assign each to a 1990 BG using our PTINPOL-invoking program. The TIGER connection here is that the BG polygon files used by this program were generated by our software from TIGER files. A critical assumption here is that the latitude-longitude coordinates used on the 1980 MARF2 file were accurate and in the same "base" as used on the 1990 TIGER files. This would only be a valid assumption if the TIGER file from which our polygons had been generated had not been newly redigitized for 1990, but instead was based on the (relatively inaccurate) 1980 DIME coordinate system used in most metropolitan areas. We also had to assume that the MARF2 coordinates were "accurate" relative to the DIME coordinate system. Our experience to date indicates that the first assumption proved to be pretty good - the two coordinate systems seemed to be a pretty good fit when we plotted them over each other. However, we did see some rather glaring examples of where our 2nd assumption did not seem to hold true, i.e. where the MARF2 coordinates were apparently just not accurately assigned.

On a first-shot test (using only St. Charles county) we had 271 test cases and were able to assign all but of these to a 1990 BG. Because the first 4 digits of the census tract almost never changed between '80 and '90 - tract 3109.00 was subdivided into 3109.01, etc - we were also able to run a basic validity test of the algorithm to see if it put any areas with the wrong tract. We found a dozen cases where it did. Then we drew a picture of what we had and looked at the points to see how well we did. There were some borderline cases where the algorithm failed, probably due to lack of precision in the coordinate assignments on MARF2 and/or DIME/TIGER, and a few examples of inaccurate coordinates on MARF2. We also had an algorithm problem related to PTINPOL not working for polygons with "holes" in them, of which we had quite a few. As this is being written the routine is being studied for how to fix this apparent bug. And we are still evaluating whether or not this method of automating the geographic correspondence problem is sufficiently reliable and convenient for us to go ahead and use it in other metro portions of the state and on the Illinois side of the St. Louis area.

Using TIGER to Link Address Files

One of the most important applications of TIGER line files, at least in urban areas with well defined street addressing schemes, is its use as a tool for relating street addresses to census geography. The observations in the TIGER1 data set which correspond to city streets have variables containing the name of the street feature and the numeric address ranges on either side of the street. Since the record also has the census tract, census block, place (city), and county subdivision codes for each side of the feature it serves as a street directory file. Say, for example, you had a list of 200 customers with their home addresses and you wanted to place each of them into a census tract. You might want to do this just so you could get some idea of where your customers are coming from, or you might not care about the geographic location, but you might want to look up the median housing value on the 1990 STF1 census file for each of the tracts and use this to get some idea of what kind of neighborhoods your customers come from. Using nothing but your TIGER SDL you could (assuming it were on disk) use PROC FSBROWSE to search the TIGER1 data set for a match on each street name. Lets say your first address was 123 Maple Ave. You could enter a F(ind) command to bring up the first observation (if any) with "Maple" as the value of the NAME variable. You would then look at the address range variables LEFTLOW, LEFTHIGH, RGHTLOW and RGHTHIGH to see if these numeric ranges contained the value 123. Suppose the first observation you find has LEFTLOW=1000 LEFTHIGH=1098 RGHTLOW=1001 and RGHTHIGH=1099. Since these ranges do not contain 123 you would hit the "RFIND" key to bring up the next observation for Maple. Suppose this one had LEFTLOW=101 LEFTHIGH=199 RGHTLOW=102 and RGHTHIGH=198. This would be a match on both the street name and the house number/range. So what is the census tract? Suppose you see that TRACTL=1001.00 and TRACTR=1015.00; which one goes with house number 123? Since the address ranges on the left side are odd and the address you are searching for is odd the match is on the odd (left) "street face", so the correct geocode is the left-side tract, 1001.00. Or is it?

Just because you have found an address range that matches your address, how do you know its the *only* such range. Maple is a pretty common street name. If you are working with a large TIGER file there's a good chance there will be more than one Maple street and a fair chance there will be two "100 blocks" of Maple. Of course, we haven't even considered in our example

whether the feature TYPE variable in our "matching" TIGER observation had a value of AVE or ST or CT. We also did not check to see if there was a qualifying value for DIR, the street direction. Many streets in many cities have repeated address ranges for West and East or North and South sections of streets; what if the reference file had a 100 block of W Maple and a 100 block of E Maple? As you can see, address linking can get complicated. Doing it manually with a browsing procedure would allow you to make some common sense judgments about whether the match "looks good" but a program that attempted to do the searching and "best matching" would have to be pretty sophisticated.

One of the things that most geocoding programs do is use some kind of a geographic qualifier to limit the search for a given address. You might limit the search to a county, a city or a ZIP code. Of these, the most reliable and typically the most readily available and accurate is the 5-digit ZIP postal code. The chances are you have a ZIP code to go with your customer's address at 123 Maple Ave. What's more, you probably have this code sitting in a separate field in your accounts receivable or master customer file. The city and county may be "obvious" to anyone familiar with the area, but it is less likely that there are codes representing these geographies on your customer file. So ZIP codes can be important for the address linkage process, especially for large file automated linkage. Fortunately, the TIGER files have variables ZIPL and ZIPR to provide the ZIP codes along the sides of the street faces. The Census Bureau classifies ZIPs in the same category as the address range items, i.e. as part of a record's address information - not in the same category as census tracts or places. This turns out to be an important distinction since it stems from the fact that in the TIGER data base as built by the Bureau, ZIPs are not a "coverage" the way tracts or blocks are, but rather just extra address information that has to be entered by a clerical encoding process and *not* by digitizing of the ZIP boundaries. As far as the Census Bureau is concerned - and they are technically correct - ZIP codes are sets of address ranges, not polygons, and cannot be digitized.

Address Information in TIGER Files

There are essentially no missing census tract or block entries on a TIGER line file as delivered by the Census Bureau. Given the way the files are built, they cannot be missing since this would create a topological hole that the software would not permit. But address range data (including ZIP codes) are not like this and are very often missing on the files. The degree to which these fields are reported vary widely from area to area. They are most likely to be present in areas where a 1980 DIME file was available (this limits it to the larger metro areas) and where the street segment existed in 1980. You will not find range/ZIP data for rural counties, small non-suburb cities (under 50,000), or most new-growth urban areas. The Census Bureau has indicated an interest in trying to go back and fill in these missing data but it is not clear whether or when this will be done. How important this limitation is to someone wanting to use the TIGER files for geocoding varies with the kind of application. Someone wanting to geocode a file of city voters in St. Louis or Detroit would not have much of a problem since these are old established street networks and the address information would be more complete. If your interest was in a smaller city like Cape Girardeau, Mo. you would have to do a major update to add the address range data to the TIGER file before you could begin. If you need to do coding for a complete metro area like St. Louis using address files from all over the area, then you will have a problem when dealing with addresses outside

the old established 1980 street pattern. If the purpose of your geocoding is to turn large address files into statistical counts by census geography (for example, a file with total live births by census tract by year based on geocoding a vital statistics address level file) then there will be serious problems with your results - you will have to augment your automated (TIGER-based) coding with expensive "hand-coding" to avoid under-counts which are biased towards under-reporting activity in high-growth (new street) areas.

Geocoding with SAS/TIGER

While it is possible to do geocoding "manually" with a TIGER file, either by using the kind of tedious FSBROWSE approach described above, or by generating an address range report from the file and "looking up" addresses in the alphabetized street directory, these are not satisfactory solutions for large file applications. When thousands of addresses need to be linked to the TIGER file it becomes critical to automate the process using sophisticated address matching software. We have looked at two of the newer desktop GIS systems which have geocoding capabilities and have been surprised at their lack of sophistication. They seem more designed for interactive 1-at-a-time address coding, the purpose of which is to place the address at a point on a CRT map for you - the "pizza delivery" application - rather than the large file-oriented applications. We found it difficult to get the programs to locate addresses even when we knew where they were. Neither used ZIP codes to help distinguish between duplicate address ranges.

The Census Bureau really invented geocoding in the late 60's when they created the first readily-available address reference files (DIME) and the first sophisticated software package for making use of those files in a geocoding system. That software package was called ADMATCH. It was written in IBM Assembler, ran only in OS and DOS (IBM mainframe operating systems in those days), and was a sterling example of a very clever program with an incredibly awkward and error-prone user interface. Everything was driven by control cards in fixed formats and if you typed any of the dozens of program specs in the wrong column you could get some very strange results. The ADMATCH system had two very large, sophisticated modules called PREP and MTCH. The PREP preprocessor did the work of parsing address fields and standardizing them by creating a uniform 75-character "match key" which it pre-appended to each record it processed. You used PREP to standardize both your reference file (DIME-based) and the data file you wanted to match to the reference file. The MTCH program did the work of actually linking the sorted preprocessed reference/data files under the direction of control cards which would tell it what fields you wanted to match on, what kind of reliability you wanted, what fields you wanted to transfer to the data file when a match was found, etc. Both of these programs were very sophisticated for their day, were extremely fast and efficient (they had to be because the Census Bureau ran them against very large files on very slow IBM 360 computers), and proved to be very reliable. At least they were reliable in the sense that they were consistent and predictable. The reliability of the results of using them was another matter, since it was very easy to make a mistake in coding the control cards which might or might not be detected.

SAS/ADMATCH

The Urban Information Center got in on the ground floor of the geocoding business in the early 70's. The ADMATCH system was

used extensively to do projects like geocoding files of mental patients which were then aggregated and matched to the 1970 census files, allowing us to generate choropleth maps with titles like "Schizophrenics Per Capita" by census tract. This software was tedious to use and took lots of time to set up, but programmers were still relatively cheap back in the early 70's. But by the late 70's things had changed. We discovered SAS and started using it seriously in 1976. Where ADMATCH jobs required lots of job control language (it took a dozen lines of error-prone JCL just to sort a file and ADMATCH requires *lots* of sorts), and a number of fixed-field control cards, we saw that SAS let you sort files with just a simple line or two of code and used mnemonic easy to read and code parameters and options. And SAS data sets were so easy to create and manage as compared to OS system files that had to be allocated and managed.

We tried various approaches involving special preprocessors and JCL generators for the ADMATCH system before going to a SAS-based system around 1980. (We even tried writing our own versions of PREP and MTCH in PL1 but they weren't as good and when the programmer who wrote them left, we abandoned them and went back to the classics.) The basic idea for SAS/ADMATCH is that we would continue to use the extremely reliable and efficient PREP and MTCH programs, but we would build a user-friendly interface to them so that no programmer would ever again have to code one of those error-prone control cards. The whole thing would be coded as a SAS application, taking advantage of SAS's ability to invoke non-SAS modules. Our system would build the control cards and the input files for the preprocessor and then simply use PROC PREP;.

It is beyond the scope of this paper to go into the details of the SAS/ADMATCH system. What started out as a relatively simple front end got more complicated over the years as we tried to implement "pre-checking" of addresses (testing for probable matches against an extract file before matching against the full reference file) and a fairly complicated sub-system for coding and applying updates to unmatched address records. But the PREP and MTCH modules still remain (for now) at the heart of the system. This severely limits the portability of the system, since these assembler/load modules will not cross platform boundaries. We have considered once more rewriting the modules in SAS, but have been deterred from doing so both by the programming effort that would be required and the fact that it would be a much slower program to run if it required compiling these very large SAS DATA step programs each time. With Version 6's ability to create stored program modules, one of these barriers has been removed. Another feature which could be used to greatly simplify the way the system works is the ability to use indexes and to do keyed direct access reads. We already have the indexing capability in 6.06 and the keyed reads (already available in SCL) will soon be available for the DATA step.

The TIGER interface

While TIGER is a vital part of our geocoding system, the actual programming that has to be done related directly to the TIGER SDL is rather trivial. We had a macro called PREPDIME which we used to access DIME files and create the preprocessed reference file (created by PREP and accessed by MTCH.) It took less than a day to write PREPTIG, a variation on PREPDIME that dealt with the new TIGER features. The transformation was fairly trivial since the two files are so structurally similar. The only tricky part was writing it so that we could create a single geocoding reference file based on multiple (one per county) TIGER SDL's. For details see

PREPTIG.SAS and PREPTIG2.MVS.

Conclusions

TIGER files are big and complex. Many of the kinds of applications for which they are most appropriate involve sophisticated GIS (geographic information system) software. These systems are typically run on powerful workstations or high-speed PC's. These systems are often geared toward fast interactive graphical presentations and spatial analyses, allowing the user to get fast answers to what-if questions and other types of queries that can be best expressed by interacting with a map image. (SAS Institute has its own GIS package under development for these kinds of applications.) The applications which we have discussed in this paper, however, illustrate some important uses of TIGER which are appropriate for the more traditional world of the SAS DATA step and batch processing (or at least they allow it as an option.) Even if you decide that most of what you want to do with TIGER files you can do with a GIS package, being familiar with the structure of TIGER line files and seeing how they can be readily accessed by a SAS DATA step should provide valuable insight into their potential.

Although we have not had time to explore the possibilities as yet, it would seem that using a carefully-indexed TIGER SDL in conjunction with the Screen Control Language and its interface to the DSGI routines would be an excellent toolbox for someone wanting to write their own custom GIS applications in SAS.

SAS and SAS/Graph are registered trademarks of SAS Institute, Cary, NC.