CENSSYS - a system for analyzing census-type data

Oldervoll, Jan

Historical Social Research, Vol. 14 — 1989 — No. 3, 17-22

CENSSYS - A System for Analyzing Census-Type Data

*Jan Oldervoll**

The very first census of Norway was taken in the 1660ies. This census contains only the male population. The same is true for the next census, in 1701. In 1769 it was decided to count the women as well. This time the local authorities were asked to make the tables for their own area, and only submitted those. The government made the final statistics based on those aggregates. The same was true for the five censuses in the period 1815-1855. In 1801, 1865, 1875, 1891 and every 10th year from 1900 onward, there are regular censuses with an increasing number of variables on every individual in the country.

At the History department, University of Bergen, the census of 1801 was put into the mainframe. The database consists of records on 879 020 individuals. At the same institution, most of the 1660-material was put into the computer as well. The project is not yet finished.

At the worlds northernmost university, Tromso, there is an institution called *RHD*, or *Registreringsseniralen for historiske data* (Norwegian Historical Data Archives). Their main objective is to make the censuses of 1875, 1891 and 1900 ready for computer analysis. They have finished approx. 1 million records. Other institutions has put some hundred thousand records on disk. All together, there are close to 3 million records on individuals in computers of Norwegian historians. This is a vast source of knowledge about Norwegian history. It has been used by a variety of people, but more by family historians, local historians, schools and by people generally interested in their past than by professional scientists.

The main reason for this is that the former is content with unrefined material (lists), which is easily made, while the scientists want to do their own analysis.

This can be done. There are programs to do it on the mainframes in Bergen and Tromso, but not without problems. Few historians seems to cope with the mainframes, even if they seem to do well on their PC's. Most of the programs were written in the late 70ies, and may be inconvenient by today's standard. And if you are situated in a different institution than the data, there may be nobody to ask when you have trouble. The old systems seem to have a too high a learning barrier. And for most people the solution seems to be, do without it. This is a negative solution.

* Address all communications to Jan Oldervoll, University of Bergen, Dept. of History, N-5007 Bergen-Universitet, Norway.

There are a positive solution as well. We can make the barrier dramatically lower. But then we have to make a completely new system for analyzing census data. To do this, there are a few requirements that have to be fulfilled:

- The system has to *run on a standard PC*, engined by anything from a 8088 and upwards. Even if many people have a 80286 or even 80386, there are still a lot of 8088 and 8086 around, and will be for quite a few years.
- The system should *use a minimum of disk space.* There are many reasons for this. One is that people *always run* out of space on their hard disk. Only data allowed to stay on the disk permanently or can used or loaded from a single floppy, will be used. Very often, you even saves time by reducing storage. To move data between disk and memory (I/O) is a very time consuming tasks in most applications.
- The system should be *very easy to run.* You shouldn't have to have formal instruction or study kilograms of documentation to use the system.
- The system should be able to *run all censuses:* You should not have to learn a new system every time you change census.
- The system should do whatever you want to do to a census. This goal is difficult to reach. But it is more important to *reach for* the stars than actually getting there.

These could be called traditional requirements. I have still another. Before having a closer look at that one, a few words on how a census is used is necessary.

A census is a description of a *society.* As everybody knows, a society is a extremely complex structure, while a census is a fairly simple one. A census is containing information on a society, but only an infinitesimal part of all information possible on that society. People is reduced to a name, an age, an occupation and a few more variables of the kind. Interaction between people is reduced to who is living with who, and who is the son of who. A truly enormous reduction!

To make statistics based on a census, you need to *code* it. 1 the 1801 census of Norway, there are approx. 50,000 different ways of expressing what occupation an individual had. To make sensible statistics, those have to be reduced to a few dozens or a few hundred. There are thousands of ways to express that a person is a *Husmann*(cottager). By coding you may reducing this to one. You may loose the fact that he also may have done some fishing. Even different spelling of the same word may carry information. Some Husmann is spelled Huusmann, with a double »u«. It could be that the one's with double has a higher social standing that the one's without. Coding is another vast reduction of information.

Cross tabulation is a frequent statistical analysis of census data. Even tabulation is a reduction of information. If you make an age by civil status table, you loose the information on occupation for the individuals behind the figures in the cells of the table, which may be vital. Or, if you find a female bishop in a sex by occupation table, is this a sensation or is it a typing error? From a table you have no way of knowing.

You can say that making statistical analysis is a step-wise reduction of information. And what has been taken away is gone forever. In most systems it is cumbersome to say the least to have a look at the individuals behind a cell in a table. It is my sincere conviction that it is extremely important to do so.

My third requirement is that from every step in the process from census to table, you should be able to trace back to the census. When you code you should be able to see what terms hide between the code for *cottager,* or you could even look at information about the individuals with that code. When you make a table, it should be possible to open a window where you can look at the individuals belonging to every cell.

In short, I want a system that is easy to use, compact, fast and where the different levels of the census (text, codes and table) is integrated. I have not found any commercial products that even come close to the fulfilling of my demands. It has to be written.

When you are going to make a new system, the most important thing is to make an efficient data structure. And efficiency must be measured according to what you are going to do. If you are going to find a particular record, the fastest way to do is probably to have an index-sequential data structure of some kind. But if you want to make a cross table, you don't need any indexes, and a regular sequential file would as fast. And fastness is only one part of efficiency. Compact storage is another. Index-sequential files are disk consuming. If the major task is to make tables, it is better to have a sequential file, and pay the price when you once in a while are searching for a certain record.

Depending on the data, the price does not have to be too high. In the coded data, the value of every variable consists of a single integer. One individual consists of a fixed number of integers, constituting a *vector.* The whole census can be interpreted as a *matrix.* Copying the matrix into the memory and searching it for a certain value is an extremely fast task. But it can be speeded. In a machine, a matrix is not stored as a matrix, but as a number of vectors chained onto each other. The logical way to store a census, is to store the material as chained vectors each containing a person. If you want to search for occupation, you have no need of any of the other variables, and you are consuming much time moving unnecessary data into RAM. If you store the data as vectors containing all the values of a single variable. The vector will then have as many dimensions as the num-

ber of persons in the census. The census will then be stored as vectors chained together, one for every variable. This very simple remedy will save a considerable amount of time, and make it possible to search tens of thousand of individuals per second on a regular AT. If you want to make a cross table, you save as much, because you only have to read the variables used. A simple, but extremely efficient remedy.

To conserve even more space, and I/O time, one can even pack data. Sex only needs one bit of storage, age 7 bits, and both can be stored in one byte. If there are 100,000 persons in the census, there are 100,000 bytes of storage saved. And there are other variables where you can save.

This is a simple, very compact and from extremely to moderate fast way of storing coded data.

It would nice if one could do the same with the text version of the census. One could store it as a text matrix, many systems does, but this is a disk and memory consuming way of doing it, because you would have to make room for the maximum length of the variable. But if one returns to the coded version for a moment, there is text involved in that one as well. When a person has occupation 18, the number 18 points to line 18 in a code book, which may have the text »Farmer«. The code book may or may not be in the machine. If one made a large code book that contained every single term occurring in the census, the census could consist of pointers to this »code book«, or list of terms. In that way every variable could be reduced to 2 bytes per person, and they could be nicely organized, like in the coded version. It would probably take something like 1/5 or 1/10 of the space, compared to a regular text file. The compactness requirement is fulfilled.

In addition there is the list of terms. But since very many terms are repeated very many times in a census, and we only have to store it once in a list, this list does not take much space. In fact, even for a large area, it is stored permanently in RAM. To find a person called Hans, would be to search the list for the word »hans«. This is done in fractions of a second. The next step would be to find all pointers pointing to Hans: My AT can search close to 100,000 records per second, if the data is in RAM, 40,000 if they must be fetched from the hard disk. In most cases, this is acceptable. And if it should be to slowly, there are technics that could speed it considerably. The price to pay is some more disk storage. I have not as to yet found it worth while.

Now we have a very compact data structure, which is essentially the same for both text and coded versions of the censuses. The structure consists of a list of terms and a set of pointers to list. All programs running on the text version can also run on the coded version. The coded version is in fact a text version where the text has been reduced to those terms defined by the code book.

What can we do with this data:

- It can be searched. But searching can be done in different ways. One is what I call blind search. You ask the computer to find for example the person *Oluff Lauritzen*. The problem is that his name can be written in many ways, for example *Ole Larsen*. You can never be sure that you found the right person. A better way is to present you a list of all first names in your area, and let you choose among them. Perhaps you choose *Olaf, Olaff, Olav, Ole* and *Oluff*. Then you can ask for a list of last names. Then you will get only the last names of the persons with the first names you have chosen, which should be a rather small list. Here you can have look at the persons with the last name of *Larsen* and *Lauritzen,* or you can choose them and ask for another variable, for example age, and then look at persons within the right age span.
- You can code it. Coding is collecting groups of terms and giving them the same characteristics, giving them a code. This is done easily just by changing pointers. While the *Husmann's* in the text version have pointers pointing to a lot of terms all meaning *Husmann*, this are all changed to the pointer pointing to the term *Husmann.* This can be done by the same procedure that are doing the searching. Searching and coding is the same thing, seen from the user's point of view. This helps make the program easy to use.
- By keeping the old pointer as well, it is very easy uncode both a variable, a term or a single person. The only thing to do is to change back to the old pointer. This is easy to do technically, but also for the user.
- At all the time you can also see the persons that hide behind a term or behind a code. You even get the household he lives in. This is done just by pushing a key.
- You can make a subset. People living in a certain area, having a certain name or certain occupation or whatever. This is done by the same procedure as search and code procedure. Easy to use. You can print a subset, copying it to a file or use it for further analysis.
- You can make cross tables. You can tabulate text version or coded version as you wish. You can edit the table, and write it to printer or disk. You can even open a window in each cell, looking at the individuals behind the figures, and even move them from the cell to another. This is a very important characteristic of the system.
- The tables can be made based on the coded version, but also on the text version as well. In most cases, the text-based tables are very large, but otherwise they behave like a table based on the coded version. You can even make your coding directly in the table. This is useful if you do not want a permanent coding.
- There should be powerful statistical and graphic tools in the system. I

am not going to make them for the time being. Instead, I am going to make good interfaces to packages.

What about the future? In my opinion, the most important task is to make good linking capabilities to link information from several censuses. The data structure is very good for that type of task. To link, you need a coded, or standardized version of the census. The standardization may be done beforehand or while the program is running. The safest is to do it beforehand, and my system has the capabilities. To do linking you need a compact data structure because there are so much data to consider. You need good searching capabilities. My system has it. Making good record linkage system should not be too difficult, and I hope to do it.