

Computer Methods in a Study of Racial Profiling by Police

James W. Dow
Oakland University

Introduction

In a modern society, enormous quantities of data are recorded for purposes other than social-science research; however such data can be often be utilized in such research to provide valuable unobtrusive measures of variables that could not be measured otherwise. The people managing the data may be eager to have it analyzed but may be unable to provide it in the form that the research requires. If one wants to utilize such data, it is usually necessary to transform it.

The prospect of transforming data in the era before computers was a daunting task, but this has changed. Today, the data is usually digital and can be read by many types of computers. However, it is more inaccessible to humans because it has been formatted to be read only by computers. Ultimately, humans get the data, but only after a computer program has put it into a human-readable form, but usually this is not the form in which the researchers need the data. If they want to see get in another form they have to transform it to meet their research needs. Fortunately, because the data is digital, the transformation can be done on a computer. However this requires computer skills that are not widely available. Either the researchers have to find the skills or hire a person with them. If this can be done, countless hours of work can be saved, and research that was previously impossible can be carried out.

This article describes a case in which data archived by a city police department was transformed for analysis. The computer experts in the police department were familiar with the operation of their computers but were not able to provide the data in a form that the researchers could use, although they were willing to loan their archive tape cassettes to the researchers.

Racial Profiling

The goal of the research was to understand how patrol officers perceived potentials for criminal activity. In particular, the study wanted to see if the race of a subject influenced the police behavior. This is usually referred to as "racial profiling." The principal investigator in this project was Albert Jay Meehan in the Department of Sociology and Anthropology at Oakland University.¹ The police department involved was a confidential collaborator whose identity must remain anonymous. Myself, an anthropological colleague in the department, was welcomed as a minor participant in the project when I suggested that I might be able to get important data directly off the computer tapes that the police department was creating for its own use.

The research project is ongoing but has reached the point where conclusions can be drawn about the nature of the racial profiling of car drivers by patrol officers. Drivers near a boundary with a Black residential neighborhood are no more likely to be investigated than white drivers; however Black drivers in pure white residential neighborhoods are three times as likely to be investigated as White drivers although they are no more likely to have criminal record. The sociological aspects of this situation have been investigated by Dr. Meehan and he believes that the police are responding to the desires of the White residents to protect the boundaries of their residential areas from incursions by Blacks.¹ My involvement was strictly with the computer methodology.

Digital Communication in a Police Department

Technology in the police world has made rapid advances in the last decades. Today, much of the communication within and between police departments and law-enforcement agencies is digital. Most of it is digital text, and, in the future, one might reasonably expect to find digital voice and video as well. This paper deals with the first step in data analysis, getting the data out of the client's computer and into a

form that was usable for research.

The communication that was analyzed in this project were the messages that were passed between mobile data terminals (MDTs) in patrol cars, fixed terminals in the police stations, and systems at other law-enforcement agencies. This communication encompassed all the textual law enforcement messages within, to, and from the police department. Many messages were sent to and from external law-enforcement databases to which the department had access. The data contained queries that patrol officers made about persons identified by car registrations, licenses, dates of birth, or names. It also contained personal and official typewritten messages passed between officers in cars and at police headquarters. The most significant and revealing communication between the officers went by this digital route, because they knew that their police radios could be overheard. Furthermore, since the messages were read only by the recipients, private conversations were also sent through the digital terminals.

The police department archived all this information continuously day by day on 4mm computer tape cassettes. Each small 4mm tape cassette could hold two gigabytes of data, enough when printed to fill 4 million sheets of paper. The problem for the researchers was that messages passing between any of 60 or more stations in the department and to and from dozens of federal, state, and local law-enforcement information services were jumbled together and interspersed with non-textual binary data. This information had to be unraveled and organized so that it could be used to analyze police behavior. No computer programs or computer packages had ever been written to do such a job.

Using the Digital Data

The first problem was to get the digital information off a 4mm tape cassette. The police department was only able to tell us that the tape contained records of the MDT communication and that it was written by a computer with an SCO Unix operating system.² We hoped that the tape contained a Unix *tar* archive.³ Computers will almost never write a tape or disk in a format that is not a standard for its operating system. Hardware may differ from computer to computer, but the same type of media will usually be formatted in a standard way. Some media may take several formats, but there is usually only a very small set of standards. This standardization has been achieved by the computer industry over the last thirty years during which frustrations with incompatible devices had to be resolved. In this case, we found a university computer with a 4mm tape drive, and the data was read off the cassettes without difficulty, although the make of the computer, the make of the drive, and the version of Unix were different.

The archive was quickly transferred over the university network to my office computer. All residues of its existence on the other university computer were erased for security purposes. Computers have varying security levels and those attached to university computer networks are open and vulnerable to hacking. Any escape of the data would have jeopardized our relationship with the police department, so security was a high priority. Only a small fraction of two tapes had been used to record five months of communication. The process was repeated with a second tape and the archives were then moved to a magneto-optical disk and erased from my office computer.⁴ A zip disk could have been used for this step.

The archive was taken to a home computer for analysis. The home computer was used because it was in a private location, not connected to a network, and, thus, much more secure. The home computer was a standard PC that could be booted to run either a Windows operating system or a Linux operating system.⁵ Analysis was done on the Linux system for two reasons: (1) the archive had been written on a computer running a Unix operating system and would most easily be unraveled with Unix software, and (2) Unix operating systems contain excellent computer languages for text analysis.

The first problem was decompressing the *tar* archive and extracting the files from it. There are standard Unix programs for doing this. The program *tar* is the most common one. The extraction could also have been done on a Windows system with the program *WinZip*. On examination, it was obvious that the files were labeled with the year and a 1 to 365 day number corresponding to the twenty-four hour period during which they were recorded. Human beings tend to create logical symbolic systems to communicate with each other. Information put into a computer is usually the last exception to this rule, because computers must operate on strict logical principles. I examined the files with a binary editor that read each eight-bit byte in sequence. I saw textual gibberish indicating that much of the data was binary and still had to be interpreted by a computer program the details of which we and the police computer

experts did not know. Knowing how to run a computer program is one thing. Knowing how it operates inside a computer to accomplish its tasks is something entirely different.⁶ Thus there are many types of “computer experts.” We received no information from the police department on how the archive was constructed and did not want to upset our excellent working relationship with them by asking them to get the source code for a program that they had probably purchased without the right to have the code. Neither of us knew how the interpreting program worked. However, I expected that this would not stop us from using the data because the text of the messages had to be somewhere in the files.

Further examination with the binary editor revealed that the entire textual content of the messages was embedded in the files and encoded in standard (ANSI) eight bit characters. It could be easily read with any one of many text editing programs. The text, was very comprehensive and indicated the time each message was sent, to whom it was sent, and who sent it; therefore the binary data became irrelevant to the research. The messages were recorded in the same sequence as they had been sent. The first message after midnight was at the beginning of the file and the last message of the twenty-four hour period was at the end of the file.

It was discovered that all the messages were preceded by a unique four byte string, probably interpreted by the police computer as an unsigned integer, and terminated by the standard Unix end-of-string character, byte zero. Pulling the messages out of each file with an editing program would have been very tedious. Each day produced from 1.4 to 3.5 megabytes of data. If printed, this would be from 3000 to 7000 pages. Therefore to be practical, I had to extract textual data from each file with a computer program. Reconstructing the database program that was supposed to interpret the files would have been a wasted effort, because we did not need the binary data that such a program might have interpreted. So, a program that searched for the initial four byte string and extracted the text up to the final end of string character was written. The C++ language,⁷ the basic Unix compiler language, proved to be a useful language in which to write the program. The program also performed some elementary formatting to make the output more readable. Each message was separated by a blank line from the next, and multiple spaces were reduced to one. Figure 1 shows this program. Once it was written, the program took only a few seconds to extract the messages for one day.

The result was a readable text consisting of all the messages in the order that they passed through the central computer. The messages were filled with names, license numbers, personal information regarding contact with government agencies, personal comments by the police officers, chit chat, etc. They cannot be illustrated here for ethical reasons. The sizes of the daily message files ranged from 1 to 2.1 megabytes.

The next step in the extraction process was to put the messages into a form that could be utilized by the research team. Previously, the project had not been utilizing computers except for the final phase of statistical analysis. Printed messages were being visually compared to dispatcher logs to determine whether the patrol officers were acting on their own in querying license plates or whether they were responding to a call that someone else had made. This proactive-reactive distinction was important to understanding how the officers might be profiling subjects on their own. Dispatcher logs were available on another tape, but the team members wanted to proceed with the manual checking. They felt more comfortable with printed text. With more work, an automated comparison of dispatcher logs with the messages could have been developed.

Team members decided they wanted to read the patrol officer's queries of law enforcement databases and read the results of these queries. These items were mixed into the stream of messages and had to be extracted and printed for the manual comparisons. I decided to start with the simplest program languages available, the Unix shell programs *sed*⁸ and *awk*.⁹ They were sufficiently powerful to do the job, and I did not have to utilize more powerful text manipulation languages such as *perl*. Versions of *sed* and *awk* are also available for DOS, but they are not as well integrated or developed.

Utilizing small programs that were written and tested in a command window¹⁰ I put together a program that extracted all the messages sent and received by each of the patrol officers. The Unix system allowed these small programs to be grouped together in batches so that the final processing of a day of communication took only around ten seconds. The text of all the messages to and from a single officer were grouped in separate files. A final program, called “person-queries.bash”¹¹ (See Figure 2)¹² extracted

the queries of license plate numbers made by each patrol officer and the responses to these queries from the law enforcement data bases. Team estimated the race of subjects by the car owners residence. They also noted any criminal records that showed up. For analysis it was assumed that if the registration was in a predominantly Black residential area there was a high probability that the driver was Black. This assumption was tested and proven correct by independent observations of drivers and license plates in the same general area.

The program “person-queries.bash” relied on two another Unix programs, “compersons.bash” and “identify.awk” that prepared a list of codes identifying the patrol officers. Figures 3 and 4 show these programs. Unfortunately, reading these programs can be like reading a foreign language. They all look like gibberish unless one knows the computer languages. I have included the programs here for readers who might be interested in these details. They may be useful as guides for someone who needs to do a similar project. However, the languages are not difficult learn. They are all described in an easy-to-read introduction by Kernighan and Pike (1984). These two authors helped to create these Unix tools, many of which have now been reprogrammed for MSDOS.

Conclusions

The vast amount of digital data now being collected has opened up new possibilities for the social scientist; however a methodological gap has also opened. New methodologies are more related to computer programming than to statistical analysis. This paper has explored the methodological problems involved in one research project using digital data.

Figure 5 diagrams the procedures that could be used to deal with digital data. The heavily outlined blocks show the steps taken in this project. They were:

1. Obtain the media from the client.
2. Transfer the data from the media to computer files.
3. Examine the content of the computer files and find the data that is relevant to the research project in question.
4. If textual data is most important, extract the relevant text from the computer files with whatever computer programs are appropriate. C++ programming may be necessary, because the data may have been written by specialized programs.
5. Restructure the text so that it can be used in printed form or as input to other lexicographic programming.

The other blocks in the diagram point the way to potential analysis of binary data that were not carried out in the example.

Serious ethical issues can be intertwined with the technological problems. The people from whom the data is gathered may not have given permission to use it in research; thus, contributors may have to be contacted for permission. Often the data can be made anonymous to a point where the analysis has no possibility of harming the contributors, the primary ethical consideration; however one has to be very careful that the underlying digital data is carefully guarded and perhaps destroyed if necessary. On the other hand, the research is usually consistent with the goals of the people who control the databases, otherwise permission to use the data would not be obtained. The researcher must make sure that their needs are also considered. They usually will not want the raw data to become public, so it has to be protected. Digital data is harder to guard than physical data, so care must be taken to shield it from computer hackers. We were successful in guarding the data by keeping our computer in a controlled location and off networks.

Figure 1: C++ program for extracting the textual data from the communication archive.

```
#include <stdio.h>
#include <ctype.h>
main(argc, argv)
    int argc;
    char **argv;
{
    char c, cl; /* Character and preceding character */
    FILE *f1, *fopen();
```

```

/*Read the file name from standard input*/
if((f1 = fopen(argv[1], "r")) == NULL)
    {printf("I can't open %s\n", argv[1]);
    exit(1);}
/*Read the file and print the good parts*/
while(-1)
    {c = newc(f1);
    /* Look for beginning sequence which seems to be an octal
    integer \000042 */
    /* Check for a " */
    if (c == '"') c = newc(f1);
else continue;
/* Finally check for a null */
if (c == '\0')
    /* The beginning of a text record has been reached
    Store the character in the preceding character
    spot */
    { c1 = c;
    /* Output the text */
    while ((c = newc(f1)) != '\0')
        {
        /* Remove extra spaces */
        if((c == ' ') && (c1 == ' '))
            {
            c1 = c;
            continue;
            }
        putchar(c);
        c1 = c;
        }
        putchar('\n');
        putchar('\n');
        }
    else continue;
    }
}
/* New character and check for end of file */
newc(f1)
    FILE *f1;
    {
    char ch;
    ch = fgetc(f1);
    if (feof(f1) != 0)
        {
        fclose(f1);
        exit(0);
        }
    }
    return(ch);
}

```

Figure 2: Program to produce queries of law-enforcement databases by patrol officers and the responses to those queries.12

```

# Syntax
# person-queries.bash <day>
# Argument 1 is the day number <day>.
# The particular persons will be in a file <day>.txt.pl
# The list <day>.txt.pl is set up first by running
# compersons.bash

```

```

# Make the list of individuals
compersons.bash $1.txt
# set up an empty file to receive the data
echo '

Individual queries and responses during day' $1 >$1.iql;

# Look at the list of individuals.
for i in `cat $1.txt.pl`
do
# Print a header
echo >>$1.iql;
echo '***' Queries and responses for officer number $i '***' >>$1.iql
# Print out the lines for those individuals followed by
# a blank space. Note that the $i cannot be inside the
# single quote because it has to be exposed to the shell
# for interpretation
# Note that there are no blank lines between the items
gawk '/[:;/ ]'$i'\/XXXX/ {print $0}' $1.txt >>$1.iql
done

```

Figure 3: Program to make a list of the patrol officers on duty during a particular day

```

# Shell script to form a sorted person list
# $1 is the input text file. Output is to <textfile>.pl
# Gawk program to get person numbers from the text file
# Syntax
# compersons.bash <textfile>
gawk -f identify.awk $1 | sort | uniq >$1.pl

```

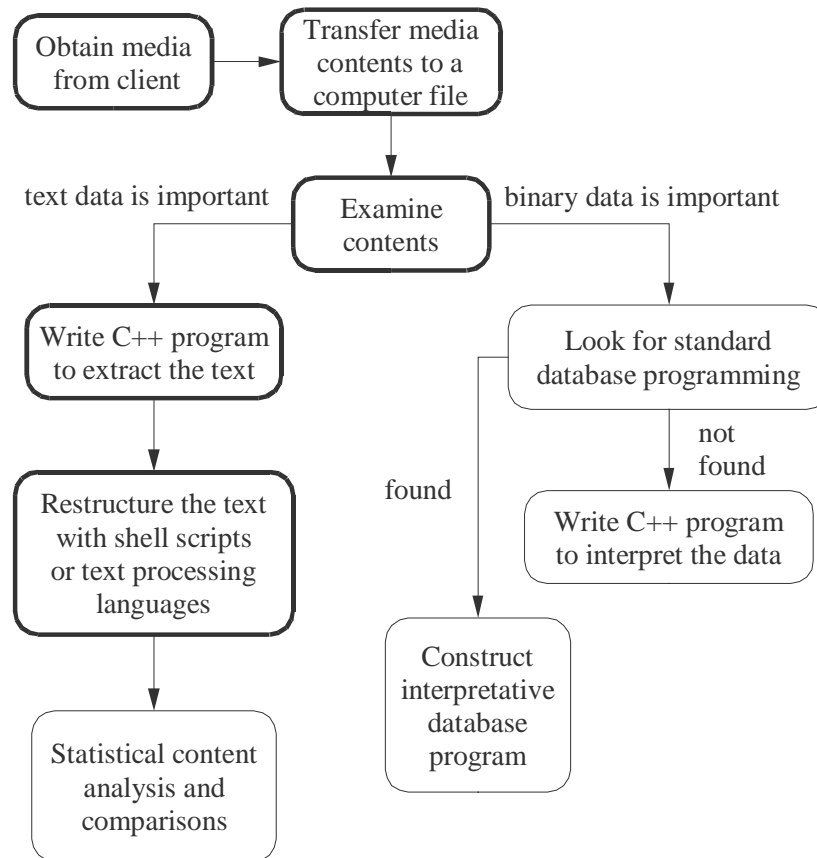
Figure 4: Program to identify the codes of the patrol officers in a daily communication text.12

```

# Awk program to identify the codes of the patrol officers.
# Changed to deal only with numbered officers
/\XXXX/ {
  n = match($0,/....[0-9]\XXXX/)
  a = substr($0,n,(RLENGTH-5));
  # Take string after any ";", ":", "/", or space
  n = match(a,/[[:;/ ]+$/ )
  while (n != 0)
  {
    a = substr(a,n+1)
    n = match(a,/[[:;/ ]+$/ )
  }
  # Don't print if nothing is left
  if (a != ""){print a}
}

```

Figure 5: Steps taken in dealing with digital data in social science research



Notes

1. For further information on the sociological aspects of this project see Meehan and Ponder (2002) or contact Meehan at the Department of Sociology and Anthropology, Oakland University, Rochester, MI 48309, or at meehan@oakland.edu.
2. Unix is an operating system for large and small computers. It is one of the oldest operating systems and predates Microsoft DOS and Macintosh OS. It was created at Bell Laboratories in 1969 (Kernighan and Pike 1984:vii)
3. *Tar* is simply an abbreviation of “tape archive” and has been a standard format on Unix computers for decades.
4. The office computer was an ordinary PC. The magneto-optical disk drive functioned like a zip disk drive, except that it used 3-1/2 inch magneto optical media. Reliable magneto-optical media have been used for over a decade, particularly in medical equipment. The data are written by a laser with a magnetic field and read by a laser. The data cannot be destroyed by water, moderate heat, or magnetic fields. Magneto-optical disk are the longest lasting digital storage media yet developed. For some reason unknown to me, they have not been widely marketed to the home computer user.

5. Linux is a common, open Unix operating system for PCs. It contains all the standard Unix programs. Open programs are freely distributed with the source code available. They can be contrasted with “commercial” programs which are sold and for which the source code is normally kept as proprietary information by the seller.
6. Computer users work with pyramidal technology. Each computer user learns how to operate programs written by someone else, but they do not understand the programming of those programs, although they may use them to write other programs. Thus, the magic of doing something with a computer is just the power of the encapsulated knowledge of dozens, perhaps hundreds, of programmers who have laid the groundwork one level after another for these things to happen.
7. The C++ language is a programming language describing procedures for the computer to follow at the level of operating system control and at higher levels. C++ is also the fundamental language in which many operating systems themselves are programmed. A suggested information web site is: <http://www.cyberdiem.com/vin/learn.html>. C++ can be compiled by the free GCC (Gnu compiler collection) which is available for Linux and Windows.
8. *Sed* is a non-interactive editor that restructures text. It is simpler than *awk* and operates on text streams. *Sed* was part of the original Unix command line programming system written of the original Unix operating system written at Bell Labs. A suggested information web site is <http://www.math.fu-berlin.de/~leitner/sed/tutorial.html>
9. *Awk* is a Unix language that operates on text files and can be included in scripts (like BAT files in MSDOS). *Gawk* is the open version of *awk*. *Awk* allows the restructuring of lines of text. *Awk* was part of the original Unix command line programming system written at Bell Labs. A suggested information web site is: <http://sparky.rice.edu/~hartigan/awk.html>
10. In Unix terminology a window for executing commands is called “shell.” A batch program in Unix is called a “shell script.”
11. The common Unix shell is called “bash,” which has a rather arcane derivation, “bourne-again-shell.” A program ending in the extension “.bash” is designated as a batch program or “script” for the *bash* shell.
12. In all the figures, characters that might identify persons or police departments have been replaced by Xs.

References

Kernighan, Brian and Rob Pike. (1984). *The Unix Programming Environment*. Prentice-Hall.

Meehan, Albert J. and Michael Ponder. (2002). Race and Place: The Ecology of Racial Profiling African American Motorists. *Justice Quarterly* 19(3):401-432.