Technical Report CS75020-R

Charcoal Iron Industry Analysis: Data Preparation and Computer Program Design

David A. Ault

Richard H. Schallenberg*

Department of Computer Science

*Department of History

College of Arts and Sciences
Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061

September, 1975

ABSTRACT

A report of the use of the computer to analyze historical data on the charcoal iron industry is presented. It includes a discussion of the problem of analyzing historical data by computer and the use of the computer to locate errors or inconsistencies in the reporting of the original data and in the transcription process from document to punched card. The type of analysis applied to the data is described and the methods of program construction and verification are explained. The results of this analysis are reported in a separate paper [13].

KEY WORDS: charcoal iron industry, data reliability, analysis of historical data, program design, program testing.

COMPUTING REVIEWS CATEGORIES: 3.49, 4.33

INTRODUCTION

One of the most crucial technological innovations which initiated the industrial revolution of the late 18th century was the substitution of coal for charcoal as the reducing agent in the blast furnace [7, p. 143]. Until the 18th century all iron was smelted with charcoal. However, when coal smelting was introduced, it created the potential for an almost infinitely expansible production of iron. The emphasis here is on the word potential: the new coal-smelted iron was different metallurgically from the traditional charcoal metal and initially there were many problems in integrating it into industrial processes.

It was for this reason that the charcoal iron industry continued to survive long after the initial coke-smelting process was invented, even though coke smelting soon proved considerably cheaper than the charcoal method [12]. The longevity of the older technology was particularly great in countries which had large timber reserves, such as the United States. The last American charcoal blast furnace shut down as recently as 1945. Nonetheless, as soon as the coke-smelting technology began to catch on in a particular country, and as soon as methods were worked out for integrating coke iron into finishing technologies, the charcoal ironmasters felt a slow but increasing competition from the new iron.

As a result of this competition, charcoal ironmasters began to engage in a considerable degree of technological innovation in an attempt to keep their product economically competitive with coke iron.

Since the smelting of iron with coke was introduced into the U.S. in the 1840's, it was in this period that the American charcoal iron industry began to experience major innovation. It was our intention to examine in detail the pattern of such innovation as a case study of the dynamics of an industry facing severe technological competition during a period of rapid technological change.

In order to do this, we sought technical data on American charcoal furnaces which operated during the period 1840-1945. Detailed data does exist on a handful of the many hundreds of late 19th - early 20th century charcoal blast furnace plants, but it is not at all clear *a priori* that these furnaces were representative of their time or locale. The only way to insure a perfectly representative sample of furnaces for particular periods during the hundred years under study would be to have a census of all charcoal furnaces existing at particular dates. Fortunately, we do possess such censuses.

In 1859, J. P. Lesley, the president of the American Iron and Steel Association, began a pioneering census of all iron works in the U. S. operating in that year [10]. Unfortunately, this was a once-only census and it was not until 1874 that the American Iron and Steel Association began the yearly publication of a census which was a continuation of Lesley's work [1]. The Iron and Steel Association's *Directory* has been published every year since then.

Unfortunately, a very serious problem with the use of such data is its lack of reliability. As with most pre-twentieth century statistics, the amount of error and incomplete data proved to be enormous.

Therefore, before creating a program for the analysis of the data, it was first necessary to write several preliminary programs to try to eliminate as much error from the data as possible, and, in effect, to reconstruct much technical information which was not in the original data. In fact, this preliminary work consumed the vast majority of all time spent on the project and presented to us a series of very intriguing problems.

Our objectives in making this report are to present our goals, plans and problems in using computer assisted analysis of historical data. The results of this analysis and its application to the study of technological changes in the charcoal-iron industry are presented in [13]. We hope this paper will aid others to more easily and effectively plan their projects so that they may make use of computer analysis of their data. The areas of concern to us are the proper gathering of the data from historical documents, the types of analysis that might be made on the data, the types of errors to anticipate in both the original data and in the transcription process from the documents to a computer file, and several techniques that may be applied to locate errors in the data and in the programs. The techniques for locating errors in the data include the use of redundency in the data, sorting the data on a number of variables to locate either similarities or discrepancies, and specific checks on individual data items where some prior knowledge is available on the expected characteristics of each item. The three principles to apply to assure reliable programs to analyze the data are the choice of

algorithm, its implementation and the techniques of testing the program.

DATA FORMAT

Each record consists of twelve fields as shown in Figure 1. These fields contain data for one furnace reported in one issue of the Directory. Each occurrence of a record may or may not contain information in each field. The first three fields are the furnace designator. These fields contain the state and county in which the furnace was reported and the furnace name. The fourth field is the census date, that is, the date in which the accompanying statistics were recorded for this furnace. The fifth and sixth fields contain the furnace height and width, measured in feet, at the given census year. The seventh field contains the annual furnace capacity for the production of pig iron, measured in tons. The eighth field contains an indicator of the temperature of the furnace blast and is recorded as H (hot), C (cold) or W (warm). Fields nine through twelve contain dates that the furnace was built or rebuilt. They are recorded in descending order, starting in field nine, i.e. field nine contains the latest date. Unused fields are left blank.

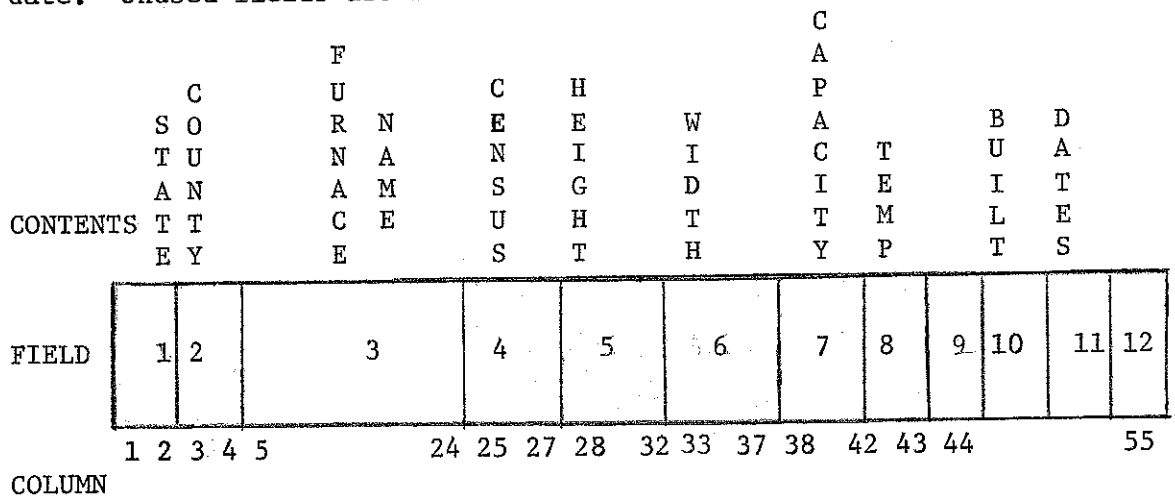

Figure 1. Record format for blast furnace data.

The ordering of the fields in the record can either complicate or simplify the programming process, depending upon the relative placement of the fields. In large data manipulation systems, it is common to have the data stored in more than one record format and possibly to have more than one numeric representation for a given field in two different versions of the record, e.g. character or integer representation of a numeric field [6, p. 7]. In a smaller data analysis problem of the type discussed here, a careful planning of the arrangement of the fields can make the programming and data handling much easier and more efficient. It is possible to rearrange the order of the fields within the record after the record image has been read, but this means that the programmer must always remember which ordering of the fields he is working with. One method of overcoming this problem is to develop one routine which will convert a record from its existing format into the one expected by the programmer [6, pp.11-14].

Having similar fields together, e.g., the four rebuilt dates in fields nine through twelve, enables the programmer to refer to these fields as an array. This allows the manipulation of these similar elements in a loop, i.e. a DO loop or a WHILE loop. This has several advantages to the programmer and ultimately to the project's cost and the accuracy of the results. Using an array reduces the amount of program code that must be written and compiled. Performing the same operation on a sequence of data elements in a loop makes verification of the correctness of the program easier because the correctness of the code need only be established once to assure that it operates

correctly on all of the array variables. This also reduces the possibility of unintentionally misnaming a variable, e.g. spelling the variable incorrectly so as to produce a variable whose properties have not been defined in the program or using the correct name of another variable, thereby introducing the wrong value.

HISTORICAL DATA PROBLEMS

The data was collected from fifteen issues of the Directory. The census years are 1859, 1876, 1880, 1884, 1886, 1890, 1894, 1898, 1901, 1904, 1908, 1912, 1916, 1925, and 1930. The census years are unevenly spaced because the data was originally collected without the idea of using a computer to analyze it. In order to check for a complete set of records for each furnace between its original built date and the last census date in which the furnace was reported, the more general procedure of using a table of census dates in increasing order as a comparison against the census dates available for each furnace was followed. This method has the advantage that new census dates can be added by simply adding them to the appropriate place in the table. No changes need to be made to the program logic.

The main problem with this historical data was its reliability. Initially we tried to connect identical furnaces reported in different census years by assuming that each furnace would be consistently reported by the same name and would be reported as being in the same state and county. This proved to be much too simplistic. In some cases there were variations in the spelling of the name and other

means of designation of the furnace. Furnaces were sometimes reported with their counties and sometimes without. Further, due to the shifting of county boundaries during the more than a century time span of the study, we suspect that many furnaces might actually have changed their county location without being moved. Also, since it was common practice to name a furnace after the county in which it was located, the problems of identifying the same furnace in different years were multiplied. A name change can also occur when the furnace is named after the owner and the furnace is sold. As one might expect with historical information, some furnaces had incomplete data reported for the height, width, capacity and blast temperature.

DATA EDITING

Three basic types of errors in the data were expected. The most obvious types of error are missing values and ones which are not members of a small set of expected values, e.g. incorrect values reported as census dates or as blast temperatures. The other type of error is a keypunch error which places a nonnumeric character in a numeric field. This latter type of error is located by reading each record as a character string to ensure that the entire record is removed from the input stream. Then the data is extracted from this character string and either subdivided into character data or converted into decimal numeric data. If an error in the conversion of the character representation of a number to a machine representation of that number occurs because

of the existence of a nonnumeric character in the number field, a conversion error is raised and the number field, the offending character, a code indicating the type of conversion error and an image of the record are printed. The offending record is not included in the new file.

The new file is sorted on a key selected by the programmer and examined for a number of specific errors. First, duplicate records are spotted by saving the designator and the census date from one record to the next and by comparing the corresponding fields for equality. The fields corresponding to the state in which the furnace resides, the height, width and the capacity are checked for nonzero values. The blast temperature code is checked to be sure it is one of the three allowable codes, or possibly a blank. Finally the rebuilt dates are checked to be sure that they are recorded in the proper order. If at least one of the above errors is found, then the record is printed and all errors are listed below the record.

One error that should have been anticipated by us while writing the editing program, but was not, is a numeric value which is out of its expected range of values. This may be caused by a mistake in transcription from the Directory or by a keypunch error, e.g. mistake a 7 for a 1. This could have been caught for census and rebuilt years by testing to be sure that all dates lay between 1700 and 1945. A related problem occurred with the range of values of the height and width data during a period that we were using PL/1 picture variables defined on the character image of the record. The values were punched correctly on the original data cards, but without the zeros punched to the right of the decimal point. For example, if 9 feet is punched as 9.ϕϕ,

where ∅ denotes a blank character, then this number is interpreted to be .09, instead of 9.00 when used in arithmetic calculations. This phenomenon is unexpected and causes errors in calculations which are hard to find. Further, a number which is punched in the correct field, but in the wrong columns within that field will be misinterpreted when using picture variables. It is for these reasons that the option in PL/1 which allows one to read from an internal character string was used to translate from character representations of numbers to binary coded decimal digit representations, even though it is more time consuming. Using this method the numbers are converted correctly inspite of human errors in recording their character representations.

Another unexpected error was corrected by the ability to sort the data on the basis of different keys. The data was first sorted on the basis of a key consisting of the state, county, furnace name and census date. Note that this will group the furnaces first by state, then within each state by county. All records with identical furnace names will be grouped together and ordered by census year. Many gaps in the data about each furnace were produced by the fact that some census years omitted the county specification and also by the problem of varied spellings of furnace names from either the Directory itself or the transcription process. Sorting the furnaces on the same basis except for ignoring the county specification produced a much better grouping of the furnaces. It did demonstrate, however, that the county specification was necessary, because there

were furnaces by the same name in the same state, but in different counties. In checking the data with this ordering we noticed that there were still stray records for many furnaces. We suspected that a recording, transcription or keypunch error could have placed the records of a furnace in different states. To bring these records together, the file was sorted for a third time. This time the key consisted of the furnace name, census date, state and county. This showed that the principle mistake in the state code was the miscoding of similar letters, e.g. E and F, O and Q, M and N, and P and K. We were able to tell that a furnace belonged in an ordering of other furnaces because of the redundancy in this set of data. The furnace name was included in each record and the census year provided a form of redundancy because a missing census year for a furnace in one state can be filled by a record for a lone furnace by the same name with that census year in another state. Also, the height, width, and rebuilt dates provided a second, but less reliable, means of comparing two records. When taken together, it was often possible to positively identify a stray record and to correct it so as to place it in its proper position.

DATA ANALYSIS

After the editing process was completed, the corrected data was sorted again on the key consisting of the designator followed by the census date. This produced an ordering of the furnaces by state and

by county within a state, although this latter property was never utilized. Further, this collects the several occurrences of a record for a furnace together and orders them by census date. This ordering creates a file which can be processed sequentially to produce statistics on the furnaces by individual state, one state at a time, and for the whole nation. If the furnaces were not ordered in this fashion, many arrays would be needed to accumulate the statistics on the states in parallel. This would take a great deal more memory space. The accumulated statistics are based on an analysis of the data available for each furnace. To accomplish this, the furnaces are read from the sorted file, one furnace at a time, and an attempt is made to use the census dates, the built or rebuilt dates and the height and width to determine the significant events in the furnace's life: the first date it was built (its birth), the dates that it was rebuilt (if any) and the date that it was last known to exist (its death). A furnace's birth is assumed to occur at the earliest built date or the earliest census date if no earlier built dates are recorded. Its death is assumed to occur in the last census year that it is known. This is certainly a source of error, but we have no data indicating the furnace's last year of operation, because if it was not in existence during the next census year, its statistics were not recorded. From its birth and death, the furnace's life span can be calculated. A check is made at this point to be sure that the birth and death of the furnace lie between the years that mark the beginning and the ending of the period of study.

One problem in analyzing the furnace data is missing census reports on a furnace between its birth (or at least the first census in which it is reported) and its death. This is checked by comparing the census dates on the existing records for the furnace with a ordered table of census dates. If a census year is missing or if a reported census year is between correct census years, an error message is printed identifying the missing or incorrect date.

A furnace is assumed to be rebuilt whenever its reported height or width changes by more than two feet. The margin of two feet was given to allow for inaccuracies in the measurement and reporting of the height and width. Further, a furnace may have been rebuilt to unknown heights and widths several times before the first census date. In this case the rebuilt dates can be determined from the first census record for the furnace. For example, if a furnace was reported in the first census year 1859 to have been rebuilt in the years 1846, 1825, and 1815, then this furnace was built and rebuilt in 1815 and 1825 to unknown heights and widths. We record that it was rebuilt again in 1846 to the height and width recorded in the 1859 census. If no built or rebuilt dates are recorded prior to the first census date, then this census date is taken to be the first built date.

The subsequent rebuilt dates are determined by noting the occurrence of the next height or width change for that furnace. If the latest rebuilt date lies between the current census date and the previous census date, then it is assumed to be the end of the last period before the height or width change. Otherwise, the current census date

plays this role. In either case, the period of constant height and width is recorded; the new height, width and rebuilt date are saved as the beginning of a new period and the process is repeated. After the last record for this furnace has been processed in this manner, a check is made to see if one of three possible situations exists. If the last recorded period ends prior to the last census date, or if a height or width change occurs at the last census date, or if there is but one record for the furnace, then this last (or single) period is recorded.

Once the periods of constant height and width are determined from the height or width changes, four general types of statistics are collected on the furnaces for the nation as a whole and three of them are collected on each individual state. The appropriate sums are kept to calculate:

1. The average height and width of the furnaces according to the years in which they were built or rebuilt,

2. The average ratio of furnace height to width of all furnaces alive in each year over the span of the study,

3. The average ratio of furnace height to width recorded in a matrix with rows consisting of intervals of furnace height and columns consisting of intervals of built or rebuilt dates,

4. The calculation for each census year of

   a. the average furnace capacity for each blast temperature,
   b. the percent of furnaces using hot, warm and cold blasts of air in the smelting process, and
   c. the number of furnaces with zero and nonzero reported capacities.

PROGRAM VERIFICATION

There are three basic components in the development of these

programs that insure that the data is processed correctly. These are the program design which includes the choice of algorithm and the use of structured programming concepts [5] in the implementation of the algorithm, the use of print statements controlled by programmer set on-off switches to print the value of important variables at strategic points in the program, and the use of test data designed to test the possible paths of flow through the program. This section discusses these components of the development in the context of each of the major procedures developed to analyze the data.

The data editing program consists of two main sections: the sort-merge routines and the editing program. The records from the file are sorted by merging smaller blocks of internally sorted records. The internal sort program closely follows the well-known quicksort algorithm [9, pp. 116 - 117] with the addition of a few checks to prevent subscript range errors. This is an exchange sort which moves the records, or more generally the keys and pointers to the records, so as to obtain a division of the records into two partitions based on a key selected from the given records. At the end of each stage of this division process one partition contains those records with keys lower than the selected key and the other contains the higher keys. The process is repeated on the smallest of the two partitions until one or more partitions contains less than eleven records and these partitions are sorted by a standard bubble sort [9, p. 107]. In the bubble sort, adjacent keys are compared starting at the left of the partition and are exchanged if the left-hand key is greater than

the right-hand key. This moves the largest key to the right of the partition. The process is continued with the remaining unsorted keys until no exchanges are made. This means that the records in this partition are sorted. A record of the partitions which are not processed immediately is kept and the quicksort algorithm returns to these partitions until each partition and hence the whole block is sorted.

The internal sort was checked by using it to sort blocks of fifty records and these blocks were verified to be in sorted order. Since its only purpose was to sort records of blast furnace data, it was not checked on data which was already sorted or data with duplicate keys. Tests of this nature should be performed if it is to be a general-purpose sorting routine.

This internal sorting program is used in the merge program to build blocks of sorted records, which are written alternately to two files, until all of the records form the original file have been processed. These two files are merged together and new blocks of twice the original block length are written alternately onto two other files. When the first two files are empty, the roles of the four files are reversed and this process is continued until the records are sorted in one file. This method is known as the balanced two-way merge [9, pp. 247 - 248]. The program was originally written by Jay Benson.

Benson made good use of the TITLE option in the programming language PL/1 to alternately associate the four physical files with the two input and the two output files in a clever way so as to use the same program code for merging in each direction. The other complexity

in programming this merge is to record the reason an end-of-block is reached as two files are being merged onto one. An end of block may occur because of a step-down, i.e. a key smaller than the previous one is read from a file, or because the end of a file is reached. Depending on the order in which these end-of-block conditions are raised, there are five possible actions that the procedure may take: continue the merge from the other input file onto the same output file, merge two new blocks from the input files onto the other output file, after an end-of-file on one file and an end of block on the other, copy the last block to the other output file, reverse the direction of the merge, or end the merge because the file is sorted. The appropriate action to be taken in each case is determined by the use of a finite-state transition table with states to identify the current history of the merge and input values to identify the cause of the step-down. This method assures the program designer that each of the possible cases has been handled. Testing the program involves verifying that the setting of the variables in each case does, in fact, cause the proper action to take place in the merge.

Other than checking for conversion errors in the data during the input phase of the merge procedure, most of the data checking is done by the editing routine. This procedure was designed to test for specific errors on each data card. Any errors which are found are listed along with a copy of the record. Testing this procedure consisted of submitting it to a sequence of data cards containing one or more of the anticipated errors and checking to see that the procedure correctly reported all of the errors.

The critical portion of the blast furnace analysis program is the section which reads one furnace at a time, calculates the life of the furnace, checks the spacing of the census years available for that furnace and determines the built and rebuilt dates for the furnace. To test the program's performance in each of these areas, several sets of test furnace data are needed. The program must sequence through the file properly, selecting one furnace at a time. The furnace life should be calculated correctly for furnaces with one or more data cards. Furnace data with missing or incorrect census dates should be identified properly and furnaces with census dates out of range should be rejected by this program.

When calculating built or rebuilt dates, the date which occurs immediately prior to the first census date or the first census date itself, if no earlier dates are recorded, should be reported as the earliest date that the furnace is known to be of the height and width which is reported in the first census. All previous dates should be reported with unknown heights and widths. Any subsequent change in height or width of more than two feet should be noted and either that census year or a reported rebuilt date prior to that census year, but following the previous census year, should be reported as the rebuilt date.

The one phenomenon which was not expected when the original test data was derived, and which was discovered by using some of the actual data, was the possibility that a furnace could change its height or width in the last census year. This means that the furnace is known to exist at the new height and width for only one year. A check for this phenomenon was included in the program and it was found to occur a number of times. Once the test data has been used to determine that

the above checks by the program are being made correctly, testing the remainder of the program involves a hand check on some small data sets to verify that the desired averages are being calculated correctly. Two errors found here were a misnaming of one variable, which caused one average to be double its correct value, and a variable which was declared with too small precision. In the latter case, the accumulated sum overflowed the capacity of the variable causing the average to be much too small. This was easily rectified by increasing the precision of the variable.

Finally, a useful method of checking the flow of control and the intermediate results in a program is to place statements which cause a literal string or selected variables to be printed from strategic points in the program. These statements might be placed immediately after a variable has been changed or just before a decision point in the program. These print statements are controlled by IF-THEN statements, each containing a boolean variable whose value is set by the programmer. In this way the programmer can control a trace of selected variables in various sections of his program by changing the values of the boolean variables. Sections of the program which are executing satisfactorily will have the boolean variables "turned off" so that program execution time and paper will not be wasted. Should an unexpected situation arise, the boolean variable can easily be "turned on" and the program trace will be available without reinserting the print statement.

CONCLUSION

The main emphasis of this paper has been the preparation and checking of programs and their use in checking and analyzing data of an historical nature. We tried to report the types of analysis performed on the data both to locate errors in the data and to return meaningful results to aid in the study of the changing technology of charcoal iron blast furnaces. We included a detailed discussion of the types of errors in the data and of misconceptions about the data that we found in order that the reader will benefit in planning his own project.

This particular program was written in the programming language PL/1 [3]. The advantages of this language include the control the programmer has over file access and manipulation, the freedom one has in selecting variable names which identify their use in the program, the control structures that enable the programmer to write code which closely follows the principles of structured programming, and the data types, such as structures and arrays, which allow the programmer to describe the data in a manner closely related to his own mental conception of the data. The language COBOL (Common Business Oriented Language) [8] contains features similar to the needed features of PL/1 except for the lack of the control structures approximating those of structures programming. The features of the languages BASIC [11] and FORTRAN IV [4] are similar to each other in their lack of suitability for this type of data processing. They are both oriented toward scientific problem solving and, although all of the operations needed for this application are available in these languages, the clarity of the code and ease of programming would suffer.

The easy to use file handling capabilites and univariate analysis procedures of the Statistical Analysis System (SAS) [2] make it an attractive alternative to the rigorous programming knowledge requirements of PL/1 or COBOL; however, it has the definite disadvantage of having neither arrays nor structures. Thus, every field must be named and referred to individually. Moreover, the data cards for one furnace could not have been read into main storage and processed as a group; the processing must be planned to occur on one record at a time. It is possible, although it was not tried by the authors, that the sorting and editing programs could have been written more easily in SAS, especially since SAS has a built-in procedure for sorting files. We do feel that for simple data handling, combined with quite sophisticated statistical data analysis, the SAS system may be a good choice for inexperienced programmers.

No matter what language or system is chosen to be used to analyze the data, the main problems will occur in handling and checking the data. To minimize this problem, we suggest that the type of data needed, the data gathering procedure and the subsequent analysis be carefully planned before any programming is attempted.

REFERENCES

1. American Iron and Steel Association, Directory of Iron and Steel Works of the United States and Canada, 1874, 1876 - present. (Referenced in the paper as the Directory).

2. Barr, Anthony J. and James H. Goodnight, A User's Guide to the Statistical Analysis System, Sparks Press, P.O. Box 26747, Raleigh, N.C. 27611, 1972.

3. Bates, Frank and Mary L. Douglas, Programming Language One, third edition, Prentice-Hall, Englewood Cliffs, N.J., 1975.

4. Blatt, John M., Introduction to FORTRAN IV Programming, Goodyear Publishing Company, Pacific Palisades, California, 1971.

5. Dahl, O.J., E. W. Dijkstra and C. A. R. Hoare, Structured Programming, A. P. I. C. Studies in Data Processing No. 8, Academic Press, New York, 1972.

6. Date, C. J., An Introduction to Database Systems, Addison-Wesley Publishing Company, Reading, Massachusetts, 1975.

7. Derry, T. K. and Trevor I. Williams, A Short History of Technology, Oxford University Press, New York, 1960.

8. Feingold, Carl, Fundamentals of COBOL Programming, second edition, Wm. C. Brown Company Publishers, Dubuque, Iowa, 1973.

9. Knuth, Donald E., The Art of Computer Programming, Volume III, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

10. Lesley, John Peter, The Iron Manufacturer's Guide to the Furnaces, Forges, and Rolling Mills of the United States, John Wiley, New York, 1859.

11. Marateck, Samuel L., BASIC, Academic Press, New York, 1975.

12. Schallenberg, Richard H., Evolution, Adaptation and Survival: The Very Slow Death of the American Charcoal Iron Industry, Annals of Science, 32 (1975), 341-358.

13. Schallenberg, Richard H. and David A. Ault, The American Charcoal Iron Industry in the Coal Smelting Era: Regional Patterns of Technological Innovation, submitted for publication, October 1975.