Lehigh University
Lehigh Preserve

Theses and Dissertations

1984

# Computer aided caring (CAC) :

Kevin Eric Knauss
*Lehigh University*

Follow this and additional works at: https://preserve.lehigh.edu/etd

 Part of the Industrial Engineering Commons

Recommended Citation
Knauss, Kevin Eric, "Computer aided caring (CAC) :" (1984). *Theses and Dissertations*. 4467.
https://preserve.lehigh.edu/etd/4467

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

# COMPUTER AIDED CARING (CAC):
# THE USE OF COMPUTER EXPERT SYSTEMS
# IN THE LOCAL CHURCH

by

Kevin Eric Knauss

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

1984

## CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

December 14 1961
(date)

Professor in Charge

Chairman of Department

## ACKNOWLEDGEMENTS

I would like to thank the council of the Church of the Manger, Bethlehem, PA for allowing me to use the contribution records of the church, without which this project would have been nothing more than a "wish book." Special thanks are due to Wilma Reinbold, financial secretary of the Church of the Manger, who put in the time of preparing the records to insure the privacy of the members of the church. Without the help of the Rev. Dr. Earl R. Shay of Moravian Theological Seminary who served as my associate advisor, I would still be scrambling around in an attempt to collect the needed resources for this project. My thesis advisor, Dr. John Wiginton of Lehigh University, insisted that I should create a working prototype; while I would have been happy to do without the extra work involved, I would not have been happy to be without the knowledge that this approach forced me to gain ... thank you! Lou-Anne Rogers, my proofreader/editor, insured that I "did good writin'." Lastly, I would like to thank my trooper of a wife, Beth, who put up with all the midnight oil that I burned and the daylight hours that I slept away.

Computer Aided Caring (CAC):
The Use of Computer Expert Systems
in the Local Church

CONTENTS

## ABSTRACT

The advent of the personal computer has allowed local churches to purchase computer systems. The computer market within churches was previously restricted to a few large churches and denominational headquarters. Even though churches can benefit from the traditional uses of computers, such as word processing and accounting, the Church is a special kind of business with a special kind of product: a cared-for person. This thesis is an investigation into the feasibility of using contribution records as the basis of an expert system to be run on micro-computers in local churches to aid the ministry team in the identification of families and individuals who may be in need of pastoral care.

Computer expert systems, which are designed to provide guidance in making decisions, are normally too complex to be run on micro-computers. This is especially true of the type which is based on production rules for making decisions. The focus of an expert system based on Church offering contributions involves making relatively few decisions based upon a great deal of processing. For such purposes, a procedural system capable of running on a micro-computer is both desirable and possible. It is

important in the development of such a system to maintain a high level of modularity so that knowledge in the form of rules of inference may be added and deleted as needed.

A procedural expert system was designed and implemented in the Pascal programming language in a test environment. Actual data was gathered from a local church and run through three rules of inference: 1) generating a trend of giving and predicting the current offering based upon the trend, then comparing the predicted value with the actual offering using confidence intervals; 2) accepting low discrepancies but not high discrepancies for holidays; and 3) checking to see if regular givers have not given for an "unreasonable" length of time. The prototype system generates reasonable output, but some revisions are necessary and others may be desirable for production use.

# 1. THE PROBLEM

1.1 Introduction Recent trends in the development of computer hardware and plummeting prices of this machinery have led to the use of computers by the Church. Prior to this technological revolution, there were some computers being used within this institution; but they were found only in denominational offices and in very large churches. Today these useful tools cost about as much as the annual salary of a part-time secretary, and the trend toward more powerful machines with smaller price tags is expected to continue. Because of this phenomenon, pastors, church treasurers, and other local church officials have begun to clamor for this new tool which will apparently make everyone's job much easier. Increasing demand has been met by numerous entrepreneurs who believe that the Church is like any other business, one whose needs which can be solved with traditional remedies. There is some validity to this argument. Many of the software packages which have been produced can indeed be valuable in this setting, but the Church has special needs. These needs must be addressed if this new tool is to be used to its fullest.

1.2 Presuppositions The Church is a setting in which people are cared for in a manner similar to that in a hospital. "In illness we see that we are whole persons

rather than beings who can neatly be diagnosed as having physical or emotional problems."[1] Doctors in hospitals are becoming more concerned with this wholistic approach to the treatment of their patients. Ministers (not limited to the ordained variety) are similarly concerned with this comprehensive view of an individual. However, often peoplewith no apparent need for healing attend a church, while those in a hospital setting more often require physical as well as emotional care. Thus, "Loss and grief traditionally have been the domain of the pastor."[2] and "Pastors are at the forefront in dealing with human disturbances at [the] family level."[3] The similarity between these two professions is growing as ministers learn diagnostic techniques formerly confined to the medical community. Both fields are moving toward the greater use of specialists, referring individuals to consultants in their respective disciplines. Though the medical profession is attempting to move beyond its classically reactive posture by introducing preventive medicines and health maintenance organizations, the Church will undoubtedly remain on the leading edge of active healing. "The growth approach to

1. William V. Arnold, Introduction to Pastoral Care (Philadelphia: Westminster Press, 1982), p. 173.

2. Ibid., p. 161.

3. David K. Switzer, The Minister as Crisis Counselor (Nashville: Abingdon Press, 1974), pp. 181-182.

ministry is one key to helping people seek counseling long before they're on the brink."[4]

1.3 Identification With the Church being in the forefront of active healing, one might assume that a list of people waiting to be treated is readily available. This assumption, of course, would be erroneous. Although some problems are brought to the attention of a pastor, most pastors spend many hours reading through newspapers, checking hospital rosters, and in other pursuits trying to identify potential needs within the congregation. Even with all this effort, many problems remain unnoticed until they grow into insurmountable obstacles. What then is the nature of problems of which a pastor should be aware in order to address them before escalation takes place?

"All of us experience stress regularly in a variety of circumstances. In fact, if we did not experience it, we would be dead!"[5] This statement may seem irrelevant, but studies have shown that stress sustained at moderately high levels leads in many cases to more serious ailments. Thomas H. Holmes M.D., professor of psychiatry at the University of Washington, has developed an impact scale

4. Howard J. Clinebell, Jr., Growth Counseling for Marriage Enrichment (Nashville: Abingdon Press, 1966), p. 71.

5. Arnold, p. 153.

for stress events which assigns point values to common life events. By tallying all the point values that apply to an individual and comparing the result with various ranges of scores, one may make predictions of the probability of that individual developing more serious ailments. Among the life events that are scaled are many positive factors of peoples lives.[6] This could indicate why so many problems are allowed to escalate, because they may not be seen as problems at all until they have evolved. Although it is impossible and unnecessary to identify all life events which cause stress, it is desirable to identify those events which cause changes in behavior.

1.4 Prospectus The computer discipline known as artificial intelligence has been expanded to include the subordinate speciality known as expert systems. It is through these systems that the computer has become a valuable tool within the medical profession for the diagnosis of varying illnesses and the suggestion of possible remedies. Though ministers and counselors within the Church might benefit from these specific solutions to problems, their first concern is with finding the people who may be in need of healing; for only after a person has been found to have a

6. "Rating Life Changes," The Christian Ministry, II, no. 4 (July 1971), 14.

problem can that problem be diagnosed. The time seems ripe for the introduction of an expert system derivative which will run on a micro-computer and assist the local church ministry in identifying parishioners who may be in need of healing. Such a system will take advantage of the latest advances in technology but will also take into consideration the limited resources of most local churches. After this type of system has been developed, there may be a call for systems which will assist in the narrowing of possible problems. The first step, however, is to identify the person in need; for "The process rather than the product is primary in caring, for it is only in the present that I can attend to the other."[7]

By analyzing financial data, trend analyses can be generated which may point to unusual circumstances in a family's life. Deviations from giving trends may indicate times of stress and could be used to generate pastoral contact in a situation which may have otherwise been overlooked. A drop in the normal amount given might indicate a negative factor such as an illness in the family, loss of employment by one of the family members, or any of a number of equally stressful events. Increases in giving, though always welcome by the Church, may also indicate a

7. Milton Mayeroff, On Caring (New York: Harper & Row, 1971), p. 31.

time of family stress. A steady increase in giving might indicate the attainment of a new job while a windfall gain might be the result of the death of a family member or friend. A radical change in usual behavior is recognized as an indicator of distress.[8] If a family has been giving regularly and later stops or vice versa, this too may indicate a time of stress within the family. These indicators cannot always be relied upon to show a need for counseling nor can they be depended upon entirely; but if one or two families with problems that may have otherwise been overlooked are identified each year, the benefits can outweigh the costs.

8. Howard J. Clinebell, Jr., Basic Types of Pastoral Counseling (Philadelphia: Fortress Press, 1975), p. 82.

## 2. DISCUSSION

**2.1 Expert Systems** "An expert system is a program or set of programs designed to provide the user guidance in making decisions at least as well as human experts might."[1] This definition may seem simplistic, but it contains the essence of what an expert system should entail. It is true that there are many complex systems which deserve more complete and detailed definitions, but there is no reason that all expert systems must be complex. It will be the focus of this chapter to discuss the various types of expert systems, the attributes of the different types, and the characteristics which are common to the more useful systems regardless of their type or complexity.

There are many occupations which require a deep understanding of certain information in order to carry out what might seem to be a mindless or trivial task. A physician, for example, learns about thousands of illnesses in order to be able to diagnose the common cold with confidence. A systems programmer spends years learning about programming and the internal mechanisms of operating systems in order to be able to choose between statement A and statement B in a system generation process. In this type

1. Jon Roland, "AI, UNIX and C," UNIX/WORLD I, no. 3 (1984), 100.

of situation, an expert system can be developed to free these highly trained individuals for more productive tasks. These systems might also be developed to perform tasks that might otherwise be left undone because of the tremendous amount of work required for only marginal benefits. If tremendous amounts of data have to be processed in order to determine a mere possibility, a manual system will probably not be instituted because the cost would far outweigh the benefit. A computer system, on the other hand, may be inexpensive enough to provide reasonably for the benefit.

> Researchers have discovered that although "deep" knowledge about biology and medicine might be involved in developing tests and treatments, the process of selecting them is usually reduced in practice to a set of simple associative rules of inference that assign confidence values to various alternatives on the basis of information acquired, then suggest courses of action intended to distinguish the alternatives until the confidence value of one is high enough to justify settling on it. These rules are often derived from subjective experience or from simple statistical correlations among observed data.[2]

Computer expert systems can be used to free individuals for more productive work or to create a benefit where one did not previously exist.

Expert systems generally fall into two distinct types based upon the goal of the project and into two distinct

2. Ibid., p. 100.

types based upon the method of solution. The types based upon the goal of the project are "characterized on the one hand by the psychological modeling efforts, ... and on the other by the performance oriented, knowledge-based expert systems."[3] This project deals exclusively with knowledge-based systems. The types based upon the method of solution are characterized by production systems which rely upon the data base in order to create a direction of flow through the system and by procedural systems which contain the direction of flow as part of the structure of the procedures themselves. Both production systems and procedural systems will be discussed at length in this chapter.

"Knowledge acquisition has become recognized as an issue with expert systems because it has turned out to be difficult and time consuming."[4] As a result, there has been a movement towards developing design tools which can help in the design, construction, and/or modification of an expert system. These design tools may be as complex as

3. Randal Davis and Jonathan King, "An Overview of Production Systems," Machine Intelligence, VIII, E. W. Elcock and Donald Michie, eds. (New York: John Wiley and Sons, 1982), p. 305.

4. B. G. Buchanan, "New Research on Expert Systems," Machine Intelligence, X, J. E. Hayes, Donald Michie, and Y-H PAO, eds. (New York: John Wiley and Sons, 1982), pp. 278-279.

a design system (such as EMYCIN, OPS4, XPRT, etc.) which is an expert system without the domain-specific knowledge and "assumes that production rules are an appropriate representation framework for a person's new knowledge base and that a backward-chaining, or goal-directed, interpreter is an appropriate inference mechanism."[5] Conversely, the design tools may be as simple as a list of questions which will help in the identification of the primary object(s) about which the expert system should offer advice, its various subdivisions, its attributes, and its possible values.

Besides the design tools, various methods or techniques may be applied to the design and construction to aid in later modification. Foremost among these is the use of modularity.

> We can regard the modularity of a program as the degree of separation of its functional units into isolated pieces. A program is highly modular if any functional unit can be changed (added, deleted, or replaced) with no unanticipated change to the other functional units.[6]

Though procedural systems may be written using this technique, production systems are by nature extremely modular. As new knowledge is gained through research, through advances within a field, or simply through having enough

5. Ibid., p. 282.

6. Davis, p. 316.

time to implement a familiar solution to yet another problem, it will be desirable to change the functional units of an expert system. High modularity will allow these changes to be made quickly, easily, and with few if any ill effects.

2.2 Production Systems When attempting to determine the approach to use in solving a problem, an individual must not only be aware of the tools available for use, but also the proper use of the tools. A screwdriver with a weighty handle, for instance, might be adequate for driving nails; but as the action involved in driving a nail is not circular, a hammer would be much more efficient for this purpose. Expert systems are tools available to be used in solving certain types of problems; and like their hand tool counterparts, the assorted types of expert system methods vary in appropriateness with respect to the differing actions to be undertaken. While procedural systems might be suitable in the manipulation of large amounts of data, production systems are better suited to making many varied decisions because they "emphasize the statement of independent chunks of knowledge from a domain and make a control flow a secondary issue."[7]

7. Ibid., p. 312.

Knowledge is required in making decisions. If one is to make many varied decisions, one should have knowledge spanning the entire variety. Much of our knowledge is unstable, however, since we progress in our thinking and learn new possibilities that can be used in our decision making. These new possibilities should be remembered so that they may be included in future decisions; but when dealing with computer expert systems, what we learn must be codified as a representation of knowledge. Fortunately, "production rules offer a representation of knowledge that is relatively easily accessed and modified"[8] which makes production systems very well suited as expert systems with large and/or volatile knowledge bases. Decisions are made based upon the knowledge represented in the data base; this accommodates a greater degree of modularity than could be attained from controlling the decisions from within a procedure.

> The modularity of programs written as pure production systems arises from the important fact that the next rule to be invoked is determined solely by the contents of the data base, and no rule is ever called directly. Thus the addition (or deletion) of a rule does not require the modification of any other rule to provide for (delete) a call to it. We might demonstrate this by repeatedly removing rules from a [production system]: many systems will continue to display some sort of "reasonable" behavior, up to a point. By contrast, adding a procedure to an ALGOL-like program requires modification of other parts of

8. Ibid., p. 306.

> the code to insure that it is invoked, while removing an arbitrary procedure from a program will generally cripple it.[9]

Two expert systems based upon production rules and worth examining are DENDRAL and MYCIN. Both of these systems deal with a medical knowledge base; but while DENDRAL evolved into a production system from a procedural infancy, MYCIN, "with the three criteria of utility, performance, and transparency among its design goals,"[10] was built from the ground up as a production system. The knowledge in the DENDRAL system "is extended by adding rules that apply to new classes of chemical compounds. Similarly, much of the work on the MYCIN system has involved crystalizing informal knowledge of clinical medicine in a set of production rules."[11] A contrast of these two systems should show some of the advantages and uses of production systems.

The knowledge of chemistry in the DENDRAL system was originally "custom-crafted" over many years using a procedural approach. The data provided by chemists was moulded into knowledge by the system's developers and fixed into place within the procedures, but problems with

9. Ibid., p. 316.

10. Buchanan, p. 270.

11. Davis, p. 306.

flexibility rose from this approach as one of the developers explains:

> We rewrote large parts of the systems as the knowledge base changed. After doing this a few times we began looking for ways to increase the rate of transfer of chemistry expertise from chemists into the program. Making procedures highly stylized and dependent on global parameters was a first step, but still required programmers to write new procedures. DENDRAL's knowledge of mass spectrometry was finally codified in production rules.[12]

This evolution seems to indicate that if a system's knowledge base is volatile, production rules are a better choice than a procedural approach for implementing it. By rewriting DENDRAL as a production system its modularity was increased tremendously, rendering changes in its knowledge base as mere modifications of the rule set rather than major rewrites of the system.

Unlike DENDRAL, MYCIN did not evolve but was originally designed to be a production system. This approach was taken for MYCIN in part because its authors knew of the experience of the DENDRAL system as it had been started nearly ten years before MYCIN was conceived and because its authors anticipated and desired that its knowledge base would undergo changes. Thus,

> much of the MYCIN system's capability for explaining its actions is based on the representation of knowledge as individual production rules. This

12. Buchanan, p. 278-279.

> makes the knowledge far more accessible to the program itself than it might otherwise be if it were embodied in the form of ALGOL-like procedures. As in DENDRAL, the modification and upgrading of the system are by incremental modification of, or addition to the rule set.[13]

2.3 Procedural Systems Though a screwdriver may not be appropriate for driving nails, it is far more appropriate than a hammer for driving screws. In the same manner, procedural systems are better suited than production systems when relatively few decisions are to be made based upon the manipulation of large amounts of data. "It is not necessary that expert systems are psychological models of the reasoning of experts. Statistical pattern recognition programs, for example, perform well on many important problems."[14] Procedural systems may therefore be preferred because of the type of problem that is to be solved, and they are best suited when the knowledge upon which the decisions are to be based is inherent in a process and not in previous experiences.

Occasionally, the choice of an expert system may be made because of limitations rather than benefits.

> So much work in [artificial intelligence] has run up against hardware limitations that many think that interesting [artificial intelligence] programs can only be run on large supercomputers. Certainly, little of great interest can be done on

13. Davis, p. 306.

14. Buchanan, p. 270.

> 8- or 16-bit systems, but the 32-bit supermicros, with their high speeds and large address spaces ... offer sufficient power for some useable expert systems.[15]

Despite the optimism in this statement over the new supermicros, there is a message of caution regarding the most affordable computers which should be heeded. Also, "The state of induction programs is not up to widespread use for constructing knowledge bases."[16] With these limitations of both hardware and software, it is unlikely that we will find production systems being implemented on micro-computers. If these are the only computers that a particular organization can afford at present and an expert system is desired, it would seem that a procedural system would be in order until such time that more sophisticated resources are available.

Regardless of the reason for implementing a procedural expert system, maintaining relatively-high modularity within the programs of the system should be a priority. Whereas this is not a problem with production systems because of their nature, modularity is a major concern with procedural systems especially when they are implemented because of limitations. The inference mechanisms or knowledge are bound to change with time in any

15. Roland, p. 98.

16. Buchanan, p. 279.

expert system; and if proper care is not taken to insure the ease with which it can be modified, the cost of the modification could outweigh the benefits to be obtained. This lack of foresight could result in the compromise of the entire system. "We see relatively few expert systems, and those we do see include considerable domain-specific knowledge codified over months or years."[17] When the development of a program is spread out over a period this long, modularity is necessary not only for the ease of modification but also for the integrity.

"For those wishing to build knowledge-based expert systems, ... knowledge ... can be executed by one part of the system as procedural code, and examined by another part as if it were a declarative expression."[18] This type of hybrid system can be very beneficial. One benefit of a hybrid system is that limitations may be only partial, and it would be desirable to do as much in the appropriate method as possible. Another benefit is that some decisions might be better made with a procedural method while other decisions might be better made with production rules; each could be made in its appropriate domain. One final benefit is that once a method has been chosen,

17. Ibid., pp. 269-270.

18. Davis, p. 328.

another method can be used for future developments if that method is better suited for them.

## 3. IMPLEMENTATION

3.1 Approach When designing an expert system which will help the ministry staff of a local church to identify potential families in need of caring, it is necessary to consider two major factors: 1) the resources available and 2) the type of processing to be done. As most churches do not have large budgets available for the purchase of computers and software, it is prudent to assume that an 8- or 16-bit micro-computer is the most sophisticated computer which might be found in the majority of churches in the near future. This monetary limit will also force most of the software to be in the form of inexpensive packages, general purpose compilers, and public domain programs. The type of processing as mentioned in the first chapter will involve few decisions based upon the processing of large amounts of data. Trends will be sought in the giving patterns of parishioners, and discrepancies from those trends will raise signal flags which will require a manual investigation. All of these characteristics indicate that a procedural type of expert system should be implemented.

With this in mind, a procedural system was designed and developed which will assist in the identification of persons within a local church setting who may be in need of special attention. This system is a working prototype

limited in scope but designed to be modifiable into a production model. Live data has been run through the system producing reasonable results; but because of the need to protect the privacy of individuals, the actual proving ground must remain with the ministry teams of the churches who implement such a system. The system can be maintained with a minimum amount of data entry, and this coupled with its modular design should prove it to be a practical archetype.

Three rules of inference have been implemented in this project. The first and second rules deal with abnormal fluctuations in giving amounts, while the third rule deals with irregularly long periods where a family gives nothing. These fluctuations and irregularities might indicate times of stress in a family's life. The first rule involves comparing the current offering amount with a trend of giving. As many churches ask their parishioners to pledge a regular contribution, a trend analysis should be fairly reliable. Though a family may miss a week or two of contributions, eventually most families make it a point to meet their pledge; a trend analysis will not be affected by these normal fluctuations. The second rule involves times of the year when abnormal giving habits can be expected. Obviously, if a change can be predicted,

that change will hardly be abnormal; but the change will seem to be abnormal in trend analysis. Times such as Christmas and Easter normally bring larger contributions than regular weekly contributions. Thus positive discrepancies from trends should be ignored or the confidence levels should be adjusted upward on these dates. The third rule involves families who have been giving on a regular basis and then stop giving. Since one family's regularity will differ from another's, the frequency of of each family's giving will be determined and multiplied by a common reasonableness factor in order to make a determination.

3.2 Programming The Pascal programming language was chosen for this project because of the availability of compilers and interpreters on so many different computer designs. Although there is not an official standard Pascal sanctioned be the American National Standards Institute (ANSI), there seems to be a "bare bones" standard Pascal which is based upon Niklaus Wirth's original design specifications for the language. This relatively high degree of standardization made Pascal a better choice than the more popular micro-computer programming language Basic, for Basic has no standard form and is often as device dependent as assembler languages. In order to main-

tain a high level of standardization, twenty-eight subroutines in seven different libraries were written to perform primitive functions. Some of these primitive functions can be found in various forms in many implementations of Pascal but not in all; therefore, by writing them in the most basic form of Pascal which is also the most standard, the differences in implementations will be minimized if not eliminated. Several of these routines are designed to look and work like some of the built-in functions in the C and PL/I programming languages while the input routines are loosely based on the primitives developed for the University of California at Berkeley by Bill Joy and Charles Haley.[1] Source code listings for all programs and subroutines developed for this system have been placed on file in the Industrial Engineering department of Lehigh University.

Documentation for the system (programs and files) can be found in an appendix to this document. Of the six programs documented, four (mkmast, valid, weekly, and wkmerg) deal with the creation and maintenance of the data base; and one program (clndr) is used to supply useful information in the manual use of the system. The remaining pro-

1. Brian W. Kernighan and P. J. Plauger, Software Tools in Pascal (Reading, Massachusetts: Addison-Wesley, 1981), pp. 321-331.

gram (analyze) contains the inference engine (knowledge) of the system; if a suitable data base is already available, only this program need be used. It should be pointed out, however, that although the appropriate data might be available, its form or order may render it unusable without major revision. If it is determined that existing data is useable in its current form and order, the input portion of this program (analyze) must be changed to fit the different data base. The remaining programs may be discarded. Knowledge in the inference engine is modular and may easily be added to or deleted from using procedures, but it is important that calls to these procedures be appropriately added or deleted since the implementation is in a procedural language. If data exists in computer-readable form but it has to be rearranged for use with this system, three of the programs (mkmast, weekly, and wkmerg) can be modified and used to reformat the data while the remaining programs are used as is.

3.3 Knowledge The knowledge of this system is contained in the three rules of inference which are described above and are defined as follows. If the offering amount for the process date is non-zero, the first rule of inference will be invoked and will begin by analyzing, using linear

regression, the sums of the offering amounts up to but not including the process date. With this regression line, it will extrapolate the value which might be expected for the process date. Using confidence intervals, it will compare this value with the actual value. If the actual value falls outside the range of the smallest confidence level, then a message from the confidence file associated with the largest confidence level range that the value falls outside will be printed. Printed along with the message will be the fact that the actual value is above or below the indicated range. If the offering amount for the process date is non-zero and the process date is considered to be a holiday, the second rule will be invoked and positive discrepancies of the first rule will be ignored. An alternative to this rule is the use of different confidence values with higher ranges, but this might cause some meaningful negative discrepancies to be overlooked. A combination of this rule for negative discrepancies and the use of different confidence values in a separate run of the program for positive discrepancies is recommended. If the offering amount for the process date is zero, the third rule will be invoked to analyze the frequency of giving and to determine if the time elapsed since the last offering is unreasonably long. If so, a message indicating that giving habits seem to have changed will be

printed. In order to keep these statements to a reasonable number (i.e., to not print messages for long-inactive members or for families who no longer hold membership), a maximum as well as a minimum is placed upon the unreasonable range.

$$\Sigma y_i = na + b\Sigma x_i$$

and

$$\Sigma x_i y_i = a\Sigma x_i + b\Sigma x_i^2$$

represent the "normal equations" which, when solved simultaneously, will yield the least squares equation

$y_c = a + bx$ see footnote 2

In solving for b, the equations may be reduced to

$$b = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2} ;$$

and a may be solved in terms of b as

$$a = \frac{\Sigma y_i - b\Sigma x_i}{n} .$$

The following definitions have been implemented in order to program this solution:

2. Wayne W. Daniel and James C. Terrell, Business Statistics, Basic Concepts and Methodology (Boston: Houghton Mifflin Company, 1975), p. 235.

$off_i$ = offering amount given on date i

n = number of non-zero $off_i$ in sample

$x_i$ = i = date (Julian) for non-zero $off_i$

$$y_i = \sum_{j=1}^{i} off_j$$

$$sigx = \sum_{i=1}^{n} x_i$$

$$sigy = \sum_{i=1}^{n} y_i$$

$$sigxy = \sum_{i=1}^{n} x_i y_i$$

$$sigxsqr = \sum_{i=1}^{n} x_i^2 .$$

With a and b known for a given set of data, a point prediction of the value $y_c$ can be obtained for a particular value of x ($x_p$) by substituting $x_p$ into the sample regression equation and solving it. A prediction interval with confidence of 100(1 - $\alpha$) percent may be obtained for $x_p$ by solving the following:

$$y_c \pm t_{(1-\frac{\alpha}{2})} \; s_{y|x} \sqrt{1 + \frac{1}{n} + \frac{(x_p - \bar{x})^2}{\Sigma (x_i - \bar{x})^2}}$$

with degrees of freedom used in selecting t of n-2.[see

footnote [3] In order to solve this equation with a computer program, it is desirable to reduce $s_{y|x}$, $\bar{x}$, and $\Sigma(x_i - \bar{x})^2$ in terms of the variables defined above.

$s_{y|x}$ can be expressed as

$$s_{y|x} = \sqrt{\frac{\Sigma(y_i - y_c)^2}{n-2}},$$

but it is obvious that the terms have not yet been reduced sufficiently.[4] Considering the equation

$$\Sigma(y_i - \bar{y})^2 = \Sigma(y_c - \bar{y})^2 + \Sigma(y_i - y_c)^2 \text{ ,see footnote 5}$$

the undefined portion of the previous equation can be isolated as

$$\Sigma(y_i - y_c)^2 = \Sigma(y_i - \bar{y})^2 - \Sigma(y_c - \bar{y})^2$$

The parts of this equation can be defined as

$$\Sigma(y_i - \bar{y})^2 = \Sigma y_i^2 - \frac{(\Sigma y_i)^2}{n}$$

and

$$\Sigma(y_c - \bar{y})^2 = b^2(\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}) \text{ see footnote 6}$$

3. Ibid., p. 248.

4. Ibid., p. 245.

5. Ibid., p. 240.

which when substituted in the previous equation gives

$$\Sigma(y_i-y_c)^2 = \Sigma y_i^2 - \frac{(\Sigma y_i)^2}{n} - b^2(\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}) .$$

This reduces $s_{y|x}$ in terms of the implemented definitions with only the addition of

$$\text{sigysqr} = \sum_{i=1}^{n} y_i^2$$

needed.

Of the remaining unknowns, $\bar{x}$ can be reduced as

$$\bar{x} = \frac{\Sigma x_i}{n}$$ see footnote 7

and $\Sigma(x_i-\bar{x})^2$ can be reduced in a manner similar to y above as

$$\Sigma(x_i-\bar{x})^2 = \Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n} .$$

With all of the unknowns reduced in terms of implemented definitions, the radicals in the prediction interval expression may be evaluated as

6. Ibid., p. 240.

7. Ibid., p. 16.

$$\sqrt{\frac{\Sigma y_i^2 - \frac{(\Sigma y_i)^2}{n} - b^2\left(\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}\right)}{n-2}} \sqrt{1 + \frac{1}{n} + \frac{(x_p - \frac{\Sigma x_i}{n})^2}{\Sigma x_i^2 - \frac{(\Sigma x_i)^2}{n}}}$$

and saved in a variable (defined here as confindx) so that it will not have to be reevaluated for use in several prediction intervals of the same group of data, since only the t value will change. Thus, the prediction intervals for y given x may be calculated from the equations:

$$\text{upper bound} = y_c + t_{(1-\frac{\alpha}{2})} \text{ confindx}$$

and

$$\text{lower bound} = y_c - t_{(1-\frac{\alpha}{2})} \text{ confindx.}$$

The rate of giving for the third rule is calculated as

$$\frac{x_1}{n}$$

where

$x_1$ = i = date (Julian) for the last non-zero $off_i$

and n is the number of non-zero contributions as defined above. Since $x_1$ is a number of days, this rate will show an average of how many days have elapsed between each two

contributions. The check for reasonableness is achieved by multiplying the rate by a constant (which is set by the church staff) and representing a factor of reasonable irregularity. The result is then compared with the time elapsed since the last contribution ($x_p - x_1$). Reasonableness is maintained as long as the calculated figure remains greater than the elapsed time figure.

3.4 Future Considerations In order to implement this system in a production environment, some modifications should be made. The system has been programmed for only the current year's data. This will be a problem for early months of the year as there will be an insufficient base of data for meaningful trends to be established. The modification of the programs to accommodate multiple years will be relatively trivial, involving mostly the dimensions of arrays and the bounds of iterative loops. The most formidable task to overcome involves the coordination of families' contributions from one year to the next as in many churches different envelope numbers are assigned to families each year. This is not because the church enjoys playing musical envelopes. Rather, as new members join the church and others leave, the order is disrupted. Since the system is designed to work with envelope numbers rather than with names, a system of transposing one year's

numbering system into the next will be necessary.

Besides the modifications necessary for this system, there may be some unneeded enhancements which would simply be desired. In the problem statement, for instance, it was mentioned that there might be two types of changes in giving trends: 1) a single contribution which does not fit the trend or 2) a gradual metamorphosis from one trend into another. The first discrepancy is considered by the currently implemented rules of inference; but while a radical change in trends might be detected, it is very likely that a gradual change would go unnoticed. Thus, it may be desirable to add a new rule of inference to the system that would compare on a floating basis any recent trend with any prior trend. There may of course be any of a number of such changes desired which can be implemented at any time.

Extensive feedback and tuning will also be necessary not only for an initial implementation but also for an ongoing basis. In the initial study, less than one half of one percent of the envelopes which had weekly amounts within the 98% confidence interval seemed to reflect data of interest when checked manually. Roughly seventeen percent of the envelopes which had amounts outside of this range were chosen to be evaluated through the church

staff. Sixty-six percent of this data was filtered out by the financial secretary because of clerical errors detected. The remaining data reflected only families who increased their giving. Of the envelopes indicated by the system to have changed giving habits, fifty percent were chosen to be evaluated through the church staff. Seventeen percent of these were due to families transferring out of town, and another seventeen percent of the families made large contributions later to meet their pledges. Thirty-three percent of the families made subsequent contributions but did not realize their pledges. A total of seventeen percent of the families evaluated by the church staff were found to have legitimate need of pastoral counselling. An infant must learn to walk before it learns to run; this project represents the first few steps of the infant: "Computer Aided Caring: The Use of Computer Expert Systems in the Local Church."

## BIBLIOGRAPHY


Arnold, William V. Introduction to Pastoral Care. Philadelphia: Westminster Press, 1982.

Clinebell, Howard J., Jr. Basic Types of Pastoral Counseling. Nashville: Abingdon Press, 1966.

----------. Growth Counseling for Marriage Enrichment. Philadelphia: Fortress Press, 1975.

Buchanan, B. G. "New Research on Expert Systems," Machine Intelligence, X, J. E. Hayes, Donald Michie, and Y-H PAO, eds. New York: John Wiley and Sons, 1982, pp. 269-299.

Daniel, Wayne W. and James C. Terrell. Business Statistics, Basic Concepts and Methodology. Boston: Houghton Mifflin Company, 1975.

Davis, Randal and Jonathan King. "An Overview of Production Systems," Machine Intelligence, VIII, E. W. Elcock and Donald Michie, eds. New York: John Wiley and Sons, 1982, pp. 300-332.

Dudewicz, Edward J. Introduction to Statistics and Probability. New York: Holt, Rinhart and Winston, 1976.

Hoel, Paul G., Sidney C. Port, and Charles J Stone. Introduction to Statistical Theory. Boston: Houghton Mifflin Company, 1971.

Kernighan, Brian W. and P. J. Plauger. Software Tools in Pascal. Reading, Massachusetts: Addison-Wesley, 1981.

Mayeroff, Milton. On Caring. New York: Harper & Row, 1971.

Newton, Byron L. Statistics for Business. Chicago: Science Research Associates, 1973.

"Rating Life Changes," The Christian Ministry, II, no. 4 (July 1971), 14.

Roland, Jon. "AI, UNIX and C," UNIX/WORLD, I, no. 3 (1984), 97-103.

Solomon, Les. "Motorola's Muscular 68020," Computers & Electronics, XXII, no. 10 (October 1984), 74-79.

Switzer, David K. The Minister as Crisis Counselor. Nashville: Abingdon Press, 1974.

Warner, Homer R. Computer-Assisted Medical Decision-Making. New York: Academic Press, 1979.

## APPENDIX A: CAC System Documentation

This appendix contains the documentation needed to use the CAC system. It includes only the documentation of the major programs and files used in the system; for knowledge of subroutines, minor files, and auxiliary programs is not needed at this level. Program documentation included here consists of:

analyze - ANALYZE OFFERING TRENDS

clndr - CALENDAR CONVERSION

valid - VALIDATE MASTER FILE

weekly - CAPTURE WEEKLY OFFERING

wkmerg - MERGE WEEKLY OFFERING

The following files are also discussed:

confidence file

master file

weekly file

Documentation is arranged alphabetically by program name, and the files are intermixed alphabetically with the programs. Descriptive headings (as seen above) are used at the top of each page to facilitate the identification of the material being described on any given page.

## ANALYZE OFFERING TRENDS

### NAME

**analyze - analyze offering trends for possible problem indicators**

### DESCRIPTION

Analyze is a Pascal program which will analyze offering trends in order to find indicators of possible problems that may exist in a family's life. This program is the heart of the CAC expert system, and it is written modularly so that rules of inference may be added or deleted as seen fit without necessitating a major rewrite of the program or system.

Analyze is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following are the messages which can be expected and allowable responses:

```
Enter file name containing confidence levels:
Enter input file name:
```

Both statements expect a valid file name construct. Both files should exist, and the contents of the file containing confidence levels should conform to the format of confidence file while the contents of the input file should conform to the format of master file.

```
Enter processing date (Julian ddd form):
```

This statement expects an integer value of three or fewer digits in length. The integer is interpreted to be the number of days that have elapsed in the year to be processed (e.g. 46 = February 15; 31 days in January plus 15 days in February). The date entered will be used as the date upon which all processing will be focused and no data beyond (after) this date will be used in any processing. Obviously, since this value is the number of days elapsed in a year, it can never be greater than the maximum number of days in any year which is 366 days. If the value entered exceeds this value, the user may expect the following statement to be printed:

Bad process date entered 366 assumed!

The last statement expecting a response relates to the process date, involves a yes or no answer, and asks:

Is this date to be considered a holiday?

If answered in the affirmative, a flag will be set in the program which will facilitate special processing (ordinarily this special processing will merely be an ignorance of data which is expected to be abnormal with respect to the remainder of the year). Any response which does not begin with a 'y' will be considered to be a negative response and will facilitate normal processing.

The remainder of the program will draw its input from the confidence file and from the master file. The confidence file and the header portion of the master file will be loaded in their entirety while the data portion of the master file will be processed by envelope. As each envelope's offering amounts are entered, the data will be processed through the various rules of inference which apply. The rules which are currently implemented and their appropriate responses follow:

[1] If the offering amount for the process date is non-zero, analyze using linear regression the sums of the offering amounts up to but not including the process date. With this regression line, extrapolate the value which might be expected for the process date; and using confidence intervals, compare this value with the actual value. If the actual value falls outside the range of the smallest confidence level, the following message will be printed:

Offering for envelope # nnn is (above/below) the level indicating possible (message from confidence file).

where the message from the confidence file will be the message associated with the largest confidence level range that the value falls

outside. Also indicated in this message is the fact that the actual value is above or below the indicated range.

[2] If the process date is considered to be a holiday, positive discrepancies of the above rule will be ignored. An alternative to this rule would be the use of different confidence values with higher ranges, but this might cause some meaningful negative discrepancies to be overlooked. A combination of this rule and the use of different confidence values in a separate run of the program is recommended.

[3] If the offering amount for the process date is zero, analyze the frequency of giving and determine if the time elapsed since the last offering is unreasonably long. If the time elapsed since the last offering seems to be unreasonable, the following message will be printed:

> Giving habits for envelope # nnn seem to have changed

In order to keep these statements to a reasonable number (i.e., to not print messages for long-inactive members or for families who who no longer hold membership), a maximum and a minimum are placed upon the unreasonable range.

DIAGNOSTICS

The only diagnostic messages incorporated in this program deal with the order of records in the input file. These are here merely as a token, as there is a program whose sole function is to perform data validation on the master file(s).

BUGS

Since only the current year's data was available, the program was designed to analyze the data for one calendar year only. This will be a problem for early months of a year as there will be an insufficient base of data for meaningful trends to be established.

## ANALYZE OFFERING TRENDS (continued)

When a data base of more than one year is available, this program should be revised to analyze the data for two consecutive calendar years (note that this will be no trivial task as in many churches envelopes are assigned to different families each year and coordination from one year to the next will be a formidable task to overcome).

SEE ALSO

confidence file
master file

## CALENDAR CONVERSION

### NAME

**clndr - create a Gregorian-Julian conversion calendar**

### DESCRIPTION

Clndr is a Pascal program which will create a conversion calendar that is designed to aid an individual in working with the various programs and files in the CAC system. It requires as input a file which contains a date record (for details on its format, see master file documentation).

Clndr is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following are the messages which can be expected and allowable responses:

```
Enter input file name:
Enter output file name:
```

Both statements expect a valid file name construct; the input file should exist and its contents should conform to the header portion of master file; the output file should not previously exist as only the output from this program will be present in the file upon its completion!

### SEE ALSO

master file

## CONFIDENCE FILE

NAME

confidence file - confidence level data file for CAC system

DESCRIPTION

Confidence file is used in the analysis of offering data for the CAC system. It contains information pertinent only to the rules of inference being processed and currently consists of the following records:

[1] (dormancy factor) : this record shall contain a free form valid integer or real number (no exponential notation) representing the factor to be used in determining reasonableness for the length of time since the last offering. This statement must be the first in the file, and there may only be one such record. [dormancy factor record]

[2] (confidence level value)-(confidence level message) : this record shall contain a free form valid integer or real number (no exponential notation) representing the confidence level of a two-tail Student's t-Distribution test (100 - a/2); a dash (-) must follow immediately after the free form number; and a message associated with the confidence level must immediately follow the dash. The remainder of the file may contain from one to seventeen of this type of statement. [confidence level record]

SEE ALSO

analyze - ANALYZE OFFERING TRENDS

## MASTER FILE

NAME

master file - offering data file for CAC system

DESCRIPTION

Master file is used in every phase of the CAC system. It contains all major information pertinent to the system as a whole and consists of the following records:

[1] (year)-(first day of week) : columns one through four shall contain the year for the data to be processed; column five must contain a dash (-); and column six must contain a code representing the day of the week upon which the year starts (1-Sunday, 2-Monday, 3-Tuesday, etc.). [date record]

[2] t(category code)(period code)-(category title) : column one shall contain a 't'; column two shall contain a single digit category code number; column three shall contain a valid period code of d (daily), w (weekly), m (monthly), or s (special); column four must contain a dash (-); and columns five and following shall contain the title of the category associated with the code. [title record]

[3] (Julian date [less year])-(special title) : columns one through three must contain the Julian date for the special event; column four must contain a dash (-); and columns five and following shall contain the title of the special event. The Julian date shall consist of the number of days that have elapsed in the year (e.g. 046 = February 15: 31 days in January plus 15 days in February). These records may only follow a title record (above) which has an s (special) as the period code. [special record]

[4] ========== : column one must contain an equal sign (=); the remainder of the record should contain characters which will create a visible division in the file (more equal signs

are recommended). [sentinel record]

[5] ***-(envelope number) : columns one through three must contain asterisks (*); column four must contain a dash (-); and columns five through seven shall contain the number of the envelope which the following records (until the next record of this type) will be in reference. [envelope master record]

[6] *(category code)*-(envelope number) : column one must contain an asterisk (*); column two shall contain a single digit category code which was defined in one of the title records above; column three must contain an asterisk (*); column four must contain a dash (-); and columns five through seven shall contain the envelope number defined by the associated envelope master record above. [category record]

[7] (Julian date [less year])-(offering amount) : columns one through three must contain the Julian date on which the offering was made; column four must contain a dash (-); and columns five through twelve shall contain an offering amount. The remainder of the file shall contain records of this type as well as its associated envelope master and category records. [detail record]

SEE ALSO

mkmast - MAKE MASTER FILE
valid - VALIDATE MASTER FILE
wkmerg - MERGE WEEKLY OFFERING

## MAKE MASTER FILE

NAME

**mkmast - make an initial master file of offering amounts**

DESCRIPTION

Mkmast is a Pascal program which is to be used in the initial creation of a church offering master file. It requires as input a file which contains a date record and title records (for details on their formats see master file documentation).

Mkmast is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following are messages which can be expected and allowable responses:

```
Enter input file name:
Enter output file name:
```

Both statements expect a valid file name construct; the input file should exist and its contents should conform to the header portion of master file. The output file should not previously exist as only the output from this program will be present in the file upon its completion!

```
Input first envelope number:
Input last envelope number:
```

Both statements expect an integer in the range of 1 to 999 inclusive; the first number must be numerically less than the last number!

```
Any input for envelope # NNN ?
```

This statement is looking for a yes or no response, but two other responses are recognized. 1) a carriage return with no other keys hit will be construed as a "no" answer and 2) typing the word "break" will terminate the program saving all data collected to this point.

Any input for CATEGORY NAME ?

This statement is looking for a yes or no response, and a carriage return with no other keys hit will be construed as a "no" answer.

MONTH NAME week # NNN :

This statement expects the offering amount to be entered as either an integer or real value. It also accepts several codes which help in locating the appropriate date: 1) a carriage return with no other keys hit will be construed as a zero offering amount and will make no entry in the master file; 2) a minus sign (-) will allow the entry of the previous date; 3) a plus sign (+) will allow the entry of the date four weeks in the future (skip ahead feature); 4) the word "duplicate" will enter the value of the previous date as the current value (remember that if a carriage return is used by itself, the previous value is zero!); and 5) typing the word "break" will terminate the current category and proceed immediately to the next.

As a shorthand notation, any of the above responses may be shortened to the first character of the word.

DIAGNOSTICS

The only diagnostic messages incorporated in this program deal with the title records in the input file. These are here merely as a token as there is a program whose sole function is to perform data validation on the master file(s).

BUGS

Since the output of this program is written to the file in blocks, there may be times when the prompting message is delayed.

SEE ALSO

master file
valid - VALIDATE MASTER FILE

## VALIDATE MASTER FILE

NAME

valid - perform validity checking on master file

DESCRIPTION

Valid is a Pascal program which is to be used to check a file (which has the format of a master file ) for records which are out of order, records which do not adhere to proper form, and data in a record which is invalid or unexpected.

Valid is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following message

Enter input file name:

expects a valid file name construct; the file should exist and its contents should conform to the format of master file. All other input will be taken from the specified file; thus no further user interaction is required.

Diagnostic output from valid will consist of pairs of lines. The first line of each pair will be in the form:

nnn ---> line

where nnn is the relative line number of line in the file to be checked. The second line of each pair will be a message indicating the reason that the previous line has been flagged. Note that only lines in the file that are questionable will be flagged or printed. The various messages and their appropriate meanings follow:

BAD CATEGORY (should be between 1 and 9 inclusive) or
BAD CATEGORY IN TITLE RECORD : the single digit category code is used as an index and thus must be between 1 and 9 inclusive; found a value which does not comply.

BAD DATE or
BAD DATE IN SPECIAL RECORDS : Julian dates must be numeric and fall within the range of 0 and the number of days in the year (365 normally, 366 leap year) inclusive; found a value which does not comply.

BAD FIRST DAY OF WEEK : first day of week must be between 1 (Sunday) and 7 (Saturday) inclusive; found a value which does not comply.

BAD OFFERING AMOUNT : offering amount must be numeric and non-negative; encountered illegal offering amount.

BAD PERIOD CODE IN TITLE RECORD : valid period codes are d (daily), w (weekly), m (monthly), and s (special) only; found a value which does not comply.

BAD YEAR : a reasonableness check: year must be between 1900 and 2100 inclusive; found a value which does not comply.

CATEGORY AND ENVELOPE RECORDS DIFFERENT ENVELOPES : category records following envelope master records must have the same envelope number; category record encountered with different envelope number from associated envelope master record.

CATEGORY OUT OF ORDER : categories must be in ascending order.

DATES FOR SPECIAL RECORDS OUT OF ORDER or
DATES OUT OF ORDER : dates must be in chronological order.

DUPLICATE CATEGORY : encountered a category record with a category number which has already been processed for the current envelope.

DUPLICATE DATES or
DUPLICATE DATES IN SPECIAL RECORDS : only one entry for each date may exist; found more than one.

## VALIDATE MASTER FILE (continued)

DUPLICATE ENVELOPE : encountered an envelope master record with an envelope number which has already been processed; only one is allowed.

ENVELOPE MASTER IMMEDIATELY FOLLOWS ANOTHER : expected category record after envelope master; found another envelope master.

ENVELOPE OUT OF ORDER : envelopes must be in ascending order.

MALFORMED RECORD : most records require a dash (-) in column four (4); expected for this record but not found.

PURGED DUE TO PREVIOUS ERROR : records that are not in the proper order will be purged until a record which is expected for this part of the file is encountered.

STATEMENTS OUT OF ORDER (expecting category record)
or
STATEMENTS OUT OF ORDER (expecting envelope master)
or
STATEMENTS OUT OF ORDER (expecting sentinel line) : certain records must follow certain other records; encountered a record which is illegal in this position.

TITLE MISSING OR NOT LEFT JUSTIFIED : all titles must start in position five (5); encountered a record in which the title is not entered properly.

TOO MANY BAD RECORDS --- PROGRAM TERMINATED : a software limit of 100 has been placed on the number of consecutive records which are out of order; this limit has been exceeded.

TOO MANY SPECIAL RECORDS (24 maximum) : a software limit of 24 entries has been placed on the number of special records allowed; this limit has been exceeded.

ZERO OFFERING AMOUNT : encountered an offering amount that was zero.

SEE ALSO

master file

## WEEKLY FILE

NAME

**weekly file - weekly offering data file for CAC system**

DESCRIPTION

Weekly file is used in the capture of the weekly offering data for the CAC system. It contains information pertinent only to the weekly offering being processed and consists of the following records:

[1] (year)-(first day of week) : columns one through four shall contain the year for the data to be processed; column five must contain a dash (-); and column six must contain a code representing the day of the week upon which the year starts (1-Sunday, 2-Monday, 3-Tuesday, etc.). [date record]

[2] (Julian date [less year]) : columns one through three must contain the Julian date for the data to be processed. The Julian date shall consist of the number of days that have elapsed in the year (e.g. 046 = February 15: 31 days in January plus 15 days in February). [process date record]

[3] t(category code)(period code)-(category title) : column one shall contain a 't'; column two shall contain a single digit category code number; column three shall contain a valid period code of d (daily), w (weekly), m (monthly), or s (special); column four must contain a dash (-); and columns five and following shall contain the title of the category associated with the code. [title record]

[4] (envelope number)-(offering amount) : columns one through three shall contain an envelope number; column four must contain a dash (-); and columns five through twelve shall contain an offering amount. The remainder of the file shall contain records of this type only. [weekly entry record]

SEE ALSO

weekly - CAPTURE WEEKLY OFFERINGS
wkmerg - MERGE WEEKLY OFFERING

## CAPTURE WEEKLY OFFERINGS

NAME

weekly - capture weekly offering amounts by envelope

DESCRIPTION

Weekly is a Pascal program which is to be used to capture the weekly church offering amounts by envelope number. It requires as input a file which contains the header portion of a master file. Output will be a file in the form of weekly file. After this program has been run, it will be necessary to run another program to merge the weekly file with the master file (for details see wkmerg documentation).

Weekly is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following are the messages which can be expected and allowable responses:

Enter input file name:
Enter output file name:

Both statements expect a valid file name construct; the input file should exist and its contents should conform to the header portion of master file; the output file should not previously exist as only the output from this program will be present in the file upon its completion!

Input date (mm/dd):

Here the user should input the date of the data to be captured; NOT THE CURRENT DATE unless they are the same. It must be entered as a two-digit month field (e.g. 01 = January, 11 = November) followed immediately by a slash (/) which in turn is followed immediately by a two digit day of month field (e.g. February 15 = 02/15). The date will be translated into a Julian date by the program.

Input code of category to be used:

[redacted] statement will follow the printing of a list of categories with their respective codes. The user

should select the code of the category which applies to the data to be processed and enter that code to this command. Note that codes will always be between 1 and 9 inclusive, and the upper bound may be considerably lower than 9.

Working with (category title) for mm/dd(ddd)

If everything has proceeded correctly to this point, the message above will be printed. If there is any problem with the data printed in this statement, the user should interrupt the program and discard the weekly output file.

Input first envelope number:
Input last envelope number:

Both statements expect a one to three digit integer which will be interpreted to be an envelope number. This range of envelope numbers will be used in printing the following prompts for input:

nnn -

where nnn is the envelope number for which data will be stored. These prompts will be printed sequentially from the "first envelope number" entered to the "last envelope number" entered unless one of the allowable special codes are entered for the prompted envelope. Allowable responses to the prompt are: 1) any valid offering amount which will be the only response which will record anything in the weekly output file; 2) a carriage return with no other keys hit will be construed as a zero offering amount (zero offering amounts will not be recorded in the weekly output file); 3) a minus sign (-) will allow the entry of the previous envelope (backup feature); 4) a plus sign (+) will allow the entry of the envelope ten ahead of the current envelope (skip ahead feature); and 5) typing the word "break" (or just a 'b' alone as a shorthand notation will suffice) will terminate the program.

Enter input for envelope #

This statement will precede the list of prompts.

DIAGNOSTICS

The only diagnostic messages incorporated in this program deal with the title records in the input file. These are here merely as a token as there is a program whose sole function is to perform data validation on the master file(s).

BUGS

Since the output of this program is written to the file in blocks, there may be some times when the prompting message is delayed.

SEE ALSO

master file
weekly file
wkmerg - MERGE WEEKLY OFFERING

## MERGE WEEKLY OFFERING

### NAME

wkmerg - merge weekly offering file with master file

### DESCRIPTION

Wkmerg is a Pascal program which will merge a weekly file with a master file. It requires little interaction and runs for a relatively long time; because of this, an audible signal has been included at the end of the program to aid in monitoring program completion.

Wkmerg is an interactive program which prompts for all necessary information and does not allow for any information to be entered through the command line. The following are the messages which can be expected and allowable responses:

```
Enter master file name:
Enter data file name:
Enter new master file name:
```

Each of these statements expects a valid file name construct; the master file should exist and its contents should conform to that of master file; the data file should exist and its contents should conform to that of weekly file; the new master file should not previously exist as only the output from this program will be present in the file upon its completion!

### DIAGNOSTICS

Wkmerg will run with no output to the user unless there is one of two possible problems. The following are the messages, reasons for the messages, and action taken:

```
Weekly data year not same as master year!
Weekly = (year1) Master = (year2)
ABORT
```

This group of messages will be printed if the year in the weekly file (year1) does not match that in the master file (year2). As the third statement implies, the program will be terminated immediately and the

new master file should be discarded.

Weekly data already present!
Envelope = (nnn) Date = (ddd)
ABORT

This group of messages will be printed if a record is found in the master file that bears the same date as the process date of the weekly file. The envelope number (nnn) and the date (ddd) are printed to facilitate reconciliation of the discrepancy. As with the previous group of messages, the program will be terminated immediately and the new master file should be discarded.

SEE ALSO

master file
valid - VALIDATE MASTER FILE
weekly - CAPTURE WEEKLY OFFERINGS
weekly file

## APPENDIX B: Student's t-Distribution

This appendix contains the values of Student's t-Distribution used in building confidence intervals in the CAC system. The first column of each page lists the number of degrees of freedom. The headings of the other columns give probabilities (%) for the entry value to fall within the range of -t to t (two-tail test).*

* Table compiled from the following sources:
Edward J. Dudewicz, Introduction to Statistics and Probability (New York: Holt, Rinehart and Winston, 1976), p. 468,
Paul G. Hoel, Sidney C. Port, and Charles J Stone, Introduction to Statistical Theory (Boston: Houghton Mifflin Company, 1971), p. 227, and
Byron L. Newton, Statistics for Business (Chicago: Science Research Associates, 1973), p. 499.

## Student's t-Distribution

| df | 10.0 | 20.0 | 30.0 | 40.0 | 50.0 | 60.0 |
|---|---|---|---|---|---|---|
| 1 | 0.158 | 0.325 | 0.510 | 0.727 | 1.00000 | 1.376 |
| 2 | 0.142 | 0.289 | 0.445 | 0.617 | 0.81650 | 1.061 |
| 3 | 0.137 | 0.277 | 0.424 | 0.584 | 0.76489 | 0.978 |
| 4 | 0.134 | 0.271 | 0.414 | 0.569 | 0.74070 | 0.941 |
| 5 | 0.132 | 0.267 | 0.408 | 0.559 | 0.72669 | 0.920 |
| 6 | 0.131 | 0.265 | 0.404 | 0.553 | 0.71756 | 0.906 |
| 7 | 0.130 | 0.263 | 0.402 | 0.549 | 0.71114 | 0.896 |
| 8 | 0.130 | 0.262 | 0.399 | 0.546 | 0.70639 | 0.889 |
| 9 | 0.129 | 0.261 | 0.398 | 0.543 | 0.70272 | 0.883 |
| 10 | 0.129 | 0.260 | 0.397 | 0.542 | 0.69981 | 0.879 |
| 11 | 0.129 | 0.260 | 0.396 | 0.540 | 0.69745 | 0.876 |
| 12 | 0.128 | 0.259 | 0.395 | 0.539 | 0.69548 | 0.873 |
| 13 | 0.128 | 0.259 | 0.394 | 0.538 | 0.69384 | 0.870 |
| 14 | 0.128 | 0.258 | 0.393 | 0.537 | 0.69242 | 0.868 |
| 15 | 0.128 | 0.258 | 0.393 | 0.536 | 0.69120 | 0.866 |
| 16 | 0.128 | 0.258 | 0.392 | 0.535 | 0.69013 | 0.865 |
| 17 | 0.128 | 0.257 | 0.392 | 0.534 | 0.68919 | 0.863 |
| 18 | 0.127 | 0.257 | 0.392 | 0.534 | 0.68837 | 0.862 |
| 19 | 0.127 | 0.257 | 0.391 | 0.533 | 0.68763 | 0.861 |
| 20 | 0.127 | 0.257 | 0.391 | 0.533 | 0.68696 | 0.860 |
| 21 | 0.127 | 0.257 | 0.391 | 0.532 | 0.68635 | 0.859 |
| 22 | 0.127 | 0.256 | 0.390 | 0.532 | 0.68580 | 0.858 |
| 23 | 0.127 | 0.256 | 0.390 | 0.532 | 0.68531 | 0.858 |
| 24 | 0.127 | 0.256 | 0.390 | 0.531 | 0.68485 | 0.857 |
| 25 | 0.127 | 0.256 | 0.390 | 0.531 | 0.68443 | 0.856 |
| 26 | 0.127 | 0.256 | 0.390 | 0.531 | 0.68405 | 0.856 |
| 27 | 0.127 | 0.256 | 0.389 | 0.531 | 0.68370 | 0.855 |
| 28 | 0.127 | 0.256 | 0.389 | 0.530 | 0.68335 | 0.855 |
| 29 | 0.127 | 0.256 | 0.389 | 0.530 | 0.68304 | 0.854 |
| 30 | 0.127 | 0.256 | 0.389 | 0.530 | 0.68276 | 0.854 |
| 40 | 0.126 | 0.255 | 0.388 | 0.529 | 0.68066 | 0.851 |
| 60 | 0.126 | 0.254 | 0.387 | 0.527 | 0.67862 | 0.848 |
| 120 | 0.126 | 0.254 | 0.386 | 0.526 | 0.67656 | 0.845 |
| inf | 0.126 | 0.253 | 0.385 | 0.524 | 0.67449 | 0.842 |

## Student's t-Distribution (continued)

| df | 70.0 | 75.0 | 80.0 | 90.0 | 95.0 | 97.5 |
|---|---|---|---|---|---|---|
| 1 | 1.963 | 2.4142 | 3.078 | 6.3138 | 12.7060 | 25.4520 |
| 2 | 1.386 | 1.6036 | 1.886 | 2.9200 | 4.3027 | 6.2053 |
| 3 | 1.250 | 1.4226 | 1.638 | 2.3534 | 3.1825 | 4.1765 |
| 4 | 1.190 | 1.3444 | 1.533 | 2.1318 | 2.7764 | 3.4954 |
| 5 | 1.156 | 1.3009 | 1.476 | 2.0150 | 2.5706 | 3.1634 |
| 6 | 1.134 | 1.2733 | 1.440 | 1.9432 | 2.4469 | 2.9687 |
| 7 | 1.119 | 1.2543 | 1.415 | 1.8946 | 2.3646 | 2.8412 |
| 8 | 1.108 | 1.2403 | 1.397 | 1.8595 | 2.3060 | 2.7515 |
| 9 | 1.100 | 1.2297 | 1.383 | 1.8331 | 2.2622 | 2.6850 |
| 10 | 1.093 | 1.2213 | 1.372 | 1.8125 | 2.2281 | 2.6338 |
| 11 | 1.088 | 1.2145 | 1.363 | 1.7959 | 2.2010 | 2.5931 |
| 12 | 1.083 | 1.2089 | 1.356 | 1.7823 | 2.1788 | 2.5600 |
| 13 | 1.079 | 1.2041 | 1.350 | 1.7709 | 2.1604 | 2.5326 |
| 14 | 1.076 | 1.2001 | 1.345 | 1.7613 | 2.1448 | 2.5096 |
| 15 | 1.074 | 1.1967 | 1.341 | 1.7530 | 2.1315 | 2.4899 |
| 16 | 1.071 | 1.1937 | 1.337 | 1.7459 | 2.1199 | 2.4729 |
| 17 | 1.069 | 1.1910 | 1.333 | 1.7396 | 2.1098 | 2.4581 |
| 18 | 1.067 | 1.1887 | 1.330 | 1.7341 | 2.1009 | 2.4450 |
| 19 | 1.066 | 1.1866 | 1.328 | 1.7291 | 2.0930 | 2.4334 |
| 20 | 1.064 | 1.1848 | 1.325 | 1.7247 | 2.0860 | 2.4231 |
| 21 | 1.063 | 1.1831 | 1.323 | 1.7207 | 2.0796 | 2.4138 |
| 22 | 1.061 | 1.1816 | 1.321 | 1.7171 | 2.0739 | 2.4055 |
| 23 | 1.060 | 1.1802 | 1.319 | 1.7139 | 2.0687 | 2.3979 |
| 24 | 1.059 | 1.1789 | 1.318 | 1.7109 | 2.0639 | 2.3910 |
| 25 | 1.058 | 1.1777 | 1.316 | 1.7081 | 2.0595 | 2.3846 |
| 26 | 1.058 | 1.1766 | 1.315 | 1.7056 | 2.0555 | 2.3788 |
| 27 | 1.057 | 1.1757 | 1.314 | 1.7033 | 2.0518 | 2.3734 |
| 28 | 1.056 | 1.1748 | 1.313 | 1.7011 | 2.0484 | 2.3685 |
| 29 | 1.055 | 1.1739 | 1.311 | 1.6991 | 2.0452 | 2.3638 |
| 30 | 1.055 | 1.1731 | 1.310 | 1.6973 | 2.0423 | 2.3596 |
| 40 | 1.050 | 1.1673 | 1.303 | 1.6839 | 2.0211 | 2.3289 |
| 60 | 1.046 | 1.1616 | 1.296 | 1.6707 | 2.0003 | 2.2991 |
| 120 | 1.041 | 1.1559 | 1.289 | 1.6577 | 1.9799 | 2.2699 |
| inf | 1.036 | 1.1503 | 1.282 | 1.6449 | 1.9600 | 2.2414 |

## Student's t-Distribution (continued)

| df | 98.0 | 99.0 | 99.5 | 99.8 | 99.9 |
|---|---|---|---|---|---|
| 1 | 31.821 | 63.6570 | 127.3200 | 318.310 | 636.619 |
| 2 | 6.965 | 9.9248 | 14.0890 | 22.327 | 31.598 |
| 3 | 4.541 | 5.8409 | 7.4533 | 10.214 | 12.941 |
| 4 | 3.747 | 4.6041 | 5.5976 | 7.173 | 8.610 |
| 5 | 3.365 | 4.0321 | 4.7733 | 5.893 | 6.859 |
| 6 | 3.143 | 3.7074 | 4.3168 | 5.208 | 5.959 |
| 7 | 2.998 | 3.4995 | 4.0293 | 4.785 | 5.405 |
| 8 | 2.896 | 3.3554 | 3.8325 | 4.501 | 5.041 |
| 9 | 2.821 | 3.2498 | 3.6897 | 4.297 | 4.781 |
| 10 | 2.764 | 3.1693 | 3.5814 | 4.144 | 4.587 |
| 11 | 2.718 | 3.1058 | 3.4966 | 4.025 | 4.437 |
| 12 | 2.681 | 3.0545 | 3.4284 | 3.930 | 4.318 |
| 13 | 2.650 | 3.0123 | 3.3725 | 3.852 | 4.221 |
| 14 | 2.624 | 2.9768 | 3.3257 | 3.787 | 4.140 |
| 15 | 2.602 | 2.9467 | 3.2860 | 3.733 | 4.073 |
| 16 | 2.538 | 2.9208 | 3.2520 | 3.686 | 4.015 |
| 17 | 2.567 | 2.8982 | 3.2225 | 3.646 | 3.965 |
| 18 | 2.552 | 2.8784 | 3.1966 | 3.610 | 3.922 |
| 19 | 2.539 | 2.8609 | 3.1737 | 3.579 | 3.883 |
| 20 | 2.528 | 2.8453 | 3.1534 | 3.552 | 3.850 |
| 21 | 2.518 | 2.8314 | 3.1352 | 3.527 | 3.819 |
| 22 | 2.508 | 2.8188 | 3.1188 | 3.505 | 3.792 |
| 23 | 2.500 | 2.8073 | 3.1040 | 3.485 | 3.767 |
| 24 | 2.492 | 2.7969 | 3.0905 | 3.467 | 3.745 |
| 25 | 2.485 | 2.7874 | 3.0782 | 3.450 | 3.725 |
| 26 | 2.479 | 2.7787 | 3.0669 | 3.435 | 3.707 |
| 27 | 2.473 | 2.7707 | 3.0565 | 3.421 | 3.690 |
| 28 | 2.467 | 2.7633 | 3.0469 | 3.408 | 3.674 |
| 29 | 2.462 | 2.7564 | 3.0380 | 3.396 | 3.659 |
| 30 | 2.457 | 2.7500 | 3.0298 | 3.385 | 3.646 |
| 40 | 2.423 | 2.7045 | 2.9712 | 3.307 | 3.551 |
| 60 | 2.390 | 2.6603 | 2.9146 | 3.232 | 3.460 |
| 120 | 2.358 | 2.6174 | 2.8599 | 3.160 | 3.373 |
| inf | 2.326 | 2.5758 | 2.8070 | 3.090 | 3.291 |

# VITA

Kevin Eric Knauss

Born at Lower Bucks County Hospital in Bristol, Bucks County, Pennsylvania on the twenty-fourth day of April in the year of our Lord Nineteen Hundred Fifty-Six to Harold LeRoy Knauss (father) and Helen Elizabeth (Hilgendorff) Knauss (mother).

Institutions attended:

Moravian Theological Seminary, Bethlehem, PA 18018. 40 credit hours of course work. Emphasis in Pastoral Counseling. Served as chairman of recreation committee. Financed expenses by part-time and summer employment.

Rider College, Lawrenceville, NJ 08648. Received B.S. in Commerce in August, 1977. Major in Accounting with strong preparation in Decision Sciences and Computers. Elected member of Omicron Delta Epsilon International Honor Society. Financed expenses by part-time employment and partial grant-in-aid.

Bucks County Community College, Newtown, PA 18940. Received A.A. in June, 1976. Major in Business Administration with strong preparation in Co[redacted] Sci-

ence. Elected member of Phi Theta Kappa National Honor Fraternity. Financed expenses by part-time employment.

Pennsbury High School, Fairless Hills, PA 19030. Received Diploma in June, 1974. Participated in journalism, music, sports, and theatre extracurricular activities. Received "Most Valuable Staffer Award" for outstanding contributions to the school newspaper in 1972. Held part-time employment through entire four years.

Employment:

Assistant Professor of Computer Science -- Northampton County Area Community College, Bethlehem, PA 18017 -- Spring 1984 temporary; currently part-time.

Youth Director and Student Minister -- Saint Paul's United Church of Christ, Trexlertown, PA 18087 -- 1982-1984.

Computer Systems Programmer / Analyst -- Pennsylvania Power and Light Company, Allentown, PA 18101 -- 1979-1982.

Data Processing Professional Assistant -- Northampton County Area Community College, Bethlehem, PA 18017 -- 1978-1979.