The AIR Professional File Summer 1980
The Association for Institutional Research No. 6

CHOOSING A COMPUTER LANGUAGE FOR INSTITUTIONAL RESEARCH

Denise Strenglein
Data Base Coordinator
University of South Florida

A strongly felt need for a multipurpose and simple-to-use computer language emerges as more and more institutional research offices acquire computer terminals. A veritable Babel of computer languages exists (Ryland, 1979), and a good deal of thought should be given to choosing the right language. Learning a computer language represents a significant investment in time and effort, and because of the problem of training new staff when turnover occurs, only one or, at most, two languages should be introduced into an office.

The state of the art in languages has progressed far enough that, in most instances, institutional researchers need not become full-fledged computer programmers in order to access computer data. Many of the high-level languages have taken the drudgery out of programming, and having the computer do much of the data manipulation takes the drudgery out of institutional research.

The choice of the best computer language depends on the sophistication of the staff, the types of planned application, the size and type of computer, and the availability of central programming support.

Training Considerations

The characteristics that make learning and using a language easier for the institutional researcher who has little computer background are these: syntax that looks like ordinary language; a well-written manual; availability of training programs; a language that is somewhat forgiving (that is, one that does not come to a dead stop for every little infraction of the rules); a minimum of required coding; clear, understandable error messages; and job control language grouped at one end or the other of a multistep job. On-campus assistance from someone who knows the language is essential for the neophyte, especially in learning the system-specific job control language for the institution's installation; a good place to start is with an undergraduate computer course in any language or a beginner's program offered by the computer center. Certain fundamental concepts are common to all languages. These include file structure and input mediums, flow charting or other planning techniques, language coding, and job control language. Once these are learned, it is possible to gain a useful amount of skill in almost any of the report or statistical languages with a manual, relatively unimpeded access to a computer, and a lot of patience. Of course, a training program helps, and access to someone who knows the language is even better.

Applications

The types of projects done by the institutional research office should be considered next. There are two basic classes of computer applications in institutional research. The first and most common is the sort-accumulate-list type of report. This includes such things as count of students by race, by sex, by classification; aggregation of student credit hours by discipline, by level; cost figures by department; and government equal opportunity reports, among many others. The second type of application involves descriptive and inferential statistical analysis at varying levels of sophistication. This includes, for example, enrollment projection, analysis of questionnaires, salary regression analysis, and analysis of predictors of student success. An efficient language will do both sets of applications.

Both types of application require file handling, and the file-handling ability of the language chosen is of utmost importance. Institutional research offices use data from many sources and rarely can control file formats. While the easiest files to use are fixed length sequential, more often than not the nature of institutional research projects requires reading variable-length, COBOL-generated files or matching information from a number of different files. It is desirable to be able to extract small sets of information from large files in order to reduce run time and cost.

There are some automatic features which simplify report writing: opening and closing of files, reading and writing of records, movement of data from input to output, initializing variables, accumulating totals, sorting, and report formatting with overrides available. Clear and understandable selection logic and the ability to manipulate character data as easily as numeric data are also important.

It is possible to get usable, if not pretty, results almost immediately with automatic features, providing both a more rapid return on invested time and an incentive to continue learning. Automatic features, however, also tend to remove control from the programmer, so overrides are necessary if anything complex or unusual is to be produced.

Descriptive statistics such as percentages, frequency distributions, and means are, by far, the most common applications of statistical analysis; some of the nonstatistical packages offer these features. The next most frequently used statistics are regression and analysis of variance; one of the statistical packages will have to be selected for these uses. It should be noted that some of the statistical packages are beginning to offer both statistics and report features.

Size and Type of Computer

While some of the more common languages—for example, Fortran and COBOL—are machine independent, being available on almost any brand of computer, others—like IBM's PL/1—have been developed for only one type. IBM and IBM-type machines such as Amdahl, certain Itel, CDC, Magnuson, and Ryad models, comprise much of the computer market. For this reason, languages designed for IBM computers tend to dominate. It should be noted that even the nominally standardized languages, COBOL among them, are highly machine dependent since each manufacturer tends to introduce features which take advantage of that particular computer's unique capabilities.

Size is also a factor. The amount of computer memory available sets limits on the capabilities of the language and on the size of the data sets and number of variables a given job can handle. Some packages—SPSS and MarkIV, for example—come in versions to fit progressively larger computers. Other languages, such as Basic, are designed to operate best on small computers. As distributed data processing using microcomputers becomes more prevalent, available memory becomes more of a limiting factor, at least for modes of operation which are independent of a large central mainframe.

Normally, an institutional research office will simply adopt a language that is already available locally. However, if circumstances arise in which an institutional research office "goes

shopping" for a report or statistical language, it would be wise to get advice from someone who is familiar with hardware.

Availability of Central Computing Support

The motivation for acquiring an office terminal, in many cases, is the lack of sufficient resources in a central computing facility to provide all the production programming needed for reports, especially *ad hoc* reports. However, there are many levels of independence from central support.

If central support is nonexistent due either to lack of hardware capability or staff, the institutional research office might consider acquiring a micro- or minicomputer and hiring or training a full-time programmer-analyst who can create and document an internal data system. (The documentation is even more important than the system itself.) Another option is to tie into a network on a timeshare basis, but since the cost of trial and error can be truly awesome at the rates charged for computer time, the presence in the office of a computer professional is recommended.

More commonly, the central computing facility can provide support for routine repetitive reports, but *ad hoc* reports or summaries in a slightly different order remain time-consuming hand jobs. The regular institutional research professionals, in this case, can often learn a report or statistical language well enough to increase dramatically the efficiency and timeliness of these reports and the productivity of the office.

In the enviable case where the level of central support is high, the institutional research office might still want to use one of the statistical packages for more sophisticated analysis. In addition, if a database query language is available, the institutional research staff can save time and misunderstandings by specifying reports directly.

Types of Language

A language for use by institutional researchers, in most cases, should allow people who are not programmers to produce reports or do analysis independent of central computer-center support and with a minimum of coding.

Ryland (1979) gives a comprehensive description of the types of proprietary software available today. What follows is a somewhat in-depth examination of representative samples of different types of language. Table 1 summarizes the author's opinion concerning language features of COBOL, Fortran, MarkIV, Easytrieve, SPSS and SAS. The ratings range from "−2" which means poor or hard to use, to "+2," powerful or easy to use. A "zero" signifies that the function does not exist or is not applicable to the language.

Programmer Languages

Two of the oldest and most commonly used languages are COBOL and Fortran. These are machine independent languages which usually come with the larger computers, and national standards exist for them. COBOL was developed for business applications and uses ordinary English for its command syntax. However, it tends to be excessively longwinded, and records must be completely described, even if only part of the record is being accessed by the program. It has no automatic features but is a highly flexible language. It gives the programmer total control over the data and the results, but it also requires the programmer to exercise that control at all times.

COBOL is probably best mastered on the job, working with someone who already knows it. Manuals are supplied by the computer manufacturers, and their comprehensibility varies from system to system; but COBOL, because of its complexity, is not the type of language one can learn from a manual. COBOL is taught almost anywhere computer courses are available. There are also numerous texts available as well as an instructional version of the language, from which the fundamentals can be learned.

Fortran is a scientific language. It is the "number cruncher" *par excellance*. It is, however, very poor at handling character data; under normal operation, the maximum number of letters a character variable can contain is four. Fortran, like COBOL, has no automatic functions but does permit a great deal of flexibility and control. It is symbol oriented rather than language oriented, as expected of a scientific language. For example, compare the following two read statements:

```
COBOL   READ "infile" FROM "FD-infile record" AT
        END GO TO "branch name."
Fortran 10 READ ("n", 20, END="branch number")
        Prefix, Anum, Credit
        20 FORMAT (A4,F3.0,4X,F4.1)
```

COBOL provides a separate file description which ties the variable name to its field size, type, and location. Fortran supplies a READ statement which lists the variable names and a FORMAT statement which provides the field size and type. Fortran has fewer required statements than COBOL, making it both more concise and somewhat easier to learn initially, but because of its symbolic structure, it is more complex to follow. COBOL is somewhat self documenting: variable names can be up to 40 characters long, so if descriptive names are used, it is fairly easy to follow the program logic without having separate explanatory statements. Fortran, in contrast, must be completely documented or programs become incomprehensible within a short time, even to their creators. Courses in Fortran are probably even more readily available than courses in COBOL, and a teaching version exists also.

Table 1
Features of Several Computer Languages

Features	Languages					
	COBOL	Fortran	MarkIV	Easytrieve	SPSS	SAS
Training						
Has language-oriented syntax as contrasted to symbolic	+2	−2	−2	+2	−1	+2
Provides well-written manual	0	0	−1	+1	+2	+2
Has training programs available	+2	+2	+2	+1	+1	+1
Is forgiving	−2	−1	−2	+2	+1	+1
Requires minimum of coding	−2	−1	+1	+2	+2	+2
Sends understandable error messages	−2	−2	+2	+2	+1	+2
Groups job control language	+2	−2	−2	−1	+2	+2
Applications						
Handles variable length files	+2	−1	+2	+1	0	+2
Matches, merges, adds variables, updates	+2	−1	+1	+1	−1	+2
Extracts small data sets	+2	+2	−2	−1	+2	+2
Has automatic functions (read, write, move, sum, sort, etc.)	0	0	+1	+2	+2	+1
Has understandable selection logic	+2	−1	−1	−1	+1	+1
Has report formatting	0	0	+2	+2	+2	+2
Has format overrides	0	0	+1	+1	+1	−1
Handles character data	+2	−2	+2	+2	−1	+2
Provides descriptive statistics (mean, min, etc.)	−2	−2	+2	−1	+2	+2
Provides inferential statistics (ANOVA, Regression, etc.)	0	0	0	0	+2	+2
Machine						
Is machine independent	+1	+2	+1	0	+1	0

If Fortran and COBOL are the only two languages available, COBOL is probably the best choice for institutional research, primarily because it handles character data and report writing better than Fortran does.

Several programming aids have been developed recently which are reported to make COBOL coding much more efficient. COBOL, then, would be a fairly good choice for institutional research reports if these aids are available to the institutional researcher.

IBM's PL/1 is reputed to have many of the advantages of both COBOL and Fortran, being language oriented but still being good for scientific calculations and having report capabilities also.

Report Languages

MarkIV—a file maintenance and report language—is a proprietary language created by Informatics, Inc. It comes in three progressively larger versions which operate on various models of IBM-type, Univac, and Siemens computers (Informatics, Inc., 1977). It produces simple reports very easily, but even moderately complex applications are quite difficult. The natural environment for MarkIV is one in which there is a high level of central support to create and maintain the catalog of files, to supply the complex job control language, and to guide the casual user through the somewhat obtuse symbolic syntax and selection logic.

The user's manual for MarkIV is in four volumes and assumes a high level of sophistication. Informatics has released an index which may alleviate some of the problems associated with locating information in the manual (Informatics, Inc., 1979). The language is well supported with training programs which use well-prepared lesson materials, all geared to business applications, not to higher education. There is a free-form special feature which takes more or less freely keyed information and pre-edits it for input to the regular MarkIV processor. This adds an additional complication to a language which is already complicated enough.

MarkIV handles files very well, especially those it creates, but difficulties have been experienced by programmers using files not created by MarkIV if key fields contain invalid data or if the file is out of sequence. The language is designed to take advantage of the space savings permitted by variable fields.

MarkIV lends itself to production applications which allow users a certain amount of flexibility in specifying the contents and order of a report. MarkIV permits users to specify reports without having to concern themselves with the details of file handling and description when a high level of central support for file and catalog maintenance is available. It is capable of providing—quite economically—numerous reports simultaneously from the same file systems. It is not as economical to run MarkIV reports one at a time.

It is worthwhile for an institutional research office at an institution which already supports MarkIV to take advantage of the report-writing flexibility it provides. It is too time consuming, however, for the office to set up and maintain its own MarkIV library. This is probably true also of any of the database or quasi-database languages currently available.

Easytrieve, in contrast, requires little or no central support. This is a proprietary report language from Pansophic, Inc. which operates only on a large computer such as an IBM 360, 370, or an equivalent. It can be learned from the manual, is language oriented, and its free input form makes for easy terminal use. Easytrieve handles data in any form—numeric, character, binary, packed, or other. Only the fields actually required for the program need be defined to Easytrieve. It permits cataloging of file descriptions for future use, but it does not require such a catalog. Both of these features, plus automatic reading, writing, sorting, and totaling, help keep coding to a minimum.

Easytrieve's file extraction ability is somewhat awkward and, while it does read variable records, getting it to sort on a field from the variable portion of a record is a complicated task. Its file handling strengths are its file matching and table search capabilities. It also automatically writes out the complete input record without the programmer having to specify output format, a feature which amounts to totally automatic movement of modified input to output data, a real advantage for file updating. Easytrieve automatically totals numeric data, writes the totals in the same column as the detail information, and, in addition, permits computations to be performed on these control totals. It has a full set of automatic report format functions with fair ability to override. Its main report limitation is that it won't normally "go around corners" and continue output on the next line. When the print line overflows, it comes to a halt until columns are closed with a space override or a field is removed. It can be instructed to write multiple-line reports, but the data then must be all character. The current version of Easytrieve requires job control language (JCL) between each job step in a multi-step program, but a new version has just been released which allows numerous job steps to be strung together without inserting JCL at every step. Easytrieve is an excellent sort-accumulate-list report writer, and it makes economical use of computer time.

None of the languages discussed has extensive statistical features. Easytrieve does sums automatically, MarkIV does sums and averages and provides minimums and maximums, but none of them does inferential statistics automatically. One must turn to one of the statistical packages for these features.

Statistical Languages

The Statistical Package for the Social Sciences (SPSS), from SPSS, Inc., is perhaps the best known of the statistical languages. It is relatively machine independent and will operate on the IBM 360, 370, or larger machines (or equivalent) or on the CDC6000, CYBER 70, Univac 1100, Xerox, and Burroughs B6000 or B7000 (Nie, Hull, Jenkins, Steinbrenner, and Bent, 1975). It is based on the Fortran language and shares Fortan's weakness in dealing with character data, although recent modifications have done much to alleviate this problem. The new SPSS report feature (Hull, 1979) allows concatenation of character fields so that names can be listed properly. The new report feature also provides many desirable report format features including automatic totals, means, and other descriptive statistics, and in addition, it allows calculations with the summary variables. The SPSS manual is one of the best written manuals available, explaining clearly how to use the language and including adequate examples. It is also a fairly good statistical reference work. SPSS offers a complete collection of inferential statistics including various types of correlation and regression, analysis of variance, discriminant analysis, factor analysis, canonical correlation, and in the latest version, a full array of nonparametric statistics.

This new version also offers expanded data-handling capabilities, reading packed data, zoned decimal data, and double precision numeric data. A major file-handling weakness of SPSS, for institutional research, is that it reads only fixed length files. However, it does read multiple files, merge data, and add variables as well as create extract files and write out statistically derived variables such as correlation matrices. It also permits cataloging of SPSS data sets and procedures for future use.

SPSS, like Fortran, is a symbolically oriented language, and the record descriptions and assignment of variable names occur normally in two different program statements. The fixed format source code of SPSS is not as easy a form for terminal use as is a free format input. SPSS has just recently become a much more useful language for institutional research with the introduction of its new report writer and extended data capabilities. It will not, however, process data on variable length files.

Another, newer, statistical language is the Statistical Analysis System (SAS), marketed by SAS Institute, Inc. SAS is only available for the IBM 360 or 370 or larger machines or equivalents. There are a number of manuals for SAS, from the primer for beginners (Helwig, 1978) to the programmer's man-

ual for the expert (Helwig & Reinhardt, 1979). It sometimes seems that the more flexible a language is, the more complex it is to learn, and although SAS can be learned from the manual, it does take time to tie the pieces together. Unlike SPSS, SAS gives very little detail about the statistical principles behind its procedures, but it does provide references to texts from which the algorithms were taken.

SAS is excellent for file manipulation. It reads both fixed and variable length files and does extraction of data fairly well. It processes all types of data format and has a number of different ways to specify file description. It is suggested, however, that a user settle on one style, to avoid confusion. SAS will read multiple files in a single job and will sort, match, merge, add new variables, update, and subset very easily. An automatic table look-up is about the only feature it lacks.

SAS has a completely automatic report feature, PROC PRINT, which automatically compresses data columns as far as possible and starts a second print line if data still overflows a single line. There is a second report option which permits complete specification of the report format using PUT statements.

SAS accesses the IBM operating system and provides central processing unit (CPU) time data for procedures as well as record counts and label information for data sets. The language has a fairly complete array of descriptive and inferential statistics. It has fewer nonparametric functions than the new SPSS but has some parametric options which SPSS lacks, especially for exploratory regression analysis. In contrast, SPSS shows how to handle dummy variable coding for regression, while SAS does not. Both languages allow interfaces with other statistical packages. SAS will interface with BMDF, OSIRIS, SPSS, and DATA-TEXT and will use statistical procedures from those packages instead of using its own, while retaining its own data structure; SPSS interfaces with OSIRIS.

The SAS Institute sponsors an active user organization which holds conferences and workshops and which publishes a newsletter called "SAS Communications." Users are given an opportunity, in an annual survey, to vote on the functions they want developed in new versions.

SAS is probably the better of the two statistical languages, for institutional research, because of its file-handling ability and its free-form, language-oriented syntax. However, its utility is somewhat limited since it is available only on large, IBM-type computers.

Conclusion

A statistical language which provides report features is probably the best language choice for an institutional research office which does both straight reports and inferential statistics. SAS, if it is available, is probably the more useful of the two described in this paper, mainly because of the ease with which it manipulates all types of files and its free-form, language-oriented coding format which lends itself to terminal operation.

If the office does not get involved in inferential statistics, Easytrieve or an equivalent, straight report language is preferable to either the file management languages, such as MarkIV, or the COBOL-Fortan-PL/1-type of programmer language.

An institutional research office which does some of its own programming should keep in mind the following:

- Programs that produce reports should be documented clearly in English. When the staff member who wrote the program leaves, it is almost impossible for another staff member to determine from the program code exactly what went into the report.
- There should be only one or two languages used in the office, and more than one staff member should know each one used.
- A new staff member, who already knows another language, should learn the "official" language or languages rather than introducing a new one. Training time may be increased, but it will be well worth the extra effort to keep institutional research programs consistent and reusable under changing circumstances.
- Existing official data files should be used whenever possible, rather than separate ones being created or maintained. A "freeze" schedule may have to be arranged to pick off data for storage from live data files, but the temptation to create "your" data should be avoided; "your" data can become inconsistent with "their" data very quickly. If frozen files are used, the whole file in its original format should be obtained to avoid having to rewrite the data access program when changes to the file are made.

Learning a programming language can be a time-consuming and sometimes frustrating task, but once the language becomes familiar, it extends the capability of the institutional researcher to provide meaningful analysis by removing much of the burden of routine clerical detail. It is possible to see the forest when it is no longer necessary to count by hand all the rings on all the trees.

References

Helwig, J. T. *SAS introductory guide*. Raleigh, N.C.: SAS Institute, Inc., 1978.

Helwig, J. T. & Reinhardt, P. S. (Eds.). *SAS programmer's guide*. Raleigh, N.C.: SAS Institute, Inc., 1979.

Hull, H. C. & Nie, N. H. *SPSS update: New procedures and facilities for release 7 & 8*. New York: McGraw-Hill, 1979.

Informatics, Inc. *MarkIV systems: IBM OS operations guide* (2nd ed.). Canoga Park, CA: Author, 1977.

Informatics, Inc. *MarkIV systems: Index*. Canoga Park, CA: Author, 1979.

Nie, N. H., Hull, H. C., Jenkins, J. G., Steinbrenner, K., & Bent, D. H. *Statistical package for the social sciences* (2nd ed.). New York: McGraw Hill, 1975.

Ryland, J. N. Proprietary software and the institutional researcher. In E. M. Staman (Ed.) Examining new trends in administrative computing, *New directions for institutional research*. San Francisco: Jossey-Bass Inc., 1979, pp. 37–40.

Bibliography

Cress, P., Dirksen, P., & Graham, J. W. *Fortran IV with WATFOR and WATFIV*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1970

Helwig, J. T. & Council, K. A. (Eds.). *SAS user's guide*. Raleigh, N.C.: SAS Institute, Inc., 1979.

Informatics, Inc. *MarkIV Systems: Reference manual* (2nd ed.). Canoga Park, CA: Author, 1977.

Informatics, Inc. *MarkIV systems: Special features*. Canoga Park, CA: Author.

Informatics, Inc. *MarkIV systems: Technical information bulletin no. 6.1*. Canoga Park, CA: Author, 1977.

International Business Machines Corporation. *IBM vs. COBOL for OS/VS* (2nd ed.) (IBM Systems Reference Library). New York, Author, 1978.

Panasophic, Inc. *Easytrieve reference manual*. Oak Brook, IL: Author, 1978.

Woolley, G. *Contemporary COBOL*. San Francisco: Rinehart Press, 1971.

The *AIR Professional File* is published by the Association for Institutional Research, 314 Stone Building, Florida State University, Tallahassee, FL 32306. Distributed as an insert to the *AIR Newsletter*, the *Professional File* is intended as a presentation of papers which synthesize and interpret issues, operations, and research of interest in the field of institutional research. Paper authors are responsible for material presented. The editor is Richard R. Perry, Associate Vice President for Academic Affairs, The University of Toledo, 2801 W. Bancroft, Toledo, OH 43606.