Programming Languages:
A Short History for Economists

by

Marc Nerlove

WP 03-17

Department of Agricultural and Resource Economics

The University of Maryland, College Park

Prepared for a special issue of *The Journal of Economic and Social Measurement*, to appear.

REVISED 09/20/03   ©Marc Nerlove 2003

Programming Languages: A Short History for Economists[1]

Marc Nerlove
Department of Agricultural and Resource Economics
University of Maryland
Tel: (301) 405-1388   Fax: (301) 314-9032
e-mail: mnerlove@arec.umd.edu
http://www.arec.umd.edu/mnerlove/mnerlove.htm

> The White Rabbit put on his spectacles. "Where
> shall I begin, please your Majesty?" he asked.
> "Begin at the beginning," the King said gravely,
> "and go on till you come to the end; then stop."
>   Lewis Carroll, *Alice's Adventures in Wonderland,* XII

The Encyclopedia Britannica [16] defines a computer programming language as "...any of various

languages for expressing a set of detailed instructions for a digital computer. Such a language consists of

characters and rules for combining them into symbols and words.

The definitive treatment of the history of programming languages is contained in the two volumes of conference proceedings: Wexelblat [45] and Bergin and Gibson [4]. A very good brief treatment from a general perspective may be found in a standard textbook on programming languages widely used in computer science departments Pratt and Zelkowitz [30]. The history of computing and of software is an area of active research; for example Ceruzzi [7] and Campbell-Kelly [5]. The very popular exposé, Cringely [9] on which a PBS miniseries, "Triumph of the Nerds" was based, is a delight to read. An amusing fictional account is the novel, Gibson and Sterling [17] .

Two important papers by Charles Renfro [31, 32] published in this issue deal more specifically with econometric software, supplement my discussion.

The history of the development of computer languages shows an evolution towards an ever closer approximation to natural or mathematical language. This history in the last half of the twentieth century is summarized in the table at the end of this paper.

Computers only recognize yes-no or 0-1 or off-on internally. Von Neumann's great contribution to computation was the idea that a machine could be programmed to follow a sequence of instructions stored in its memory as off-on's. In 1946, at the University of Pennsylvania's ENIAC, the switches were really set by hand.[2] In the early days, *circa* 1956, I vividly recall Hy Weingarten of our group at the USDA actually writing simple programs in *machine language*, i.e., binary bits, which was equivalent to setting these internal switches "by hand." This clumsy and time-consuming method was quickly supplanted by the first assembler programs which established a one-to-one correspondence between machine code and easily remembered "words."

At first, programmers, Hy among them , would first write out what he wanted the machine to do in semi-symbolic form, *assembly language,* and then laboriously translate these instructions into appropriate patterns of bits (binary numbers) to be stored in the machine's memory, used by the machine actually to execute Hy's program. Of course, it was pretty silly to have a human being do the laborious translation. What was needed was a standard symbolic representation of the algorithms to be implemented so that the machine itself could do the grunt work of translation. These programs are called *assemblers.* What an assembler does is to translate each of the symbols in the assembly language program into machine language symbol of binary bits. By collecting all of these machine language symbols, the assembler produces a machine language program that sets the switches as the programmer intended. Hy could now write statements like "Add A B" instead of "101101010011101001011010" in order to add two numbers.

Assembly languages are direct word for word translations of machine language. For each machine language instruction, there is a symbol that indicates what it is used for; and for each storage location in memory, there is a symbol that tells what it is meant to contain. After some experimentation with assemblers, it became apparent that computer programs could do much more than just translate word for word; it is also possible for the translation program to rearrange the words so that the syntax of the language is different from the syntax of the machine language. The ability to change the syntax

---

[2] EDSAC at the University of Cambridge, constructed under the direction of Maurice Wilkes, came along about the same time, as did the Mark 1 at Manchester, Davis [11, p. 194].

of the language lets the programmer write statements like a = 1 + 1, rather than a sequence of instructions

to the hardware, that will store one in a memory location, add that location to itself, then store the resulting

value in a new location. Thus were invented the first *programming languages:* FORTRAN in 1957 by

IBM,[3] and ALGOL in 1958 by a European consortium,[4] COBOL (Common Business OrientedLanguage)

in 1960,[5] and LISP, in 1958, for basic list processing, by John McCarthy, then at MIT.[6] LISP, however, is

a rather different sort of creature and I will discuss it more thoroughly below. These languages are known

in the trade as high-level languages (HLL), that is, a programming language which provides some level of

abstraction above assembly language. They normally use statements consisting of English-like keywords

such as "FOR", "PRINT", or "GOTO", where each statement can correspond to several machine language

instructions. It is much easier to program in a high-level language than in assembly language though the

efficiency of execution depends on how good the compiler or interpreter is at optimizing the program.

---

[3] *FORmula TRANSlation,* John Backus, IBM, 1954-57.
[4] ALGOL (ALGOrithmic Language) is one of several high level languages designed specifically for programming scientific computations. It started out in the late 1950's, first formalized in a report titled ALGOL 58, and then progressed through reports ALGOL 60, and ALGOL 68. It was designed by an international committee to be a universal language. Their original conference, which took place in Zurich, was one of the first formal attempts to address the issue of software portability. ALGOL's machine independence permitted the designers to be more creative, but it made implementation much more difficult. Although ALGOL never reached the level of commercial popularity of FORTRAN and COBOL, it is considered the most important language of its era in terms of its influence on later language development. ALGOL's lexical and syntactic structures became so popular that virtually all languages designed since have been referred to as "ALGOL - like"; that is, they have a hierarchical in structure with nesting of both environments and control structures. See Backus [1, 2], Baumann, Feliciano, Samelson [3], Naur, Backus, Bauer, Green, Katz, McCarthy, Perlis, Rutishauer, Samelson, Vauquois, Wegstein, van Wijngaarden, and Woodger [28a] and many subsequent papers. The papers by Naur and Perlis in Wexelblatt [45, pp. 75-139] give a very comprehensive discussion of ALGOL's history on both sides of the Atlantic. Although ALGOL was implemented on a new powerful Boroughs machine in about 1960 (IBM wouldn't have anything to do with it), it was originally intended as a language for the expression and communication of *algorithms* in general; at one time, *Communications of the Association for Computing Machinery* published algorithms in ALGOL.
[5] "Grace Hopper led a group at Univac to develop FLOWMATIC in 1955. The goal was to develop business applications using a form of English like text. In 1959, the U.S. Department of Defense sponsored a meeting to develop Common Business Language (CBL), which would be a business-oriented language that used English as much as possible for its notation. Because of divergent activities from many companies, a Short Range Committee was formed to quickly develop this language. Although they thought they were designing an interim language, the specifications, published in 1960, were the designs for COBOL (Common Business Oriented Language). COBOL was revised in 1961 and 1962, standardized in 1968, and revised again in 1974 and 1984." Pratt and Zelkowitz [30, p. 6].
[6] The original version was LISP 1, invented (some historians prefer the use of the word "discovered") by John McCarthy at MIT in the late 1950s. McCarthy [27, 28]. LISP is actually older than any other high level language still in use except FORTRAN, and has undergone considerable change over the years. Modern variants are quite different in detail. See below.

There is an important distinction between compiling a program and interpreting it.[7] But, in the end the program has to be expressed in a manner the machine can understand. Whereas assembly languages let programmers express things directly in machine terms in a different way than in terms of binary bits, they still tell the machine what is to be put in memory locations and what is to be done with the contents of these memory locations. The actions that can be expressed are closely tied to the actions the machine can do, such as, putting the contents of one memory location into a register and adding that number to another in another location. HLLs bring us much closer to natural language.

At about this time, many of the early econometric programs were written. These were not general purpose programs designed to do a variety of statistical and econometric tasks but were rather limited, and were, moreover, machine-specific. In following the history of programming languages, one has to distinguish among several different types of software although the boundaries between them are often blurred. On the one hand, there are true programming languages which are used for many purposes in communicating with the machine, beginning with the first versions of FORTRAN, ALGOL, etc., and ranging all the way in recent times to FORTRAN90, C, C^{++}, and Java. The most recent versions of Mathematica, while emphasizing doing computer assisted mathematics and graphics, nonetheless aim for this level of general use. At the next level, one finds somewhat greater specificity in purpose and function. Examples of particular interest to economists include S, S^{+}, R, GAUSS, MATLAB and MAPLE. Modern versions of TSP, SAS, SPSS, LIMDEP and STATA, among others, come close to falling in this category. Then, there are what I would call "libraries" or collections of special purpose algorithms designed to be used within programs written in one of the major languages, e.g., the LINPACK, MINPACK and EISPAC, IMSL Libraries of Visual Numerics and the NAG Fortran Library Subroutines of the Numerical Analysis Group, first developed in the 1960s. There are, in addition, collections of programs, sometimes called packages, linked together and designed for special purposes. These generally must be used within some of the languages one step removed from the basics. Examples include MATLAB Toolboxes many of which are written by third parties. Finally, there are programs which although made generally available cannot be used stand-alone in any sense. Soon after the development of FORTRAN and ALGOL, econometricians

---

[7] Compilers transform an entire program from one language to another (e.g., the assembly language) for subsequent execution; interpreters execute a program sequentially, translating at each step. Compiled programs almost always run faster than interpreted programs but are a lot harder to debug.

and statisticians began writing and making available to colleagues programs for doing basic statistical analyses such as regression. Special mention should be made of the regression program written by Stroud [42] at the University of Wisconsin, the programs by Zellner, Stroud and Chau [46 , 47] for computing seemingly unrelated regressions estimates and two and three stage least squares, and a program by Eisenpress [14] for limited-information maximum likelihood. Regression programs became common. Five OLS regression programs dating from that era were evaluated in Longley [26]. In this period, the once widely used package TROLL (Timeshared Reactive OnLine Laboratory) was developed at MIT under the guidance of Ed Kuh by Mark Eisner [13], although the first readily available version wasn't released until 1972 (see Renfro [31]). Mitch Kapor, later of Lotus 1-2-3 fame, developed a version for the early Apple PC called Tiny TROLL.

IBM developed PL/I, a precursor of C, in the 1960's.[8] PL/I (Programming Language One) was an attempt to combine the best features of FORTRAN, COBOL and ALGOL60 was developed by George Radin of IBM in 1964. PL/1 provided the foundation for development of the SAS system on IBM operating systems. Later SAS was rewritten in C so as to be portable to other systems.

The influence of LISP (John McCarthy [27, 28]) on all further developments in the evolution of programming languages has been fundamental and pervasive. LISP and its variants have been very important in the development of data base management and word processing programs -- and in the development of still higher level HLLs. WORD, WordPerfect, and LaTex, in many respects have their origin here. In his essay on the history of LISP, Herbert Stoyan [40], writes:

"LISP is understood as the model of a functional programming language today. There are people who believe that there once was a clean 'pure' language design in the functional direction which was comprised by AI(Artificial Intelligence)-programmers in search of efficiency. This view does not take into account, that around the end of the fifties, nobody, including McCarthy himself, seriously based his

---

[8] With the introduction of its new 360 line of computers in 1963, IBM developed NPL (New Programming Language) at its Hursley Laboratory in England. After some complaints by the English National Physical Laboratory, the name was changed to MPPL (Multi-Purpose Programming Language), which was then shortened to just PL/I. PL/I merged the numerical attributes of FORTRAN with the business programming features of COBOL and the syntax of ALGOL. PL/I achieved modest success in the 1970s, but its use today is dwindling as it is replaced by C, C++ and Ada. The Cornell University educational subset PL/C achieved modest success in the 1970s as a student PL/I compiler.

programming on the concept of mathematical function. It is quite certain that McCarthy for a long time associated programming with the design of stepwise executed 'algorithms'.

"On the other side, it was McCarthy who, as the first, seemed to have developed the idea of using functional terms (in the form of "function calls" or "subroutine calls") for every partial step of a program. This idea emerged more as a stylistic decision, proved to be sound and became the basis for a proper way of programming - functional programming (or, as I prefer to call it, function-oriented programming). We should mention here that McCarthy at the same time conceived the idea of logic-oriented programming, that is, the idea of using logical formulae to express goals that a program should try to establish and of using the prover as programming language interpreter.[9]

For my purposes in this paper, however, a more important related development was the development of FORMAC (FORmula MAnipulation Compiler) at IBM in the early 1960's (Sammet and Bond [34]). This was the first step in extending the use of computers to do formal mathematics, as distinct from numerical mathematics. MAPLE and MATHEMATICA, which are discussed here, are the "children" of FORMAC. But all modern HLLs, whether designed primarily for numerical or for symbolic manipulation, incorporate list processing elements and the ideas of John McCarthy.

At this point in the history of programming languages, the ideas behind the development of ALGOL came to dominate: algorithmic and information structures. Algol68 was released in 1968, after having been under active development by a group associated with A. van Wijngaarden in the period 1965-68. Intel was founded by Robert Noyce, Andrew Grove and Gordon Moore in 1967. Donald Knuth's multivolume landmark [24] began publication in 1968.[10] The now widely-used "econometric software packages," e.g., SAS, SPSS and TSP, were created about this time, and have since grown into major

---

[9] For a an explanation "...in the simplest possible terms [of] what McCarthy discovered...," see Graham [19]. But this paper is not so simple for one who does not already know a lot about computer programming! There is some controversy, however, about where exactly the concept of subroutine originated. See Giloi [18], who cites the early work of Konrad Zuse [48] in 1943-45, published only in 1972, and see Graham [19], also Sebasta [35, p.55]. I suspect that all this is an example of Stigler's Law of Eponymy, [37, pp. 277 - 290]. Whoever originated the idea, it is McCarthy's influence that has been pervasive.

[10] And is projected to run to seven volumes, of which only three have so far appeared: 1. *Fundamental Algorithms,* 1968; 2. *Seminumerical Algorihtms,* 1969; 3. *Sorting and Searching,* 1973. Volumes 1 and 2 have gone into third editions (1997). Volume 3 is now in its second edition (1998). According to Knuth's homepage, Volume 4 is in "beta test" version.

commercial enterprises.[11] BASIC (Beginner's All-purpose Symbolic Instruction Code) was designed by

John G. Kemeny and Thomas E. Kurtz at Dartmouth College in 1963. It first ran on an IBM 704 in 1964

and was designed for quick and easy programming by students and beginners. BASIC exists in many

dialects, and is popular now on microcomputers with sound and graphics support.[12]

  At about this time (the mid-1960's), a number of groups began to develop libraries of computer

routines, which could be incorporated in other programs and which were efficiently written from a

numerical point of view. By the early 1970's, EISPAC, LINPACK and MINPACK by Argonne National

Laboratories, in the public domain, and the IMSL Libraries of Visual Numerics and the NAG Fortran

Library Subroutines of the Numerical Analysis Group, which are proprietary, were available.[13] These

collections of standard numerical analysis results, written in FORTRAN or in C, but have to be put together

within still higher level programming languages in order to be useful to statisticians and econometricians.

---

[11] SAS software was originally written as a by-product of a university project to analyze agricultural data at North Carolina State University in the early 1970's. The first references to SAS appear in 1972. SAS Institute Inc. was formed in 1975. SPSS Inc. was founded in 1968. The development of TSP is described in some detail by Brown Hall in a recent e-mail: "The original TSP was developed in 1965/66 by Robert Hall with the help of several other graduate students at MIT (Robert Gordon, Charles Bischoff, Richard Sutch, etc). Bob took a version to Data Resources Inc in 1969/70 and it became the basis for their first interactive econometric program. I did not really do any development for them, although I worked on a different program (capital budgeting) as a consultant briefly. The current TSP is based on a version I found in the basement of Littauer in 1970 (on cards) which was sent by Dale Jorgenson from Berkeley to Harvard when he moved. It was developed and distributed by me at Harvard until 1977 and became a commercial product in 1977 when we moved to Stanford (basically because of customer demand for better support and service and the need to finance it). Thus although the official name of the company was adopted in 1982, the actual start date was 1977."

[12] Visual Basic is a derivative peculiar to Microsoft Windows applications. The role of BASIC in the origin of Microsoft (and Bill Gates' rise to fame and fortune) is described by Campbell-Kelly [5, pp. 204-205].

[13] These originated in the work of the NATS Project (National Activity for Testing Software), sponsored by the National Science Foundation in the 1960's, and are precursors of BLAS (Basic Linear Algebra Subprograms), a collection of high quality "building block" routines for performing basic vector and matrix operations. Level 1 BLAS do vector-vector operations, Level 2 BLAS do matrix-vector operations, and Level 3 BLAS do matrix-matrix operations. Because the BLAS are efficient, portable, and widely available, they're commonly used in the development of other high quality linear algebra software. Lawson, Kincaid, Krogh [25]. A modern version is LAPACK, which is written in Fortran77 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision. http://www.netlib.org/lapack/index.html , accessed 06/17/03 .

These subroutines are the building blocks of GAUSS and MATLAB, and many other still higher level packages for doing statistical and econometric calculations, such as, *inter alia,* SAS, TSP, and LIMDEP.[14]

PASCAL, after the French mathematician Blaise Pascal (1623-1662) was a programming language designed by Niklaus Wirth around 1970. PASCAL was designed for simplicity and for teaching programming in reaction to the complexity of ALGOL 68. It emphasizes structured programming constructs, data structures and strong typing. PASCAL has been extremely influential in programming language design and has a great number of variants and descendants.

The next major development in the evolution of computer languages was C, a programming language designed by Dennis Ritchie at AT&T Bell Labs ca. 1972 for systems programming on the PDP-11 and immediately implemented in the Unix operating system.[15] C^{++} is an improved version designed by Bjarne Stroustrup [41].[16] C is terse, low-level and permissive. Unix is written in C. Partly due to its distribution with Unix, C became immensely popular outside Bell Labs after about 1980 and is now the dominant language in systems and personal computer applications programming. It has grown popular because of its simplicity, efficiency, and flexibility. C programs are easily adapted to new environments. C has been acerbically described as "a language that combines all the elegance and power of assembly language with all the readability and maintainability of assembly language." A modern variant is C^{++}, which has become since its introduction, the language of choice for many programming applications such

---

[14] According to Charles Renfro [32], LIMDEP began in 1974 at the University of Wisconsin as an implementation of a RAND Corporation report by M. Nerlove and S. J. Press on multivariate loglinear and logistic models for the analysis of categorical data. RATS was also launched about this time as well.

[15] Subsequently described in Kernighan and Ritchie [23].

[16] Opinions, however, are not uniformly laudatory. One critic describes C^{++} as " a huge, bloated, hack-ridden monster." C++ is a fairly complicated object-oriented language derived from C. The syntax of C++ is a lot like C, with various extensions and extra keywords needed to support classes, inheritance and other object-oriented features. C++ was originally developed as an extension to C, but quickly evolved into a separate language.

Object-oriented languages define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an *object* that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can *inherit* characteristics from other objects. One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

The original object-oriented language, Smalltalk, was developed by Alan Kay in 1973. Modern variants include Java as well as C^{++}, Dictionary of Programming Languages [12].

as MATLAB and GAUSS.[17] Cribari-Neto [1999] recommends that all serious econometricians do at least some C or C^{++}. My own view, is that it's not a bad idea to do at least some C^{++} programming because all of the third-level programming languages may be slow for repetitive calculations such as are common in bootstrapping or estimation by simulation.[18] On the other hand, in the last few years many of these third-level languages have been greatly improved with respect to speed and numerical accuracy, so that for many econometric purposes resort to C, C^{++}, or FORTAN may no longer be necessary. Moreover, MATLAB, GAUSS, and MATHEMATICA now support the use of C and C^{++} for user extensions, that is C or C^{++} programs may be called directly from programs written in these three higher level languages.

Throughout the 1970's there was intensive development of "packages" of all sorts, especially for statistical and econometric analysis. In a recent paper, Charles Renfro [32] gives a thorough summary of these developments.[19] Many of these packages have since fallen by the wayside, but a large number still have their loyal adherents. As mentioned above, SAS and TSP among others were largely developed in this period. Graphics got off to a slow start during this period but gained momentum in the next decade with the development of Postscript by Adobe Systems in 1982, the release of the first version of WINDOWS (1984), and by the release of the Adobe Illustrator (1985) and Video Graphics Array in 1987. Improvements to FORTRAN (FORTRAN77), to PASCAL, and to BASIC (first introduced in 1964) were made during the 1970s. But the most momentous development, with the most far-reaching consequences, was the introduction of the first personal computers by Apple in 1977, followed closely by the first spreadsheet program, developed by Dan Bricklin, Dan Fylstra and Bob Frankston in 1978 and launched as VisiCalc in 1979, and by DOS (Disk Operating System), developed sometime before 1978. Spreadsheets and an easy to implement system for handling files turned the personal computer into a real tool from what had been basically a toy.[20] Mention should also be made here of S, a high-level procedural language

---

[17] These languages themselves are largely written in FORTRAN, C and C^{++}.

[18] Speed comparisons do not bear out MATLAB's lack of speed in comparison to GAUSS; while MATLAB is slightly slower for many calculations, it is faster for some. The newest versions (6+) represent a significant improvement in this respect over the versions evaluated by Cribari-Neto [8], and by Küsters and Steffen [24a].

[19] In a paper also in this issue, Houston Stokes [38] describes his development of B34S which evolved from an LS regression package beginning in about 1968.

[20] Ceruzzi [7], Chapter 7, "The Personal Computer, 1972-1977," pp.207-242, especially pp. 236-237; Campbell-Kelly [5] "Early Development of the Personal Computer Software Industry, 1975-1983," pp. 201-230. The early history of DOS is somewhat convoluted: IBM had something called DOS for its mainframe computers before 1971, Ceruzzi [7, 237]. The Altair 8800, manufactured by Micro

designed and used for statistics, numerical modeling, data analysis, and simulation, designed to provide

powerful interactive statistics and data analysis. S was developed during the period 1974-78 by a group

which included R. A. Becker and John Chambers at what was then AT&T Bell Labs.[21]

The 1980s were a period of extremely rapid development and innovation. C^{++}, mentioned already,

was developed in this period by Bjarne Stroustrup [41]. In order to compete with Apple, IBM introduced

its first PCs in 1981, and this development was followed the next year by numerous clones. [22]   In the same

year Osborne introduced the first truly portable PC[23], i.e., Notebook, Microsoft released the first

FORTRAN for DOS, and WordPerfect was first introduced. The following year, 1983, Microsoft WORD

and Lotus 1-2-3 were released, followed in 1984 by EXCEL and WINDOWS 1.0 and the first networking

software. In 1985, Adobe released Illustrator, and the Video Graphics Array (VGA) was introduced in

1987. Fast laser printers became common in the 1980s; color inkjet printers and MicroSoft PowerPoint

were introduced in 1988. For the purposes of this essay, however, the most significant development of the

1980s was the almost simultaneous founding of the companies that market GAUSS, MAPLE, MATLAB

and MATHEMATICA: Aptech was founded by Sam Jones in 1983 and the first version of GAUSS was

released in mid-1984. The MathWorks was founded by Jack Little and Cleve Moler in 1984 and the first

version of MATLAB released that year. Although MAPLE was the product of a decade-long research

---

Instrumentation Telemetry Systems (MITS), was not the first microprocessor-based computer, but it was
enormously influential. It was sold in kit form to computer hobbyists. Its appearance, according to Ceruzzi,
on the January 1975 cover of *Popular Science* is "perhaps the best-known event in the folk history of the
personal computer. There were imitators of course, but more importantly there were bright "kids" like Bill
Gates, Paul Allen and Gary Kildall who developed software for these machines. But the defining event was
the launch of Apple II in April 1977. It was this machine for which Visicalc was created by Dan
Bricklin, Dan Fylstra, and Bob Frankston and released in the fall of 1979. The "'electronic spreadsheet" it
created transformed the perception of the PC as a hobbyist's dream to a serious business machine.
[21] The successor to AT&T Bell Labs, Lucent Technology has not maintained the language, but it is
available commercially as S-PLUS and as a freeware version (http://www.r-project.org/). Both are very
widely used today in statistics applications. There are several useful books: Venables and Ripley [43, 44] ;
Dalgaard [10].
[22] Cringely [9] says that these clones were made possible by the technique of "reverse engineering" which
IBM had used to produce a non-patent-infringing version of the Apple using components available off the
shelf, but such a characterization appears nowhere else that I have been able to find. However, Kildall and
IBM did develop a key element crucial to the wide-spread use of the PC, which was spread by "reverse
engineering," the code they called BIOS (for Basic Input/Output System). This code, or modifications of it,
permitted essentially the operating system to run on many different clones. (Ceruzzi {7, pp. 238-239].)
[23] There were antecedents, of course: IBM created a machine in 1975, the Model 5100, with a proprietary
IBM cpu, which is sometimes described as the first portable, but it weighed about 50-60 pounds, so is
questionably described as "portable. The Model 5100 incorporated in one piece all the parts, including the
screen, the keyboard and the system unit. It is this one piece construction that suggests its portability. The
Osborne 1, released in 1981, although it weighed in at 24 pounds, could be described as the first
"Notebook."

project at the Department of Computer Science, University of Waterloo, Ontario and, later with the collaboration of a group at the Eidgenössische Technische Hochschule, Zürich , it was first released commercially in 1985. Stephen Wolfram founded Wolfram Research in 1986; the first version of MATHEMATICA was released in 1988. These programming languages are one step up from FORTRAN and C and make extensive use of the LINPACK, MINPACK and EISPAC, IMSL Libraries of Visual Numerics and the NAG Fortran Library Subroutines of the Numerical Analysis Group, first developed in the 1960s and discussed above. But, more importantly they are infinitely more flexible and powerful, than the statistical and econometric packages commonly in use. These languages have become the way to disseminate statistical and econometric research. Many mathematical, statistical, and econometric texts are now written making extensive use of one of these languages. The range and extent of these languages and packages, their differences and capabilities, are spelled out in some detail by the two papers by Renfro appearing in this issue.

It is interesting to speculate on the reasons for the efflorescence of languages and packages occurred in the 1980s why, in particular, "third-level" programming languages appeared almost simultaneously in the mid-1980s, followed then and through the 1990s by the development of a large number of econometric software packages. It seems to me that these developments are a classic case of Adam Smith's famous theorem that the division of labor is limited by the extent of the market, Stigler [36]. By the end of the 1970s, use of mainframe computers had become very expensive; for academic users at any rate; high priority and consequently rapid turn-around was virtually restricted to those supported by the AEC and Defense Department. The rest of us ran at low priority and usually at night. Because any original programs required, and still require, extensive debugging, many runs were usually required. While two or three runs might under the best of circumstances be possible at most university computer centers in the course of a night, only those with a low value of time were able to camp out at the computer center the requisite hours. It might take several weeks to debug even a relatively simple program. Such high time costs greatly reduced the incentive of many of us to learn programming skills and to do econometrics and statistics requiring more than extant statistical packages. The very success of packages like SAS and TSP, available at that time in mainframe versions, which do more or less everything but which produce huge volumes of output which have to be pored over, was more or less a response to the high value of time and

high time costs of "doing one's own thing." But the development of the PC and the rapid evolution of both

hard- and software technology for personal computers changed all this.[24] The dramatic fall in costs, both

pecuniary and more importantly time costs, greatly expanded the market for computation of all sorts and

opened a niche for satisfying the demands of those who wanted to do "hands on" mathematics,

econometrics and statistics, wanted more than the "hands off" computational power provided by packages

such as SAS and TSP, but did not want to engage in "serious" programming by writing their own in C^{++} or

FORTRAN. Successful packages did, of course, adapt to the PC in order to survive, but retain to this day

more than vestiges of their origins. Perhaps this accounts for the roughly simultaneous appearance of the

four programming languages considered above in the mid-1980s.

Finally to bring the story down to the present day, the decade of the 90s, just finished, has

witnessed continued technological improvements in hardware: greatly increased storage capacity and

accessibility and considerable increases in speed, but software development seems to have slowed as

compare with the previous decade. Recently, however, the third-level languages, in particular, MATLAB,

GAUSS, MAPLE and MATHEMATICA have begun to converge in capabilities. MATLAB and GAUSS

have added symbolic manipulation capabilities, while MAPLE and MATHEMATICA have added

enhanced numerical capacity and, more importantly, speed.[25] These languages in turn have spurred the

development of many specialized packages by third-party programmers. The reader need only to consult

the websites of the companies that provide these languages to discover the extent and variety of third-party

applications. The major change through the nineties has been the explosive growth of the internet, the

community of users all sharing programs and tips, and the associated development of object-oriented

programming languages such as JAVA (http://java.sun.com), which take full advantage of the network

---

[24] Of course, on-line use of mainframe computers through remote terminals had been possible since virtually the beginning. TSP, for example, was further developed as part of an on-line system for data retrieval and analysis via telephone to what was then an advanced Borroughs computer at DRI headquarters in Lexington, Massachusetts. But these systems were far too slow for interactive "programming," which became attractive only with the advent of the stand alone and relatively cheap personal computer. Many of the econometric software packages in use today, began as mainframe packages later "ported" to PCs or Macs, but widespread use is much more recent.

[25] Many computations can be best done by first expressing the result one wants analytically and then evaluating the analytical result numerically for some specific values of the arguments. This convergence is surely an example of the Hotelling principle in which hot dog vendors competing on a finite beach with a uniformly distributed demand converge toward the center of the beach. Hotelling [22]

infrastructure provided by the internet.[26] The implications of the profound changes in the economic

information structure which are taking place and are expected in the near future for the economics

profession are explored in a recent paper by Bill Goffe and Bob Parks [20]. Of major significance in the

present context is the availability, on line, of extensive machine-readable economic data bases and of

programs and add-ons supporting all four of the third-level programming languages discussed.[27] Several

journals, notably the *Journal of Applied Econometrics* and the *Journal of Economic and Business Statistics*

have established data archives for all of the articles they now publish, making replication of results possible

at long last.

Implicit in my account of the history of programming languages with particular relevance to

econometric research is the idea that there is a continuum of languages, ranging at one extreme from

machine and assembly language, through FORTRAN and C and the ISML and NAG libraries, through

third-level languages such as GAUSS, MATLAB, MAPLE and MATHEMATICA, to software such as

SAS, TSP or LIMDEP at the other extreme. There is another dimension as well, that is languages such as

MAPLE and MATHEMATICA emphasize symbolic manipulation, GAUSS and MATLAB emphasize

numerical calculation , although as indicated above they are converging. All four of these languages will do

econometrics and statistics, but some will be more efficient for our purposes than others for reasons related

to their location on this dimension of the spectrum. Specifically econometric software is generally designed

for a limited range of tasks, although there has always been a tendency for the developers to expand this

range and to incorporate more and more methods.

It is apparent that there has been a constant interplay between the development of programming

languages and econometric software. In turn, the availability of ever more powerful software and more

especially the increasing power and ease of accessibility of programming languages have enhanced the

capacity of econometricians to write programs for the solution of specific problems. As the availability of

a variety of data sets has increased, so has the range of econometric problems considered and methods

developed. It would not be fair, however, to say that the development of programming languages has been

influenced by econometric developments, but the opposite is surely true.

---

[26] Eliens [15] . C^{++} and FORTRAN90 both have some element of object-oriented programming, which basically assures portability independently of programming environment, Cary, *et al*, [6].

[27] A good source for accessing such resources is Bill Goffe's Resources for Economists on the Internet: http://wuecon.wustl.edu/cgi-bin/mfs/03/EconFAQ.html.

REFERENCES

1. Backus, J. W., "The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich   ACM-GRAMM Conference on Information Processing," UNESCO Paris, June 1959, 125-132

2. Backus, J. W., F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, M. Woodger, and P. Nauer, "Revised report on the algorithm language ALGOL 60," *Communications of the ACM, 6*: 1-17, January, 1963

3. Baumann, R., M. Feliciano, F. L. Bauer and K Samelson, *Introduction to ALGOL,* Englewood Cliffs: Prentice-Hall, 1964

4. Bergin, Thomas J. and Richard G. Gibson, eds., *History of Programming Languages - II,* New York Academic Press, 1996

5. Campbell-Kelly, Martin, *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry,* Cambridge: MIT Press, 2003

6. Cary, John R., Svetlana G. Shasharina, Julian C. Cummings, John V. W. Reynders and Paul J. Hinker, "Comparison of C++ and Fortran 90 for Object-Oriented Scientific Programming," Los Alamos National Laboratory, Report No. LA-UR-96-4064, 1996, *Computer Physics Communications, 105:* 20-36*,* September 1997

7. Ceruzzi, Paul E., *A History of Modern Computing,* 2nd ed., Cambridge, MIT Press, 2003

8. Cribari-Neto, F., "C for Econometricians," *Computational Economics*, 14, 135-149, 1999

9. Cringely, Robert X., *Accidental Empires* New York: Harper Information, 1996

10. Dalgaard, Peter, *Introductory Statistics with R,* New York: Springer, 2002

11. Davis, Martin, *The Universal Computer: The Road from Leibnitz to Turing*, New York: Norton, 2000

12. Dictionary of Programming Languages*,* 2000, http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl, accessed 06/14/03

13. Eisner, M., "TROLL/1 - an interactive computer system for econometric research," *Annals of Economic and Social Measurement, 1*: . 95-96, 1972

14. Eisenpress, H., *Forecasting by Generalized Regression Methods. Limited-Information Estimation Procedure IBM 704 Program IB LI.* New York: International Business Machines Corporation. Data Systems Division, 1959

15. Eliens, A., *Principles of Object Oriented-Software Development,* Reading, MA: Addison-Wesley, 1994

16. Encyclopædia Britannica, "Computer Programming Language," Retrieved June 12, 2003, from *Encyclopædia Britannica Online*, http://www.search.eb.com/eb/article?eu=25460

17. Gibson, W., and B. Sterling, *The Difference Engine,* New York: Bantam Books, 1992

18. Giloi, Wolfgang, K.: " Konrad Zuse's Plankalkül: The First High-Level 'non von Neumann' Programming Language" *IEEE Annals of the History of Computing, 19*: 17-24, 1997

19. Graham, Paul, "The Roots of LISP,"   http://www.paulgraham.com/rootsoflisp.html

20. Goffe, W. L., and R. P. Parks, "The Future Information Infrastructure in Economics, "1997,
            http://econwpa.wustl.edu/eprints/mic/papers/9704/9704001

21. Hopkins, T., and B. Horan, *Smalltalk: An Introduction to Application Development Using VisualWorks*,
            Prentice Hall, 1995

22. Hotelling, Harold, "Stability in Competition," *The Economic Journal*, 39: 41-57, 1929

23. Kernighan, B. W., and D. M. Ritchie, *The C Programming Language,* Englewood Cliffs, NJ: Prentice-
            Hall, 1978 2nd. ed. 1988

24. Knuth, Donald, *Art of Computer Programming*, Reading, MA: Addison-Wesley, Vol. 1. *Fundamental
            Algorithm,* 1968 ; Vol. 2. *Seminumerica*l *Algorithms,* 1969; Vol. 3. *Sorting and Searching,* 1973;
            projected to run to seven volumes

24a. Küsters, Ulrich, "Matrix Programming Languages for Statistical Computing: A Comparison of
            GAUSS, MATLAB, and Ox," with Jens Peter Steffen, in: Faulbaum, F. und W. Bandilla (eds.),
            SoftStat'97, Stuttgart: Lucius & Lucius,1997

25. Lawson, C. R., Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for Fortran
            Usage," *ACM Transactions on Mathematical Software*, 5: 308-325, 1979

26. Longley, J.W., "An Appraisal of Least Squares Programs for the Electronic Computer from the Point
            of View of the User," *Journal of the American Statistical Association*, *62:* 819 - 841, 1967

27. McCarthy, J. "An Algebraic Language for the Manipulation of Symbolic Expressions," MIT AI Lab,
            AI Memo No. 1, Cambridge, September, 1958

28. McCarthy, J., "Recursive Functions of Symbolic Expressions and Their Computation by Machine,"
            Communications of the ACM, 3: 184 - 195, 1960

28a. Naur, P., J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauer, K.
            Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger, "Revised Report
            on the Algorithmic Language ALGOL 60," *Communications of the ACM, 6(1)*:1--17, 1963

29. Nerlove, M., and S. J. Press *Univariate and Multivariate Log-Linear and Logistic Models*), RAND
            Corporation R-1306-EDA/NIH. Santa Monica: RAND Corporation, 1973.

30. Pratt, T. W., and M. V. Zelkowitz, *Programming Languages: Design and Implementation,* 4th ed.,
            Upper Saddle River, NJ: Prentice-Hall, 2001

31. Renfro, Charles G., "Econometric Software: The First Fifty Years in Perspective," *Journal of Economic
            and Social Measurement*, 200?, this issue

32. Renfro, Charles G., and "A Compendium of Existing Econometric Software Packages," *Journal of
            Economic and Social Measurement*, 200?, this issue

33. Ritchie, Dennis, *The C++ Programming Language,* Reading, MA: Addison-Wesley, 1986

34. Sammet, J. E., and R. E. Bond, "Introduction to FORMAC," *IEEE Transactions on Electronic
            Computers, EC-13(4):*386 - 394, August, 1964

35. Sebesta, Robert, *Concepts of Programming Languages*, Addison Wesley Publishing Company, 1996

36. Stigler, George J., "The Division of Labor Is Limited by the Extent of the Market," *Journal of Political Economy 59:* 185-193*, 1951

37. Stigler, Stephen M., *Statistics on the Table: The History of Statistical Concepts and Methods,* Cambridge: Harvard University Press, 1999

38. Stokes, Huston, "The Evolution of Software Design: A Developer's View," *Journal of Economic and Social Measurement*, 200?, this issue

39. Duplicates 40.

40. Stoyan, Herbert, "Early LISP History (1956-1959)," http://www8.informatik.uni-erlangen.de/html/lisp/histlit1.html , accessed September 1, 2003

41. Stroustrup, Bjarne, *The C++ Programming Language,* Reading, MA: Addison-Wesley,1986

42. Stroud, A., *The Multiple Regression Program RGR*. 1961, Madison, WI: Social Systems Research Institute, University of Wisconsin.

43. Venables, W. N., and B. D. Ripley, *S Programming,* New York: Springer, 2000

44. Venables, W. N., and B. D. Ripley, *Modern Applied Statistics with S-Plu,s 3rd ed.,* New York,: Springer, 1999

45. Wexelblat, Richard L., ed., *History of Programming Languages,* New York: Academic Press, 1981

46. Zellner, A., A. Stroud, and L.C. Chau, *Program for Computing Seemingly Unrelated Regressions Estimate,* Madison, Wisconsin: Social Systems Research Institute, Department of Economics, 1963

47. Zellner, A., A. Stroud, and L.C. Chau, *Program for Computing Two and Three Stage Least Squares and Associated Statistics*, Madison, Wisconsin: Social Systems Research Institute, Department of Economics. 1963

48. Zuse, Konrad, "Der Plankalkül," *Gesellschaft für Mathematik und Datenverarbeitung*, *63*, BMBW - GMD 63, 1972

MAJOR DATES IN THE HISTORY OF PROGRAMMING LANGUAGES

1946	ENIAC, first stored programmed machine.
1950's	Assembly language programs.
1957	FORTRAN
1958	ALGOL
1960	COBOL, LISP
1963-64	BASIC
~1964	FORMAC
1964	PL/1, IBM introduces the 360
mid-1960's	EISPAC, LINPACK, MINPACK, FFTPACK, and later BLAS
1965-68	Algol68
1968	SPSS, TSP, first volume of Donald Knuth, *The Art of Computer Programming*
1970	SAS
1971	PASCAL
1972	C
1973	Smalltalk, the first object-oriented programming language, developed by Alan Kay
1977	First micro computer introduced by Apple, FORTRAN77, Prolog, S
1978	First spreadsheet program, VisiCalc, TeX
~1978	DOS
1981-82	IBM introduces the PC, Osborne introduces the first Notebook, FORTRAN for DOS
1982	Postscript
1983	WORD, Lotus 1-2-3
1984	EXCEL, WINDOWS 1.0, first networking software, GAUSS, MATLAB
1985	C^{++}, Adobe Illustrator, MAPLE
1987	VGA (Video Graphics Array)
1987	MATHEMATICA
1990	FORTRAN90
1990's	The Internet, JAVA, etc., etc.

Sources: *Encyclopedia of Computer Science, 4th ed.,* London: Macmillan, 2000, Timeline of Computing, http://www.macmillanreference.co.uk/Science/ComputerScienceTimeline1.htm , accessed 06/14/03.

Dictionary of Programming Languages, 2000, http://cgibin.erols.com/ziring/cgi-bin/cep/cep.pl, accessed 06/14/03.