Stages in the Evolution of S

Here is a summary of how S has evolved over its long history. The following table summarizes the stages. Version numbers were applied retrospectively when work on Version 4 began. At this time it became clear that earlier, more casual terminology would no longer do. (Up to that time we had resisted anything as commercial or, perhaps, as organized as version numbers.) The practice of calling the 1988 version ``New S'' left a choice of ``Newer S'' or version numbers, once that version came to be replaced. There is a hint here that S has somewhat forced itself on its developers as a formal system, at least through the early history.

Version 1	1976-1980	Honeywell GCOS, Fortran-based
Version 2	1980-1988	Unix; Macros, Interface Language
	1981-1986	QPE (Quantitative Programming Environment)
	1984-	General Outside Licensing, books
Version 3	1988-1998	C-based; S functions & objects
	1991; rev. 1992	Statistical Models; informal Classes & Methods
	1993-	Exclusive license to StatSci (later MathSoft) for S-Plus.
Version 4	1998-	Formal class-method model; Connections; Large Objects
	1999-	Interfaces to Java, Corba?

Version 1

The earliest beginnings of S came from discussions in the spring of 1976, among a group of five people at Bell Labs: Rick Becker, John Chambers, Doug Dunn, Paul Tukey, and Graham Wilkinson. Each of us brought a slightly different mindset to the discussions. Rick had been involved in a statistical system at the National Bureau of Economic Research in Cambridge, Massachusetts before joining Bell Labs. I had been involved in an abortive attempt to design a system 11 years earlier (a little of the history of that project is recalled in the 1999 Computing with Data paper). Doug was particularly interested in time-series software. Paul had experience with APL. Graham Wilkinson was visiting Bell Labs; he had, with John Nelder, designed the GenStat system a decade earlier.

To understand the nature of S, it's important to note the context and motivation. We were looking for a system to support the research and the substantial data analysis projects in the statistics research group at Bell Labs. This motivation was different from either the perspective of a service organization or of an individual researcher in an academic situation, although we shared some of the concerns of each of those situations. For a service provider (e.g., a statistics group in an agricultural or medical consulting department), the critical requirement was to provide for an existing stream of analyses, at least some of which would be fairly standard and repetitive. For an individual researcher, the motivation for starting work on a statistical system was usually to demonstrate the value of a particular approach, in other words a ``theoretical'' motivation.

We were concerned to support serious data analysis (although for some time some of our colleagues were skeptical about serious analysis from an interactive system). However, little or none of our analysis was standard, so flexibility and the ability to *program* were essential from the start.

There was also interest in different approaches or theories of computing, and much more so in later versions. However, there seemed always to be an unquestioned assumption that the essential criterion was a system that people would *use* and in particular one that provided the techniques considered essential. Much of the early discussion was therefore about which techniques we most needed to supply, and how to do it.

The initial version of the system drew on existing tools (we called them algorithms then, nowadays module would perhaps be the appropriate term). Nearly all of these were Fortran subroutines or subroutine packages, for instance a graphics subroutine library. Rick Becker's history paper describes more of the contents, and the context. (One confusion in reading the early history is that we did use version numbers for the first couple of years, then abandoned them. These numbers are *not* the versions discussed here. Think of them usually as sub-versions of the initial version. Sigh!)

More details.

Version 2

Before and contemporarily with our initial work on S, much else was going on in the computing world. In particular, a few floors away from us the initial Unix system was evolving.

Precisely when the first version of S was evolving, the early work on Unix was taking an extremely important turn: Steve Johnson and Dennis Ritchie were working towards making Unix portable. As soon as we heard of this, and specifically of a port to a 32-bit machine, we began to realize that this was the chance to move S into a wider environment. Version 1 was written for and used only on a Honeywell (originally GE) system that itself had arisen out of an experiment in the mid 1960's to create the Multics operating system. It was an environment none of us had chosen and that none of us loved; in addition, even if one were to go with a proprietary operating system, the only sensible choice in those days would have been IBM.

A few more changes were required before we could actually move to a Unix-based S: an implementation on generally available hardware, and the writing of a Fortran compiler (we were still largely a Fortran-based system). With these changes and with some restructuring of how S worked, but with little external change from the user's view, we moved to a new version on Unix.

By mid-1980, we had a Unix implementation and a manual. By early 1981, Bell Laboratories was licensing S in source form for Unix. A first book had been written, *S: A Language and System for Data Analysis*, but initially this was printed and distributed as part of the program licensing, not published separately. At this stage, Becker and Chambers were the core developers of the system, and the authors of the book. The growing user community, however, was very much involved in developments.

Over the next few years, S began to move out, particularly into the world of statistics research, while at the same time growing in internal use at Bell Labs and AT&T. In 1984, we published a second book (or the first ``real'' book through a commercial publisher) *S: An Interactive Environment for Data Analysis and Graphics*.

Did you notice a certain uncertainty about what to call the thing? We started out with *System*, then added *Language*, then switched to *Environment*; with the next version, we would switch back to *Language* and drop *System*. We were sure, however, that we wanted to avoid the P word: S was not to be considered a statistical package in the usual sense of the term.

The ambiguity is real and goes to a key objective: we wanted users to be able to begin in an interactive environment, where they did not consciously think of themselves as programming. Then as their needs became clearer and their sophistication increased, they should be able to slide gradually into programming, when the language and system aspects would become more important. This philosophy would be articulated explicitly later, but it was implicit from the start.

More details.

Version 3

While the distribution and publication of Version 2 was still evolving, parallel research work was starting to shape the next major version. At first this seemed to be a move away from S altogether: something called the ``Quantitative Programming Environment'' was initially a separate research project, aimed more explicitly at programming and trying to emphasize that users need not be statistically sophisticated. By 1986, however, the decision was made to merge this work with the facilities (especially the graphics) underlying S, to produce a new version of S. (This explains, by the way, why the main program file for S is called Sqpe, another one of those little puzzles for users.)

The initial non-programming user would still see a fairly similar environment, but once programming began, Version 3 was very different. It had become a functional, object-based language, in its own fashion. Everything is now an object, including functions and expressions. Databases of objects (which tended to be called libraries if they mainly contained functions) got a new independence, and user libraries started to have nearly equal status with built-in libraries. The core of the system was now C-based, and the interface to Fortran (and C) subroutines was not through a preprocessor and compilation, but through S functions that invoked a subroutine dynamically from the evaluator.

Another book, *The New S Language* appeared, with Allan Wilks as a third author. The expansion of material about programming, and the QPE philosophy of trying to encourage non-statisticians to use the system, tended to reduce the statistical content of the book.

By the time this book came out, a move was underway to compensate any loss of statistical content by a project and a book that would extend the capabilities of S for statistical work. Initially this had a fairly broad framework (aiming, for example, at broad support for statistical simulation), but a lack of volunteers for other topics reduced the eventual scope to statistical models. A group of ten authors produced a corresponding library of S functions and yet another book, edited by John Chambers and Trevor Hastie, *Statistical Models in S*.

There was no fundamental change in the language itself with the work on statistical models, but one important addition was made: a first attempt at including classes and methods. The main impetus was to present a simple user interface to dealing with the diverse kinds of data encountered in fitting and examining statistical models. Objects had an *optional* class attribute, and functions could dispatch methods by explicit program control. The added code in the S evaluator was localized and not extensive.

Version 4

The next stage of research after the completion (and, about a year later, some revision) of the statistical models software, was to turn again to the issue of S as a programming environment. In a sense, this was a continuation of the QPE work of a decade earlier, but of course against a very different general computing background.

A number of partially inter-related changes were made, with the overall goal of working out a programming environment that could move smoothly from the initial interactive use to increasingly large-scale software projects. Included were a more explicit approach to classes and methods, an introduction of S-dependent C programming, some techniques to handle larger objects, and explicit management of tasks and events. The design was described, midway in the project, in the 1996 Evolution of the S Language paper, and of course in various places in the **Programming with Data** book that accompanied the released version in 1998.

From a general computing view, the philosophy tries to combine aspects of functional and object-oriented (i.e., method-centered) approaches. But as in previous stages of the evolution of S, adherence to a formal approach tended to be compromised when it conflicted with what we saw users as needing. An additional limitation, not to mention a major overhead, was the attempt to maintain the largest possible level of back-compatibility with Version 3.

John Chambers<jmc@research.bell-labs.com>
Last modified: Tue Mar 7 10:41:20 EST 2000