Journal of Statistical Software

February 2005, Volume 13, Issue 7. http://www.jstatsoft.org/

On Abandoning XLISP-STAT

Jan de Leeuw
University of California at Los Angeles

Abstract

In 1998 the UCLA Department of Statistics, which had been one of the major users of Lisp-Stat, and one of the main producers of Lisp-Stat code, decided to switch to S/R. This paper discusses why this decision was made, and what the pros and the cons were.

Keywords: software, Lisp.

1. Introduction

The UCLA Statistics Department had invested much in XLISP-STAT. And the same thing is true for me personally. The department used XLISP-STAT for graduate teaching (undergraduates use Stata). Some of us extensively used XLISP-STAT in our research. Much of the **Gifi** software was re-written in XLISP, and Ker-Chau Li wrote his **SIR** programs in XLISP as well. The department maintained a large repository of Lisp software relevant for statistics, still available at the hidden location `http://www.stat.ucla.edu/xlispstat/code/`. And some of us, me in particular, contributed a large number of statistics and utility programs to the XLISP archive. This was both intended as a service to the community and as a way to make the acceptance of XLISP-STAT more widespread.

Then, about 5 years ago, we started noticing that more and more work was done in S/R and less and less in XLISP-STAT. The gap between the actual XLISP-STAT implementation and the available documentation was growing. Updates to the basic distribution came with larger and larger time intervals, and then stopped alltogether. People started looking more and more startled when you told them XLISP was your main language. And, on a somewhat larger scale, the Lisp user community was not growing much either, and much of what was done in Lisp was now done in newer (an leaner) interpreted languages such as Perl, Python, or Ruby. All in all, the developments did not seem to indicate a healthy state of affairs.

Something had to be done. In 1998 we decided to switch from XLISP-STAT to R.

2. Why we had to do it

It is obvious now, and it was obvious then, that S was rapidly becoming the lingua franca of statistics. When JSS was started in 1996 it seemed likely that many of the submissions would be written in XLISP, and indeed in the first few years many of them were. In the RFC phase of JSS, some people warned us to open up the journal, and not to restrict is to "XLISP Jockies". Although JSS publishes materials written in many languages, the only jockies around at this point are R jockies.

It is very important for a discipline to have a common computer language. Engineering and numerical analysis has MATLAB, and statistcs has S/R. But, at a higher level, sociologists have LISREL, psychologists have SPSS, business has SAS and Excel. It makes communication easier if files are portable and various computer terms mean the same things to the same people. The more important the computer becomes, the more important it becomes to standardize.

We also felt we could not ignore the fact that most statistics software, certainly at the research level, was now written in S/R. My current personal R binaries at http://gifi.stat.ucla.edu/pub/ come with close to 500 add-on packages, implementation a great number of both standard and esoteric statistical techniques, as well as a great many utilities.

And it was not very responsible to teach our students XLISP-STAT, if we were one of the few places in the world where they could actually use it and share it with their colleagues. We were not teaching a marketable skill. It is already difficult enough to get open source projects such as R, accepted by industry and business, but there at least we have the excuse that all of statistics uses it. Moreover there is the commercial product S-PLUS which comes with the usual guarantees and support contracts. XLISP-STAT would not have a chance.

3. Why it hurt

Learning a new programming language every once in a while is both necessary and refreshing. Professional programmers are sometimes advised to learn at least one new language each year. It is seldom pleasurable, however, to switch one's main language, for the same reason why it is stressful to move to another country in the middle of one's life. This must be one of the reasons why a language such as FORTRAN refuses to die. It does its job well, and people have no reason to switch. I was not so fortunate, since I had to give up PL/I in the seventies, APL in the eighties, and now Lisp in the nineties.

Switching to S from Lisp was painful for other reasons. Although the R implementation of S is based internally on Scheme, a Lisp dialect, the two languages are quite different. People either love or hate Lisp, and I happened to love it, parentheses and all. It is a nice clean language, interesting from the mathematical point of view, and with a ANSI standard available as well. S is an evolving language, obviously specialized for statistics, which huge chunks still being added. There is no byte compiler yet (although it's on its way), namespaces were put in recently, there are two competing systems for OOP, there are competing ways to link in compiled C or FORTRAN, and so on.

As I mentioned above, we had written a large amount of code in XLISP, and that became virtually useless and had to be translated or reprogrammed. Not only that, but there is an enormous amount of Lisp code available on the net that can be made to run in XLISP-STAT with minor changes. We lost that too. Together with the Lisp community, the mailing lists,

and so on. Also, of course, major projects developed by us and others became threatened. No matter how vigorously you develop **SIR**, **ViSta**, **Arc** or **Gifi**, it remains true that the runtime engine and the support code is no longer updated and may not keep up with hardware developments. As this special issue of JSS shows, there are still some courageous people keeping up the good work, but in the back of their minds they may fear that they are rapidly heading for oblivion. More rapidly than the rest of us, I mean.

Even more importantly, Lisp was so well documented. It had the Common Lisp ANSI standard, and the definite book by Steele (1990). The best computer programming book (Abelson, Sussman, and Sussman 1985) in the world used Scheme as its language. The books by Norvig (1992) and by Graham (1994, 1996) were masterpieces of programming style, and they use Common Lisp. There were books on how to do OOP in Lisp, how to write compilers and interpreters, and so on. We gave up compilers from FORTRAN to Lisp, or from Lisp to C. None of this is really available for S/R. The brown, blue, white, and green books (Becker and Chambers 1984; Becker, Chambers, and Wilks 1988; Chambers and Hastie 1992; Chambers 1998) to a large extent focus on statistics, not so much on programming, and they describe a slowly evolving and not obviously converging language. The books by Venables and Ripley (2000, 2002), however excellent, are severely handicapped by the fact that they cover at least two competing dialects, and that they will rapidly become outdated.

I also should mention that I very much liked the idea of taking a general purpose programming language, such as Lisp, and adding the statistics on top as a library or a set of plug-ins. There are now various more or less satisfactory attempts to put S/R on top of Java, Perl, and Python but they have the major disadvantage that they involve mapping two competing systems. Imagine the situation in which R was actually written in a combination of Python and C. There would be no need to replicate the general purpose programming tools and functions already available in Python. Thousands of Python extensions and hundreds of books and tutorials would immediately become available. We would be back in the situation that we were in with Lisp-Stat.

As I said, having a common language is important for a discipline. That does not mean it is necessarily optimal from a more global point of view. As we know, most new research is interdisciplinary and as a result some of the researchers in a project will use MATLAB, some will use R, and in some projects some of them may even use Excel or SAS. Thus a reorganization of research implies that the language problem arises again. This is one main reason why putting the scientific software on top of a general purpose programming language (in the same way as we used to put subroutine libraries such as IMSL on top of FORTRAN and C) is still a good idea, maybe even a better idea than developing special-purpose little languages.

It is also important not only to standardize on a language, but even on a GUI and a runtime system. That is why it is unfortunate that we still have two major and incompatible implementations of the S language, S-PLUS and R. One of the main tasks of the S/R community in the immediate future is to make sure R and S-PLUS will become different interfaces to the same language.

The academic origins of S/R have the usual unfortunate anarchistic corollaries – there are now maybe 25 different GUI's and more and more are being developed. Somebody should be in charge here. Also, for similar reasons, R suffers from the curse of unfinished projects which is very common in academia. Many pieces of software, especially some of the very interesting

bridges to other languages, are proofs of concept. In the relatively low-pressure academic situation it is more rewarding to go on to the next proof of concept, instead of finishing and documenting the previous one. Because of standardization, commercial input, and a much larger community of both developers and users, more got finished in the Lisp environment.

We also had to give up some XLISP-STAT components which still have not been replicated in R. Dynamic graphics, for instance, and the byte compiler. R has to do its dynamic graphics by making calls to the standalone **xgobi** or **ggobi** programs, and it does not have tools to do dynamic graphics programming yet. I am sure this will come at some point in time, and it also seems dynamic graphics is not as hot a topic any more as it was in the late eighties, but it has still not been possible to create something as complex and as satisfactory as the **Arc** or the **ViSta** packages in R. It may be possible to do something similar with the right combination of plugins and helpers, but so far nobody has taken the time to actually try it.

4. Conclusion

The main difference between Lisp-Stat and S/R is that between a set of commands added to a large and popular general purpose programming language and a special-purpose little language for statistical computing. My personal opinion is that it is unfortunate that the statistical community made the choice that it made, because more was given up than was actually gained. If Lisp-Stat had been developed as actively, and by as many people, as R has been over the last 15 years, then perhaps we would have a more satisfactory product at this point in time. Nevertheless, as soon as the statistical community had made its choice, departments such as ours really had no choice, and had to promote R (or become sectarian).

Of course one can still argue that Lisp-Stat has its place as research software, or to do dynamic graphics, but we interpreted it from the start as a tool to teach statistics to graduate students and to communicate statistical techniques to others. As a common language for statistics, that is. Clearly it no longer is a candidate for that position.

References

Abelson H, Sussman G, Sussman J (1985). *Structure and Interpretation of Computer Programs*. MIT Press.

Becker R, Chambers J (1984). *S: An Interactive Environment for Data Analysis and Graphics*. Wadsworth.

Becker R, Chambers J, Wilks A (1988). *The New S Language: A Programming Environment for Data Analysis and Graphics*. Wadsworth.

Chambers J (1998). *Programming with Data: A Guide to the S Language*. Springer.

Chambers J, Hastie T (eds.) (1992). *Statistical Models in S*. Wadsworth.

Graham P (1994). *On Lisp: Advanced Techniques for Common Lisp*. Prentice Hall.

Graham P (1996). *ANSI Common Lisp*. Prentice Hall.

Norvig P (1992). *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann.

Steele G (1990). *Common Lisp: The Language*. Digital Press.

Venables W, Ripley B (2000). *S Programming*. Springer.

Venables W, Ripley B (2002). *Modern Applied Stastistics with S*. Springer, 4th edition.

Affiliation:

Jan de Leeuw
Deparment of Statistics
University of California at Los Angeles
Los Angeles, CA 90095-1554, United States of America
E-mail: deleeuw@stat.ucla.edu
URL: http://gifi.stat.ucla.edu/

Journal of Statistical Software
February 2005, Volume 13, Issue 7.
http://www.jstatsoft.org/

Submitted: 2004-06-23
Accepted: 2005-01-05