The usefulness of the L_S book would be greatly increased by including more complete descriptions of each function, organized either alphabetically as in the S book or by topic as is done in the description of the Common Lisp language (Steele, 1984).

I end with some minor criticisms of the L_S book.

First, there is disconcerting ambiguity throughout the book about which version of Lisp underlies L_S. It would have been better to commit to presenting L_S as it is implemented in XLISP, and then if/when it is released for Common Lisp, this could be accompanied by a document describing the differences. It is confusing that some ideas are presented generally when only one case applies to the current implementation of L_S. Examples of this are the discussion of lexical and dynamic scoping, and tail recursion, neither of which are relevant to L_S.

Second, the coverage of Lisp is varied. For example, though macros are mentioned obliquely in the text, their use is never discussed. This might be have been a concious choice since Abelson and Sussman (1985) deprecate the use of macros in their description of Lisp. But macros play an important part in Lisp programming and should not have been ignored.

Third, the index is not complete (for example, no entry on eval), but this applies to both books.

Fourth, there is too much Lisp code. I am someone who has a high tolerance for reading code and loves nothing better than wallowing around in piles of parentheses, but the density of Lisp code was too much even for me in some of the sections. Much better to my mind would have been to supply all of the Lisp code with the L_S system, and refer to relevant new ideas as they are introduced. This is worst in the last chapter, discussing dynamic graphics examples. But also in some of the earlier chapters, there is too much reliance on presenting code. Sometimes an entire function is presented many times as its evolution is discussed. Again, it would have been better just to present the relevant changes to the function in each new version.

7. CONCLUSION

By providing a broad range of statistical and mathematical primitives and an interpretive language, combined with good graphical facilities, together, these two systems define the state of the art in computing environments for statisticians. Each system is better suited to certain applications, and for data analysis and research, statisticians can only benefit by acquainting themselves with both.

ACKNOWLEDGMENTS

Thanks are due to John Chambers, Rick Becker, Allan Wilks and Luke Tierney who all provided me with valuable comments on this review.

Rejoinder

Luke Tierney

I would like to thank the reviewers for their comments and for their efforts in working through the book and the software, and I would like thank the editor for this opportunity to comment briefly on a few of the issues raised in the reviews.

FUNCTIONALITY AND EXTENSIBILITY

Several of the reviews point out that the basic Lisp-Stat system does not include a wide range of specialized analyses. This is quite deliberate. A major advantage of an extensible system is that it allows experts in using and developing a particular methodology to provide tools for implementing the methodology. If the Lisp-Stat system is found to be useful then, over time, this should lead to a wider set of better tools than can be provided by a single implementor or small group of implementors.

A comparison with the evolution of the S system may be helpful. The basic S system as described in Becker, Chambers and Wilks (1988) also does not directly support a side range of different analyses. But few sites now provide only the basic S system. Most augment it with the facilities of S-Plus, a variety of locally written code, selections of code

Luke Tierney is Professor, School of Statistics, University of Minnesota, Minneapolis, Minnesota 55455.