Comment

Forrest W. Young

1. INTRODUCTION

This review, which covers the Lisp-Stat software as well as the Lisp-Stat book, is organized around the full title of the book, to which I have added the words *functional, extensible* and *free*. Fortunately, Luke Tierney didn't use all of these words in his title (which would have become something like "Lisp-Stat: A Functional and Object-Oriented Environment for Statistical Computing and Dynamic Graphics that Is Extensible and Free"), but this expanded title does incorporate what I see as all of the most important aspects of Lisp-Stat, and does make for a good way of organizing this review.

2. LISP

The key insight in Lisp-Stat is that it would be reasonable, even beneficial, to develop a statistical system based on the Lisp language. This insight was not Tierney's, but rather it belongs to one of the computational statisticians who have advocated such a development over the last few years (see McDonald and Pedersen, 1988; Oldford and Peters, 1988; Stuetzle, 1987; and Buja, Asimov, Hurley and McDonald, 1988). At first, Lisp seems like a very strange choice, especially for those of us who are only familiar with procedural languages such as Fortran, C or Pascal. In fact, Tierney asks very early in his book (page 3) the rhetorical question: "Why Lisp?" His answer: Lisp combines the strengths of a general purpose programming language with the needs in data analysis for interactive, experimental programming. He also notes that Lisp is a functional language that is object-oriented and that can be easily and elegantly extended (see below). There are few, if any, other general purpose languages that provide all of these characteristics.

At first, Lisp is a bit strange to use, since it is based entirely on prefix notation using a very simple and consistent syntax. However, the consistency and simplicity of the notation and syntax is soon seen to be one of the true strengths of the language. Indeed, the language and the statistical system are very easy to learn and to teach to novice programmers and novice data analysts. The tutorial chapter (2) is good for getting started with Lisp-Stat and Chapters 3 and 4 on Lisp provide an introductory overview of Lisp that is sufficient for using Lisp-Stat. If one wants to do serious program development a book specifically on Lisp (Winston and Horn, 1989) will be needed.

Lisp-Stat is currently based on the XLISP implementation of Lisp, which is a subset of the industry-standard version of Lisp known as Common Lisp. XLISP has two drawbacks. The major drawback is that XLISP has no compiler, only an interpreter. While this is appropriate for "interactive, experimental programming," a compiler is better for Lisp-Stat code that is stable and that will be re-used many times. Tierney (personal communication) says that a new implementation of Lisp-Stat, based on Kyoto Common Lisp (KCL), which has a compiler, is being developed. The second drawback, which seems to be less critical, is that XLISP does not implement CLOS, the industry standard Common Lisp Object System. Since object-oriented programming is very important for dynamic graphics, Tierney has developed his own, nonstandard subsystem. While I have no experience with CLOS, Tierney's object system (TOS?) is easy to learn and use, and seems appropriate for dynamic statistical graphics and for other data analysis uses. The KCL implementation will include both object systems.

3. STAT

What Lisp lacks, and what Lisp-Stat adds, are the capabilities needed for statistics, data analysis and statistical graphics. Thus Lisp-Stat is Lisp extended to include vectorized arithmetic, functions for basic statistical computations (mean, standard-deviation, etc.), functions for maximization and maximum likelihood estimation, functions for linear algebra and matrix manipulation (transpose, svd, etc.), an interface to several windowing systems (X-Window, Amiga, Macintosh, but not yet Microsoft Windows) and tools for constructing graphs and dynamic graphs.

What this provides is an environment in which it is easy to perform basic statistical calculations, optimization, matrix algebra and dynamic graphics. *What this does not provide* (and here we have the major limitation of Lisp-Stat) is an environment in which it is easy to manipulate data (there is no spreadsheet, no datasets, etc.) or to perform

Forrest W. Young is Professor of Psychometrics and Biostatistics at the University of North Carolina, BC 3270 Davie Hall, Chapel Hill, North Carolina 27599-3270.

major, "prepackaged" statistical computations (such as analysis of variance, cluster analysis, etc.). Other than "model-objects" (analysis procedures) for multiple regression (linear and nonlinear) and approximate Bayesian computations, there are no major statistical analysis procedures. More discussion of these limitations appears below.

4. FUNCTIONAL

Lisp is a functional language based on prefix notation. Thus, if one wants to take the mean of a vector containing precipitation levels in a number of cities, the appropriate Lisp statement is

(mean precipitation),

which seems natural. However, if one wants to add the elements of two vectors, say for different types of automobile emissions, the statement is

(+ hydrocarbon carbon-monoxide),

which at first takes some getting used to. If one wants to also define a new vector as the sum of those two, the statement becomes

(def total (+ hydrocarbon carbon-monoxide)),

which seems even stranger (there is no assignment statement, assignment being done by the prefix function def). However, these initial concerns soon disappear and are replaced for an appreciation for the simple elegance of Lisp's slavish use of prefix notation, and the power of being able to nest functional statements indefinitely deep. (Just as an example of the power of Lisp's simple notation, recursive programming is so straightforward that if one hasn't used other languages such as Fortran or C, one doesn't think that recursive programming is something special.)

The statistical extensions to Lisp consist in large part of Lisp functions modeled on the S statistical system (Becker, Chambers and Wilks, 1988). These functions, most of which are vectorized, include basic data-handling functions, sorting, interpolation and smoothing, probability distributions, matrix manipulation, linear algebra and regression computation. The vectorization is particularly nice, making linear algebra and matrix manipulation straight-forward and, since the vectorization is written in the underlying compiled C language, very fast. Chapter 5 presents these functions and has examples of them being used to construct a projection operator and to create a robust regression function. The statistical functions fit cleanly into Lisp, due to its basic functional nature, and the examples are sufficient to enable the reader/user to construct his or her own additional statistical functions.

5. OBJECT-ORIENTED

Perhaps the most important, and least familiar, feature of Lisp-Stat is its object-oriented programming paradigm. This feature will be unfamiliar to most statisticians, since no widely used statistical system is object-oriented. However, recent developments in computational statistics seem to indicate that an object-oriented programming paradigm is very useful for graphics programming, for developing flexible data structures and for representing statistical models.

In an object-oriented environment, an object is a collection of data and programming code, where the data somehow "knows" how to use the code. In Lisp-Stat, the data could be "variables" and the code could implement a plot or an analysis procedure. The user could send a message to a variable asking it to "plot itself," and, depending on the measurement nature of the variable (continuous, discrete, ordinal, etc.), the plot would be a boxplot, a histogram, etc. The point is that the user doesn't have to know the measurement nature of the variable, since the variable knows its own nature and can determine which kind of plot is appropriate.

As mentioned above, Tierney has developed his own prototype-based object system, a system that does not correspond, even philosophically, to CLOS, the standard Lisp object system. While this may be a problem for Lisp aficionados, the reasons given by Tierney for the nature of his object system stem from the observation that statistical programming is "experimental programming," and from his (among others) conclusion that a prototype-based system is more appropriate for this type of programming. Lisp-Stat provides prototype data-objects, model-objects and graphical-objects. The adequacy of these objects varies a great deal, as I discuss in the next two sections.

6. STATISTICAL COMPUTING

The statistical computing environment consists primarily of statistical functions, data-objects and model-objects. As I mentioned above, the statistical functions are based on S and are very complete. On the other hand, the data-object seems to be poorly designed, providing weak foundation for further work. The example that I presented in the previous section (data that "know" their measurement nature and can use this knowledge to decide which kind of plot to produce) cannot be done with the data-object that comes with Lisp-Stat. In fact, the data-related aspects of Lisp-Stat are its weakest

point, with a lot of work needing to be done to make its data-handling capabilities as useful and powerful as many other systems. There is also very little attention given to input/output facilities, other than to standard file system manipulation capabilities, and to the extensive dynamic graphics discussed below.

Of greater concern is that I found the data-object examples in Chapter 6 to be nearly nonexistent and not particularly useful in helping me to strengthen Lisp-Stat's data handling facilities. It is not yet clear to me that the design of the data-object that is provided is useful for the development of additional data-objects. Perhaps better examples would convince me the design is good. Of course, lack of a powerful standard data-object means that individual users can develop whatever types of data-objects they find most useful for a particular application. While this could be a strength, it becomes a weakness in the absence of guidance in constructing new data-objects, let alone guidance showing what kinds of data-objects would be best for specific uses. Furthermore, lack of *any* standards may well lead to substantial divergence between developers creating objects for similar tasks.

The two model-objects (multiple regression and nonlinear multiple regression) that are provided with Lisp-Stat are well designed. While these two objects do not form a complete set, their discussion in Chapter 6 (and the discussion in the recent research report by Tierney of a model-object for generalized linear models) is quite useful in guiding the development of new model-objects. It has been fairly straight-forward for myself and my students to develop model-objects for principal components, discriminant analysis and multidimensional scaling, even though none of us had previously used an object-oriented system.

7. DYNAMIC GRAPHICS

The graphical-object system is very complete and the design is well thought out. It is described extensively and clearly in Chapters 7–10, with examples that are very well constructed. The system of five graphical-objects includes all of the recent popular dynamic graphics facilities (spinning plots, scatterplot-matrices, brushing, linking, etc.), and the examples show how to create additional dynamic graphics such as grand-tours and parallel coordinate plots. What is most exciting is that the graphical-objects can be easily modified to perform exactly what you have in mind. For example, I have created a guided-tour biplot object based on my Visuals algorithm (Young and Rheingans, 1991), which took only about 2 weeks to develop as compared to the 2 years that it took a professional programmer to do the original version! My students and I have also created SpreadPlot (Young, Faldowski and Harris, 1990), a graphical spreadsheet that involves a fundamentally new type of linkage that didn't exist within Lisp-Stat (or in any other system). While SpreadPlot is based on ideas that I have had for a number of years, I never conceived it could actually be created without hiring a professional programmer who would spend several years working on it. With Lisp-Stat, my students and I created SpreadPlot in a few months, even though we were all just beginning to learn the Lisp-Stat system! The graphical-object system is truly marvelous.

8. EXTENSIBLE

Lisp-Stat is a totally "open" environment which the competent computational statistician (and his/her students) can easily extend. The functional nature of Lisp makes it trivial to define new functions. It is also possible to write programs in Fortran or in C and to dynamically load them so that they are accessible from within Lisp-Stat. Of greater importance, it is fairly straight-forward to define new model-objects, so that one can have a complete range of desired model-objects that perform major, "prepackaged" statistical computations. It is also possible to define new graphical-objects that implement new dynamic graphics ideas. Both of these ways of extending Lisp-Stat are possible since the Lisp code for the supplied model-objects and graphical-objects are provided in the software, and since the descriptions of them in the book serve well as examples of how to define new objects. Thus, as I mentioned above, my students and I have been able to construct new model-objects and graphical-objects even though we had no prior experience with Lisp. The same should also be true for data-objects, but for reasons noted above their extension is more difficult. Note, finally, that the entire C source code of Lisp-Stat (which is, after all, a C program) is provided and can serve as a basis of further extensions.

9. FREE

The Lisp-Stat software is available at no expense by simply down-loading the Lisp-Stat source code from a server at the University of Minnesota's Statistics department. This code is then compiled and will run on any UNIX workstation under X-Window, or on Suns under SunView. Macintosh, Microsoft Windows and Amiga executables are also available for free from the server. There are no restrictions on copying the software and making it

available, free-of-charge, to others. There is, in essence, no support provided, other than an effort by Tierney to fix bugs, and minimal on-line help. This, of course, is a mixed blessing, but since the quality of the software is very high, and since there are few bugs, there seems to be little need for support.

The book, of course, is not free, though it is reasonably priced. I have already described all of the chapters in the book, emphasizing that the book is especially strong on dynamic graphics. The book is useful as Lisp-Stat documentation, providing a tutorial and examples of using and extending the system. The book is also a good introduction to functional and object-oriented programming, as used in statistics, and to Lisp.

10. CONCLUSION

Lisp-Stat is the most important, exciting and promising development in computational and graphical statistics in recent years. It provides a foundation on which computational statisticians can build a statistical system offering all types of statistical and data analysis tools—from basic to advanced—to all types of users—from novices to sophisticates. As it stands, Lisp-Stat is not (and does not claim to be) a statistical system that provides a wide range of analysis tools for a wide range of users. However, with the proper extensions, Lisp-Stat could become the standard by which other systems are judged. In sum, Lisp-Stat is the statistical environment "for the best of us," not "for the rest of us"—yet.

Comment: Two Functional Programming Environments for Statistics — Lisp-Stat and S

David J. Lubinsky

1. GENEALOGY

There is a German saying, "Tell me where you come from and I will tell you who you are," and this is perhaps even more true for the two statistical environments that are the subject of this review. They are both the products of many ancestors and each reflects its heritage. Both Lisp-Stat (L_S; the idea of this notation is that Lisp-Stat is a Lisp system specialized for statistics) and S are descendants in the line of interactive, interpretive systems, starting with APL and Lisp; L_S also draws inspiration from S, Smalltalk and dynamic graphics systems (Cleveland and McGill, 1988). The two systems are both interpretive programming environments using functional languages. They each have vectorized arithmetic operations and support a wide set of statistical primitives. In addition each has strong support for graphical display of data.

In the family of statistical and computing systems, L_S and S are very close, and as in all families, there is a natural rivalry, but also a natural affinity between them. S is the older brother, more mature and more complete. Whereas L_S is faster, incorporates many new ideas in graphics and object-oriented programming, but still has a long way to go before it can compete with S in all areas.

They are both designed to be used for more than canned analyses of data. Each allows users to combine standard analyses in nonstandard and flexible ways and, more importantly, to implement and experiment on new techniques.

The rest of this section introduces each system by presenting the same function coded in each language and discusses the general areas of application in which each system is stronger. The following section is a more detailed comparison of languages and primitives in each system. This is followed by a comparison of performance of the systems, and the last section discusses the their documentation.

1.1 A Running Example

To help make the similarities and differences more concrete, Figure 1 shows how one would implement a running smoother using S and L_S. Each function takes two required arguments, **x** and **y**, and returns a set of smoothed values at equally spaced points along the range of the **x**'s. They also take two optional arguments, the function to be used to find local values of the smooth, and the number of points in the returned smooth. Two obvious examples of local smoothing functions would be the mean (the default) and the median. These

David J. Lubinsky is a member of the technical staff at AT&T Bell Laboratories, Room 25524, Crawfords Center Road, Holmdel, New Jersey 07733.