The paradigm of being able to treat statistical procedures and data as viewable information is a useful one. One extension which is immediately possible based on this work would be in the possibilities of multi-media e-mail. One could conceivably discuss a proposed analysis through e-mail, with the relevant information passed back and forth.

There are further mechanisms, not examined here, which can be extremely useful for distance learning and the spread of knowledge. For example, there is a mechanism, MBONE, similar to usenet news and WWW , for transmitting audio and video (movies) over the Internet. For the course on C++ programming which was mentioned, students communicated with the teaching assistants through email, "talk", and the Internet Relay Chat (the latter two are means for real-time communication by typing). Other mechanisms include MUDs and MOOs (Multi-User Dungeons and MUDs, Object Oriented), which are similar to the old text adventure games, except that multiple players are involved, and objects can be left by different users for each other. For visual communication, there is the *CuSeeMe* package, which transmits audio and visual between two workstations, almost like a visual phone call. In this manner, a virtual classroom can be constructed.

Appendix: Technical Details for UNIX systems

In order to use the server software developed at Penn State, the client must be able to translate the document. For this purpose, a new information type must be defined for Mosaic, application/x-stat-xlisp, and the viewer needs to be specified. The viewer is called *xlispstat.wrap* to denote that it is a wrapper around Xlisp-Stat , i.e. a program which makes sure xlispstat starts up properly. With this installed, the following information should be put into a global *mailcap* file or into a user's personal *mailcap* file (i.e. *.mailcap* in the user's home directory).

```
# Statistical information
application/x-stat-xlisp; xlispstat.wrap %s
```

A sample *mailcap* file and the script for the wrapper are available via anonymous ftp from `ftp.stat.psu.edu` in the file `/pub/statlearn/xlispclient.shar`.

The other side of the story is that the server must know that files ending in `.lsp` are xlispstat files. This is usually declared in the configuration file of the http server (the program that serves hypertext documents to clients). This requires that the configuration files be modified. An example of the code to be added for the NCSA HTTPD server is as follows:

```
# Added for statistical information,
# AJR (091694).  EXPERIMENTAL!
AddType application/x-stat-xlisp  lsp
```

These lines must by placed into the `srm.conf` configuration file. This would vary for different HTTPD servers.

One could do a similar thing for other statistical packages; while this transfers easily to Splus and Minitab, it also could be set up for SAS or other generally batch systems. For Splus or Minitab, it would suffice to choose a local suffix—perhaps `.q` for Splus and `.mtb` for Minitab. Then one could construct a universally accepted MIME type such as application/x-stat-splus and application/x-stat-mtb and write the respective interfaces.

Anthony J. Rossini
Center for Biostatistics and Epidemiology
Hershey Medical Center at Penn State
arossini@biostats.hmc.psu.edu

James L. Rosenberger
Department of Statistics
Penn State University
JLR@stat.psu.edu

SECOND FEATURE

The Lisp-Stat Statistical Environment

by Jan de Leeuw

A Question and an Answer

Name a statistical package which

- is infinitely extensible, open, and comes with an elegant object-oriented extension language;
- is fast, use garbage collecting, and has a byte-compiler for additional speed;
- is interactive, dynamical, and graphical, and allows you to build to own graphical interfaces;
- is portable, and uses the native window system on Mac, MSW, X11;
- is free.

After this blatant commercial beginning, everybody is waiting for an answer. Here it is. **Lisp-Stat !!**

Some History

Lisp-Stat is written in a combination of C and Lisp, just like GNU Emacs. The basic language used to write the interpreter and the build-in functions is C, and Lisp is used as the extension language of the system.

We shall not discuss the complicated history of the Lisp language. If you want to know more, you can read

`~/common/docs/Evolution-of-Lisp.ps`

Note: In the sequel we shall refer to various documents on our ftp server. The root for paths we give are `ftp://ftp.stat.ucla.edu/pub/lisp`.

For our purposes it is sufficient to know that in the early eighties Lisp systems were huge and expensive. Therefore David M. Betz decided in 1985 to write a small and portable interpreter for a subset of the Lisp language. PC and Mac users could run their own Lisp systems by using his Xlisp, and they could experiment with object-oriented programming as well. The interpreter was written in C. Xlisp 2.0, released by Betz in 1988, has many additional functions and data types build in, and the newer releases which are based on the work of Tom Almy, Niels Mayer, and Luke Tierney are getting closer and closer to Common Lisp. Documentation is in `/xlisp/docs`. There is a reference manual (for Xlisp 2.1) by Tim Mikkelsen, and introduction to Object-Oriented Programming (OOP) by Tim Mikkelsen, and a little manual on how to extend Xlisp at compile time by adding additional C functions by Jeff Prothero.

In 1988 Luke Tierney published the first version of Lisp-Stat, which was documented in the technical report `/xlisp/xlisp-stat/docs/xlispstatdoc.ps` Tierney took Xlisp 2.0 as his starting point, and added functions for linear algebra, optimization, statistics, and graphics programming. Some of these functions are added at the C level (i.e. at compile time) and some at the Lisp level (i.e. at run time). Then in 1990 he published the book *Lisp-Stat. An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. Unfortunately Xlisp 2.1g, which is used in the newest versions of XLisp-Stat, is so much larger than Xlisp 2.0 that the account of the system given by Tierney in his book is quite incomplete. An update to the book is in `/xlisp/xlisp-stat/docs/changes.ps`.

The various names illustrate, by the way, the relationship between Lisp-Stat and Xlisp-Stat. Any Lisp system with the statistics and graphics added is a Lisp-Stat system, but adding the Stat component to Xlisp gives Xlisp-Stat. There is an (incomplete) version of Common Lisp Stat, and we can easily image Elisp-Stat or Scheme-Stat.

Some Comparisons

We have called Lisp-Stat a statistical package. This is somewhat misleading, because there are so many different types of packages, and Lisp-Stat is different from almost all of them. It is not similar to the dinosaurs SAS or SPSS, which apply relatively routine statistics to huge data sets, with gigantic amounts of overhead. It is also not similar to the many PC/Mac packages that do more or less the same thing as SAS and SPSS, only on a somewhat smaller scale, and with a little bit more graphics. I think the only useful comparisons are with Matlab, Mathematica, DataDesk, and in particular with S-plus.

A fairly extensive comparison between Xlisp-Stat and S is contained in a collective review of List-Stat in the November 1991 issue of Statistical Science. The issue has reviews by Baxter and Cameron, Weihs, Young, and Lubinsky. Especially Lubinsky makes fairly detailed comparisons with S. While S, especially in its S-plus implementation, is more complete and glossy, Xlisp-Stat is generally faster and has (much) better memory management. The speed advantage is more pronounced now than in 1991, because Xlisp-Stat has a byte-compiler in its newer versions, and because of its portability, it has been rapidly set up on newer and faster systems. S has more statistics, Xlisp-Stat has more graphics. S is perhaps more appropriate for doing statistics, Xlisp-Stat is more appropriate for producing statistics software, because it comes with a complete GUI builder. S is better for those who write papers in which statistics are applied, Xlisp-Stat may be better for those developing statistics.

Another important comparison is the language. Xlisp-Stat uses Lisp, a general purpose language with many applications. S uses the S-language, a special purpose language geared to statistics. I happen to think that if you work in a UNIX environment, then learning Lisp is a good investment (think of Emacs, Maxima, and so on). In fact, some free Common Lisp systems (such as cmucl) are now available that can even do fast floating point, and code for these systems is often very close to Xlisp code.

What it comes with

Xlisp-Stat comes with everything in Xlisp-2.1g, including many of the functions in the various Lisp standards. In fact I would guess (but I am not sure) that the system contains well over 500 Lisp functions not related to statistics. This provides an extraordinary powerful basis for program development. Many of the functions are completely undocumented in the Tierney book, and even in the Mikkelsen reference manual. For instance, Xlisp now contains the system package from Common Lisp, with the many functions that come with it. Xlisp now can also save workspaces, similar to APL

workspaces, which makes it easy to customize and save work between sessions.

Another important addition to current versions of Xlisp-Stat is the byte compiler. The lisp code can be byte-compiled, and byte-compiled code generally loads and executes much faster than the original lisp. For code with lots of loops and lots of consing, using the byte compiler can mean speed ups of 500% or more.

Of course all the basic arithmetic and transcendental functions are there, including the beta and the gamma. Descriptive statistics such a mean, median, standard-deviation, quartile, max, min, and fivenum are available. The numerical linear algebra functions are the usual, with inverse, determinant, solve, eigen, sweep, qr-decomposition, cholesky-decomposition. Arrays with unlimited dimensionality are native in Lisp, and there are various primitives to handle indexing (this has been extended greatly with add-on Lisp code, see below). There are two general purpose optimizers, one based on Newton's method and one based on the Nelder-Mead simplex method, and there is a somewhat idiosyncratic but very useful (they tell me) module to approximate the posterior mode using Laplace expansions.

A nice generalized linear modeling package is added at the Lisp run-time level. It is similar to the nonlinear regression package because it inherits much of the functionality of the regression model prototype. Regression is one of the areas in which Lisp-Stat really shines. It forcefully demonstrates the elegance of an object-oriented organization, and combination of the regression and graphical facilities has lead to many great tools. (Note: The OOP system in Xlisp-Stat is prototype-based, while the OOP in Xlisp is class-based. This means that actually two systems are available, but Xlisp-Stat users will routinely use the prototype-based one.) The recent *Introduction to Regression Graphics* by Cook and Weisberg is an impressive example of the power of these modules. It is a clear illustration that Xlisp-Stat is ideally suited to develop modules for statistics teaching, modules that can be used by students who do not have to learn a single parenthesis of Lisp.

The uniqueness of Xlisp-Stat are its graphical interface building facilities. It will take some climbing of a relatively steep learning curve to use this effectively, but once you are at the top (for somebody who has many other things to do this will take at least a year, for others it will take a couple of months and the usual amounts of coffee) an amazing tool chest is available.

There are also some weak points, which are partly unavoidable and partly easy to repair. Interpreters running their own shell are relatively closed systems, and consequently communication with the outside world (input, output) is somewhat of a problem. Reading data into the system is slow and tedious. This could be improved a great deal by moving some of the input facilities from the Lisp to the C level. There is no spreadsheet data editor, although it would be possible to add one. The help system is rather primitive. And there are no facilities to make publication quality plots. Some of this has been taken care of in the Lisp add-ons discussed below, but the basic facilities should really be part of the core system (which means part of the C code).

For many users it may be a nuisance that generalized additive models, cluster analysis, and many other advanced techniques which are in S-plus or SAS are not in Xlisp-Stat. Again, this merely reflects the fact that the systems have different objectives. SAS is for production, S-plus is for research and production, Xlisp-Stat is for research and teaching. It also is a self-fulfilling prophecy. As the sections below show, extensions can be easily added to Xlisp-Stat at various levels. C and FORTRAN subroutines can be loaded at runtime, and linked dynamically into the system. This makes complete libraries of techniques available with a minimal effort. But somebody has to make the effort. Tierney is maintaining and upgrading the core system, which is a lot of work. Extensions have to come from the users. Once again I will mention GNU Emacs (or tcl/tk or X11) to illustrate that if a free and open tool is adopted by a user community, while the core is maintained by experts, we can create a tool which commercial companies cannot possibly compete with.

Why I love it

In the first place because of the GUI building tools. Second, because of the Lisp language (but that's just taste). Third, because it is free. This may look trivial if you have money, but it means that we can give it to our students – not only to the graduate students, but also to the undergraduates. They can use it not only on our $ 10K elitist workstations, but also on their lowly 386 or Performa machines at home. And fourth, it is open. We can see exactly what it does, because we have the source. We can extend it at compile-time, at link-time, and at run-time level. If we get the latest greatest UNIX box, or a Pentium, or a PowerMac, we recompile to use the power of the new machine. It is easy, the system is eminently portable. Because of these four reasons, I personally don't even think of S or S-plus as an alternative.

How to get it

The complete Xlisp-Stat code distribution can be found in

`~/xlisp/xlisp-stat/dist/`

and binaries for various types of machines are in

`~/xlisp/xlisp-stat/systems/`

There is also quite a bit of Xlisp-Stat code on `statlib` in the directory `xlispstat`. (Note: By the way, `ftp.stat.ucla.edu` is no competitor of `statlib`, it is more specialized and may be more complete in this area.)

We try to keep a collection of everything connected with Xlisp-Stat in the ftp directories. There is always a version of the source code, and there are binaries for selected systems. We actually mirror the `pub/xlispstat` directory from `umnstat.stat.umn.edu`. As the path to the directory shows, there are other lisp systems besides xlisp, and other xlisp extensions besides xlisp-stat. Our server can also be reached via `gopher.stat.ucla.edu` or `www.stat.ucla.edu`.

Code

The most interesting sub-directories on our server are

`~/xlisp/xlisp-stat/code/contrib/`

and

`~/xlisp/xlisp-stat/code/homegrown/.`

In `homegrown` we have Xlisp-Stat code written at UCLA. This includes:

- the SIR package by Ker-Chau Li and his students, for Slicing Inverse Regression;
- the Terrace package by James Hilden-Minton for Hierarchical Linear Modeling with diagnostics;
- the Gifi-gardens package by James Hilden-Minton for MVA with optimal scaling;
- many utilities by Jan de Leeuw for monotone regression, b-splines, the basic APL array operators, multidimensional scaling, nonlinear MVA, FORTRAN-type formatted input, log-linear modeling;
- stem-and-leaf-plots by David A. Betz;
- bootstrap/jackknife algorithms by Jan de Leeuw. This includes bootstrap-t, percentile, abc, bca confidence intervals, bootstrap and jackknife standard error, bootstrap and jackknife bias estimation, bootstrap prediction error estimation. The bootstrap algorithms for regression are written as methods for regression-model-proto. There is also a graphical bootstrap demo. And there is code for permutation and bootstrap hypothesis testing. There are also versions of Randy Sitter's algorithms for bootstrapping in survey situations with clustered and stratified random samples (without replacement).
- Didactic Modules by Jan de Leeuw and Jason Bond. Moving the regression line, moving points in regression scatterplots, the central limit theorem.
- UCLAData. Evolving package for Introductory Data Analysis teaching by Jan de Leeuw.
- Help. Evolving graphical help system for Xlisp programmers and Users by Jan de Leeuw and Jason Bond.

In `contrib` we have xlisp-stat code written by others. This includes everything on statlib, and then some. We mention

- axis: Introductory statistics package by Robert Stine (Wharton).
- bpois: Baysian Poisson Regression using the Gibbs Sampler by Hani Doss (OSU) and B. Narasimhan (PSU).
- cw: Dynamic Graphics for Regression, by Dennis Cook and Sandy Weisberg (Minn). Called dyndiag on statlib. This has been superseded by the much-improved code that comes with their recent book.
- eltoy: Introductory Statistics and Bayesian Elicitation, by Russell Almond (U Wash).
- expsurv: Exploratory Survival Analysis by K. Neely Atkinson (U Texas).
- hasse: F tests and ANOVA tables for factorial designs by Philip Iversen (Iowa State).
- jflisp: Three packages for Model Selection in Regression Analysis by Julian Faraway (Univ. Michigan).
- naras: Code contributed by B. Narasimhan (see also bpois). This includes utility code for sliders and boxplots, as well as packages to demonstrate Markov chains, and to fit kernel density estimates. There is also a package for programmers who want to dynamically link arbitrary C functions at runtime.
- plottools: Don't know who wrote it. It provides nice graphical windows to selects symbols and colors.

- puranen: Some demos by Juha Puranen (Univ Helsinki). Demoes Box-Cox and some of the common densities.
- simulation. General purpose simulation tool by Robert Stine (Wharton).
- students: Projects by students in Nancy Reid's course at the U Toronto. Code for repeated measures with missing data, teaching demos, correspondence analysis, Weibull MLE, the general linear hypothesis.
- surv: Code for survival analysis by E. Neely Atkinson and Meg Gelder.
- times: Digital filtering code by Bill Hatch (Coleman Research Corp).
- udina: gnuplot interface to produce beautiful plots from Xlisp-Stat by Jan de Leeuw and Frederic Udina. kde kernel density estimation system by Frederic Udina.
- xlisp-to-s: Interface from Xlisp-Stat to something called S or S-plus or something. By Steve McKinney (U Wash)
- young. Vista: Visual Statistics System by Forrest Young (UNC).

Some of these packages are just snippets, some are major research efforts. Some have been abandoned in mid-stream, others are still being developed. I know of quite a few projects that are going on that we can hopefully add in the near future.

If you have code to contribute, no matter how incomplete, imperfect, tentative, please send it to `deleeuw@stat.ucla.edu`.

Conclusion

Any knowledgeable instructor can make Xlisp-Stat into the ideal tool for teaching statistics at the graduate level. With a considerable investment from many persons, we can make it into the ideal toolkit for developing introductory statistics teaching modules. With another such investment, we can make it the ideal tool for statistics research (and small-scale production). It is imperative that more statisticians start using Xlisp-Stat in the next few years, because otherwise it will either wither and die or go commercial and freeze. This would be a major loss for the statistics landscape.

Jan de Leeuw
Statistics Program
UCLA
`deleeuw@stat.ucla.edu`

DEPARTMENTAL COMPUTING

Establishing a Server

by Michael Conlon

Should your department have a "server" — a computer that performs some collection of tasks for other computers in the department? What will be gained? How difficult is a server to maintain? If you already have a server are you getting the most out of it? Do you offer the right mix of services to your "clients" — the computers that receive services?

Benefits of a server

As networked desktop computers become more common, the role of the server continues to grow. A server (regardless of platform—Mac, PC or UNIX) can be expected to offer one or more of the following services:

Print sharing

In some configurations, certain computers act as "print servers." Other computers rely on these machines for printing across the network. The benefit is reduced investment in printers, and a potential to support multiple types of printing — laser printers, high volume printers, large-format printers, color printers. To get access to a variety of print devices, the client computers are connected to a network and one or more print server computers connect to print devices. It is also possible for many types of printers to be put directly on the network without a print server computer.

Backup

A server typically is maintained by a system administrator. That person's job functions should include backing up the disk space on the server. The server may also remotely backup file space on client machines across the network. This is a tremendous improvement over the "everyone for themselves" model of backup in which some people do a better job than others. Intelligent backup systems pool hardware resources by only having tape drives on the servers and only backing up files that have changed (incremental backups).

File sharing

Disk space may be available on the server for individuals to use. This space may be accessible by other users, using logon and permission systems to enable users to control who they share files with. Such arrangements are a great improvement over file sharing by file copying