Emacs Speaks Statistics: A multi-platform, multi-package development environment for statistical analysis

anonymous

October 7, 2002

Abstract

Computer programming is an important component of statistics research and data analysis. It is a necessary skill for using sophisticated statistical packages and for writing custom scripts and software to perform data analysis using modern statistical methods. Emacs Speaks Statistics (ESS) provides an intelligent and consistent interface between the user and statistics software. ESS interfaces with SAS, S-PLUS, R, and other statistics packages under the Unix, Microsoft Windows, and Apple Mac operating systems. ESS extends the Emacs text editor to streamline the use and creation of statistical software. ESS understands the syntax for numerous data analysis languages, provides consistent display and editing features across packages, and assists in the interactive or batch execution of statements by statistics packages. We describe in detail the features which ESS provides for increasing statistical programming efficiency.

Keywords: Data Analysis, Programming, S, SAS, S-PLUS, R, XLISPSTAT, STATA, BUGS, Open Source Software, Cross-platform User Interface.

1 Introduction

Statistical research activities, particularly data analysis and communication, involve computing. This is exemplified by the idea that many daily statistical activities can be considered to be *programming with data* (Chambers, 1998). The user interface, which maps user behaviors into instructions to the computer, plays a central role in facilitating these tasks. A familiar, coherent, and well-understood set of input behaviors can increase the efficiency of statistical practice. This paper introduces Emacs Speaks

Statistics (ESS) (Rossini et al., 2002), a software package built upon the Emacs text editor which provides a common interface to a variety of statistical packages on the most common computing platforms.

Statistical package interfaces can generally be placed into 2 categories. The first and older approach is to provide a command-line or batch interface. It has been claimed that these textual interfaces, which provide extensive control over the data analysis procedures being performed, is probably the best interface for auditable research, and can facilitate reproducible research (Schwab et al., 1997). The second approach is to provide a graphical user interface (GUI); this approach either partially or completely replaces the command-line with the use of toolbars and menus along with dialog boxes for performing statistical procedures. Data display is provided through the use of a spreadsheet for entry and modification. Some GUI-based packages can also accept scripts as input for regenerating statistical analyses. ESS provides a middle ground, with a focus on writing data analysis scripts and programming code for statistical computing, while providing tools, sometimes with associated GUI tools such as menus, toolbars, and dialogs, for speeding up the associated programming tasks.

ESS is an interface to statistical packages that provides tools which facilitate both statistical software development and data analysis. ESS provides assistance with both the writing and evaluation of analysis code for many types of statistical packages. ESS currently supports the S family of languages, including S (Becker et al., 1988; Chambers and Hastie, 1992; Chambers, 1998), S-PLUS® (Insightful Corp., 2001), and R (Ihaka and Gentleman, 1996; R Development Core Team, 2002); SAS® (SAS Institute Inc., 1999); STATA (StataCorp, 2000); XLISPSTAT (Tierney, 1990) and its extensions Arc (Cook and Weisberg, 1999) and ViSta (Young et al., 1992); BUGS (MRC Biostatistics Unit, 1996–2001); and Omegahat (Temple Lang, 2000). ESS can be extended to accommodate most statistical packages which provide either an interactive command-line prompt for inputting instructions, or process batch files for instructions.

We discuss how ESS enhances a statistician's daily activities by presenting its features and showing how it facilitates statistical computing. Next, we describe the Emacs text editor, the underlying platform on which ESS is built. We conclude with a short history of the development of ESS and goals for the future.

2 ESS and Statistical Practice

Statistical programming is the writing of data analysis scripts or general computer programs for data analysis and processing. Both forms will be referred to as programs and source code (the textual result) or programming (the activity). Although these programs could be written in a general computer language such as FORTRAN, C/C++, or Java, it usually makes more sense for the the data analyst to work in specialized scripting languages that support common statistical procedures. The statistical languages (for example S-PLUS, R, SAS, STATA, and XLISPSTAT) usually do not include the same range of sophisticated programming tools for writing and debugging code as the general purpose languages.

ESS extends the Emacs text editor to provide a development environment for statistical programming languages. In particular, ESS provides many features which enhance the construction of data analysis scripts and statistical programming. The result is an environment with features targeting the complementary goals of statistical programming and data analysis. It offers a single interface for a variety of statistical computing tasks including interactive data analysis and statistical programming. ESS is able to provide a functional and extensible interface which is uniform and consistent across multiple statistical packages. This is done by adding shortcuts and features for accelerated editing of files as well as by interacting with the particular statistical packages to provide, for example, control of input/output, assistance with evaluation, and specialized display for viewing, navigating, and editing of help and documentation files.

Some features do not require direct interaction with a statistics package; these include textual script editing features such as syntactic fontification and colorization of scripts, detection of balanced and unbalanced parentheses and braces through highlighting, code indentation for readability, commenting out regions, and indexing of files by function. The more interesting features, such as help file parsing and viewing, object and function name completion, evaluation of scripts, and interactive editing of data and objects, require some interaction with the target statistical package either by controlling interactive behavior or by batch processing.

2.1 Editing

Source Code Formatting and Display The task of programming is made easier when language constructs (such as reserved words, function calls, strings, and comments) are visually identifiable and when lines of code are automatically indented to a depth appropriate to their context (e.g., if–then clauses, loops). ESS provides both of these to the programmer by including a description of the syntax of each supported statistical language in the form used by `font-lock-mode` (for a brief discussion of Emacs modes and font-lock-mode, see Section 3). The font selection and the indentation depth are automatically set while the user is typing. There are several options for mapping of colors or fonts to each of the syntactic types. We selected black-and-white font-mapping for display here. On a color terminal we might use purple for the keywords, red for comments, green for matching parens, and inverse-video purple for mismatched parens. Emacs makes default choices of colors and ESS provides several other optional schemes.

Figure 1 shows a black-and-white example of font-locking a complicated S statement. The top panel shows an `if` statement with a long expression in the condition and a multi-line consequence. The keyword `if` is shown in an underlined font, the string `"deltat"` in an italic underlined font. The comments are in an italic font. Everything else is in the standard font. The consequence is indented and the continuations of the consequence are further indented. The matching parentheses are marked by a bold foreground and a shaded background. The cursor is indicated by a solid box. In the bottom panel we replaced the matching parenthesis with an unbalanced bracket. Emacs immediately marks that with the paren-mismatch font, bright purple on a color terminal.

[Figure 1 about here.]

ESS uses the Emacs tools for reformatting code to match particular styles. For S, both common C format styles and locally customized styles can be used to define the indentation level for nested statements, location of open-braces (at the end or at the beginning of a line), indentation offsets for if-then-else constructs, and similar characteristics. Functions exist to reformat blocks of code to match the desired style. Similar functions exist for XLispStat, though good Lisp programming style is better defined and hence more restrictive.

Syntax highlighting can be used to enforce coding standards. Figure 2 illustrates a standard for SAS programming, but coding standards aids could be implemented for other languages such as R or S-PLUS.

[Figure 2 about here.]

Help: Display, Navigation and Editing. ESS provides display and navigation tools for S (R and S-PLUS) and STATA help displays. Displays of help files are done in a separate buffer. These buffers include single-key bookmarks of the main sections of the help file, such as to jump to the function arguments or examples sections. Code in the latter is sent to a running S process easily.

In addition, ESS provides an R documentation mode (`Rd-mode`) which assists in writing help files for R functions, objects, and other topics. `Rd-mode` provides the ability to view and execute code embedded in the help file in the same manner as ESS handles code from any S language source file. It provides syntax highlighting and the ability to submit code directly to a running ESS process, either R or S-PLUS, for evaluation and debugging.

2.2 Interactive Processing

The increased popularity of exploratory data analysis as well as the advent of simple GUIs has made interactive data analysis an important component of statistical practice. ESS uses three different approaches for communicating with statistical packages: inferior process control, which entails redirection of text input and output; peer-to-peer style communication, which is currently accomplished on MS Windows using DDE; and batch submission of whole or partial text files containing analysis scripts. For packages which do not support any of these, the primary use of ESS is as an editing tool, with interaction done using windowing cut-and-paste techniques; this still provides beneficial editing features, some of which extend beyond native editing environments. Examples of this last situation include Windows versions of SAS, Stata, and XLispStat.

Emacs has historically referred to processes under its control as *inferior*, accounting for the name inferior ESS (`iESS`) to denote the mode for interfacing with the statistical package. Figure 3 shows the S language program `ess-demo.s` in the top buffer in `ESS[S]` mode and the executing R process in the bottom buffer `*R*`. The `iESS` major mode of the `*R*` buffer is crafted for command-line editing. This mode remembers and uses the command history, allowing for the recall and searching of previously entered commands. Filename completion for local directories is also available.

[Figure 3 about here.]

Source-level Debugging. ESS facilitates the editing of source code files, sets of commands written for a statistical analysis package, and allows the user to load and error-check small sections of source code into the package. This is done through several mechanisms. First, the presence of unbalanced parentheses or mismatched/unterminated quotes is immediately evident with syntactic highlighting of the source code. Second, functions are provided for simple and consistent execution of user-specified or natural units of the code (function definitions in S or XLISPSTAT, `PROC ... RUN;` sections in SAS). An error-free evaluation lets the user execute the next section of code; if errors arise, the user edits the current unit and re-evaluates. Once the code is verified, an entire buffer, or file, of code can be sent to the package as a unit. This file can also be used as a batch file for routine analysis at a later time. Finally, output from the statistics package is normally captured directly by Emacs and placed into a buffer from where it can be edited and searched. Particular forms of output such as requests for help pages and log-file output can be diverted into special buffers with modes crafted to facilitate reading. These modes include tools for automatically placing the cursor on the first `ERROR`, for example in SAS and S.

Interactive transcripts. A transcript records all commands entered by the analyst and the corresponding text-based responses such as tables and comments generated by the statistics package during an interactive statistical analysis session. Once a transcript file is generated, for example by saving an `iESS` buffer, `transcript-mode` assists with reuse of part or all of the entered commands. ESS understands the transcript's syntax, especially the potential prompt patterns used during the interactive analysis. ESS provides tools to facilitate editing and re-evaluating the commands directly from the saved transcript. This is useful both for demonstration of techniques and for reconstruction and auditing of data analyses. Special ESS functions can "clean" S language transcripts by isolating all input lines and placing them in a new S language source file. Transcript cleaning facilitates the use of an exploratory interactive analysis session to construct functions and batch files for routine analysis of similar data sets.

Cooperation across Multiple Tools. Statistical packages are intended for either general or specialized forms of statistical analyses. The specialized statistical packages can be far more efficient for their intended activities, but this is balanced by their inability to perform a wide range of general statistical functions. Tightly coupled inter-operability between general and specialized packages rarely exists, but such a facility is often desired. For example, a general purpose package such as R does not perform Bayesian analyses as easily as BUGS does. On the other hand, BUGS lacks breadth in the range of analyses and results it can generate. For this reason, BUGS is often distributed with R packages, like the diagnostic packages CODA and BOA, which assist with importing and analyzing the results in R. Another point of contention is the difference in the interfaces between general packages and specialized packages. ESS helps by providing a single point of contact to both tools, though the typical interfaces (interactive for R, batch for BUGS) can be different.

Concurrent Use of Multiple Machines and Operating Systems. It can be useful to have multiple statistical processes running simultaneously, either on a single machine or a variety of machines. This capability assists with large-scale numerical simulations as well as code design and testing across multiple versions of statistical software packages.

ESS provides transparent facilities for editing files and running programs which might reside on numerous remote machines during the same session. The remote machine could be a different platform than the local machine. This is accomplished through the use of Emacs capabilities for transparent access to remote files over a network. This means that the user views, edits, and saves files on a remote machine exactly as if they were on the local machine.

This relaxes the requirement that statistics programs be available on the local machine. ESS provides both transparent editing of files and execution of statistics packages on a remote machine with `iESS[S]` or `iESS[SAS]`. All the editing and interaction features described for the local machine work equally well on the remote machine. The interaction, including all the unique features of working with ESS, appears to the user as if the program were running on the local machine. If the X11™Windowing system is running on the local machine, it is even possible to bring up visual displays and graphics from remote Unix systems onto a local display.

Interactive S family. ESS for S family statistical languages, `iESS[S]`, replaces the S-PLUS Commands window or the R GUI window. In addition to running the S family language process, `iESS[S]` mode provides the same editing features, including syntactic highlighting and string-search, as the editing mode `ESS[S]`. It also provides an interactive history mechanism; transcript recording and editing; and the ability to re-submit the contents of a multi-line command to the executing process with a single keystroke. `iESS[S]` is used with S, S-PLUS, and R on Unix and with Sqpe and R on Windows.

The S-PLUS GUI on Windows can be used as a DDE server. There are two advantages to using even this limited communication with the S-PLUS GUI through ESS. First, through `ESS[S]` mode the user gets the full editing capabilities of Emacs. Second, S language commands are sent from the editing mode `ESS[S]` buffer and from transcript buffers from previous S sessions directly to the GUI Commands window with the same Emacs key sequences as are used with ESS on Unix. Hence the user can work in a powerful editing environment and is protected from the delay and ergonomic challenges of using the mouse for copy and paste operations across windows.

For languages in the S family, ESS provides object-name completion of both user- and system-defined functions and data. ESS can dump and save objects (user- and system-generated) into formatted text files, and reload them (possibly after editing).

Interactive SAS. `iESS[SAS]` is a mode that allows text-based `PROC` by `PROC` interaction with an inferior buffer running an interactive SAS session on either the local or a remote computer. `iESS[SAS]` mode works by redirecting standard input and output from SAS to ESS. Currently, the `iESS[SAS]` mode can run on any computer, but the SAS process it is controlling must be running on a Unix machine. This process is very efficient for dial-up network connections to a remote computer with SAS installed. The resulting interface is similar to the SAS character terminal interface, but with Emacs key sequences.

Interactive BUGS. BUGS software performs Markov Chain Monte Carlo integration. ESS supports interactive processing of BUGS commands.

2.3 Batch File Processing

Batch file processing with statistical analysis packages is a better choice than interactive processing when the execution times are longer than the user is willing to wait as well as for regularly updated statistical reports and figures. ESS provides a means to shorten the debugging cycle for writing code intended for batch evaluation by containing the whole process, both writing and evaluation, within Emacs.

Batch SAS. SAS is a popular choice for processing and analyzing large amounts of data. Interactive SAS is rarely used in these situations due to the length of time involved. Instead, a file containing SAS commands is created and SAS executes these commands in the background, or batch, while the user moves on to other activities.

ESS facilitates SAS batch with `ESS[SAS]`, the mode for files with the `sas` extension. ESS defines SAS syntax so that `font-lock-mode` can highlight statements, procedures, functions, macros, datasets, comments and character string literals in SAS programs. Optionally, the same language features are highlighted in the SAS log with the addition of log notes, warnings and error messages.

For files with the `sas` extension, ESS binds the function keys in `ESS[SAS]` mode to match the definitions used by SAS Display Manager. These definitions are optionally available in all modes. They are particularly useful when viewing SAS log and listing files (with extensions of `log` and `lst` respectively).

Only one function key press is needed to submit a SAS batch process. Other function keys open the SAS program file, the log buffer, and the listing buffers. When accessed in this manner, the SAS log and listing buffers are automatically updated since they may have been appended or over-written by the SAS batch process. In addition, the SAS log is searched for error messages and the error messages, if any, are sequentially displayed with consecutive key presses.

Another function key opens a SAS permanent dataset for editing or viewing. An option is provided so that the tab and return keys operate in typewriter fashion like they do in SAS Display Manager. This option also defines a key to move the cursor to a previous tab-stop and delete any characters between its present position and the tab-stop. This is a SAS Display Manager feature that is not typically available in Emacs.

The SAS batch process runs on the computer where the SAS program resides. This is important because any SAS permanent datasets referenced in a SAS program only exist on the computer running SAS. If the SAS program resides on a remote computer, then the log and listing are also accessed remotely. The net result is that running SAS batch on remote computers is nearly transparent to the ESS user.

Batch BUGS. The BUGS interactive capability is not often used since MCMC analyses can be very time-consuming; hence, most BUGS programs are executed as batch processes. ESS facilitates BUGS batch with `ESS[BUGS]`, the mode for files with the `bug` extension. ESS provides 4 features. First, BUGS syntax is described to allow for proper fontification of statements, distributions, functions, commands and comments in BUGS model files, command files and log files. Second, ESS creates templates for the command file from the model file so that a BUGS batch process can be defined by a single file. Third, ESS provides a BUGS batch script that allows ESS to set BUGS batch parameters. Finally, key sequences are defined to create a command file and submit a BUGS batch process.

Batch S. ESS provides 2 facilities for batch processing of S family language files. The first is to execute the contents of a file using buffer-evaluation. This differs from interactive processing only by the number of commands being evaluated; errors can be found by examining the resulting transcript. The second is the load-source mechanism, which provides a means of jumping to errors in the source file, but doesn't display the evaluated commands in the transcript. These mechanisms provide different tools for debugging the source files.

3 Emacs, the Basis for ESS

Emacs is a mature, powerful, and extensible text editing system which is freely available, under the GNU General Public License (GPL), for a large number of platforms, including most Unix®distributions, Microsoft Windows® and Apple Mac™ OS. There are two open-source implementations of Emacs: GNU Emacs (Free Software Foundation, 2001) and XEmacs (The XEmacs Project, 2001). Emacs shares many features with word processors, and some characteristics with operating systems, including many facilities which go beyond ordinary text editing. More important to our goals, Emacs can control and interact with other programs. We quickly describe features which enable ESS. These are not necessarily unique to Emacs, but it was the first extensible editor to have them all available.

Keyboard and Mouse Input. When Emacs was originally written, character-based terminals were the most advanced method of computer access. Common Emacs commands were mapped to key sequences, creating keyboard shortcuts for convenience. Over the last decade, Emacs has been extended to use graphical windowing systems, such as X11, Microsoft Windows, and Apple Mac OS, which allow additional forms of input, for example using a mouse, and which encourage multiple applications to share a single display. Presently, Emacs is more often used with a GUI, with commands bound to mouse actions, but having commands also associated with key sequences is an important ergonomic and time-saving feature. Emacs menus and toolbars on the display screen allow mouse access to frequently used actions and provide a graphical alternative when the user does not know or can not recall a key sequence; these are also subject to user-customization.

Buffers give Emacs control. Emacs buffers are the interface between the user and computer. They can be considered to be a collection of scratch pads that both the user and computer can read, write, and respond to. The user can simultaneously edit many files and control numerous programs by opening multiple buffers. With disk files, the working copy of the opened file is placed in an Emacs buffer where it can be viewed and edited either by the user or automatically by Emacs or another program under the control of Emacs. Emacs can save a backup of the contents to disk at specified intervals. Emacs presents buffer contents in ways which optimize reading and navigation activities. One example of program control is the embedding of the interactive operating system command line interpreter, called a shell, within Emacs. Variations on this theme are used to control programs such as statistical packages which take input from and provide output to the command-line. The resulting buffers provide a copy of the entire transcript of the interaction, which can be edited and searched while the program executes.

Major and Minor Modes. Emacs capabilities are extended by loading files containing commands and functions written in Emacs Lisp (elisp) (Chassell, 1999), which is a dialect of Lisp (Graham, 1996). Emacs commands can be called interactively by pressing a key sequence mapped to the command or by name. The most important extensions to Emacs take the form of modes, which provide significant and specific enhancements to the user interface behavior. For example, `font-lock-mode` allows Emacs to highlight, with fonts or colors, the syntax of a programming language whose characteristics are described within a major mode like `ESS[S]`. A full description of the role that modes play is beyond the scope of this paper; there is documentation built into Emacs describing both the range of modes, how they work, and how to implement them. The critical point is that this flexible and extendable facility forms the basis for ESS's implementation.

Editing Extensions. Most programming and documentation tasks consist of editing text. These tasks can be enhanced by contextual highlighting and recognition of special reserved words appropriate to the programming language in use. In addition, Emacs also supports folding, outlining, tags, and bookmarks, all of which assist with maneuvering around a file. Emacs shares many features with word processing programs and cooperates with markup-language document preparation systems such as LaTeX, HTML, or XML.

Tracking changes to a text file made by multiple users, potentially in different locations, is the job of source-code control programs. Emacs interacts with standard source-code control programs such as CVS, RCS, and SCCS through minor modes such as `vc-mode`. These source-code control systems facilitate documenting and tracking edits and changes to a file. More importantly, they allow for branching and merging of versions so that material present in an older version of the file can be recovered and inserted into a newer version in a fairly easy manner. This can be important in tracing down the actual source file used to perform a particular prior data analysis.

Comparison of files, two or three drafts of a paper for example, is simplified by `ediff`. An example is shown in Figure 4. The lines that are similar are highlighted in the two buffers, one for each file, and the specific words that mismatch are highlighted in a contrasting color. `ediff` has many tools for working with the differences in files and in entire directories. When combined with the `patch` utility or a source-code control system, it provides the user with the ability to insert, delete or modify only the differing portions of text files. This can be critical for the data analyst who has 2 or more partial versions of an analysis in separate files which need to be appropriately merged.

[Figure 4 about here.]

Emacs has many other important features. Emacs provides file-manager capabilities, such as `dired` (*directory editor*) and `speedbar` (an index into the structure of a file), both of which interface to the computer's directory structure. Emacs stores the complete history of commands issued in an editing session, allowing a flexible and fairly complete undo capability. More importantly, for modes which control processes, the process input history is stored for recall as well as for later editing for printing or re-use. Emacs also includes web browsers, mail/newsgroup readers, and spell checking.

Mechanisms for both open (`ange-ftp` and `EFS` use ftp) and secure (`tramp` uses scp or ssh) remote file access are available. Emacs can also monitor and control remote processes running in a shell buffer.

In addition to being an extremely powerful editor, Emacs also includes capabilities usually found in an operating system. Thus, it provides a strong foundation for constructing an integrated development environment focused on the needs of statisticians. Emacs' power, flexibility, portability, and extensibility make it a solid platform on which to construct a statistical analysis user interface.

4 History of ESS

ESS would never have existed without GNU Emacs, the editing system for which Richard Stallman won a MacArthur Foundation Fellowship in 1990. Emacs is one of the oldest popular editors and has a long history of being a programmer's editor. Many statisticians got their first taste of the power of Emacs with FORTRAN mode which was introduced in 1986. As statisticians' preferences changed from general-purpose languages such as FORTRAN to specialized statistical analysis languages and packages like S and SAS, Emacs modes soon followed.

The ESS environment is built on the open-source projects of many contributors, dating back over 10 years. Doug Bates and Ed Kademan wrote S-mode in 1989 to edit S and S-PLUS files in GNU Emacs. Frank Ritter and Mike Meyer added features, creating version 2. Meyer and David Smith made further contributions, creating version 3. For version 4, David Smith provided significant enhancements to allow for powerful process interaction.

John Sall wrote GNU Emacs macros for SAS source code in 1990. Tom Cook added functions to submit jobs, review listing and log files, and produce basic views of a dataset, thus creating a SAS-mode which was distributed in 1994. A.J. Rossini extended this SAS-mode to work with XEmacs.

Returning to the S languages, we note that in 1994, Rossini extended S-mode to support XEmacs. In 1995, together with extensions written by Martin Mächler, this became version 4.7 and supported S, S-PLUS, and R. Kurt Hornik contributed a mode for editing R documentation files shortly after.

During 1997, Rossini merged S-mode and SAS-mode into a single Emacs package for statistical programming; the product of this marriage was called ESS version 5. Richard M. Heiberger designed the inferior mode for interactive SAS and SAS-mode was further integrated into ESS. Thomas Lumley's

Stata mode, written in 1996, was also folded into ESS. More changes were made to support additional statistical languages, particularly XLISPSTAT.

ESS initially worked only with Unix statistics packages that used standard-input and standard-output for both the command-line interface and batch processing. ESS could not communicate with statistical packages that did not use this protocol. This changed in 1998 when Brian Ripley demonstrated use of the Windows Dynamic Data Exchange (DDE) protocol with ESS. Heiberger then used DDE to provide interactive interfaces for Windows versions of S-PLUS. In 1999, Rodney A. Sparapani and Heiberger implemented SAS batch for ESS, which relies on files rather than standard-input/standard-output, on Unix, Windows and Mac. In 2001, Sparapani added BUGS batch file processing to ESS for Unix and Windows. In 2002, Aki Vehtari contributed BUGS interactive processing to ESS for Unix and Windows.

This history is summarized in Table 1.

[Table 1 about here.]

5 Discussion

ESS provides an enhanced, powerful interface for efficient interactive data analysis and statistical programming. It allows the user complete control over the communications among the files in which the analysis is specified, the statistical process doing the computation, and the output. Because all activities are contained within the same user environment and hence accessed with the same editing and searching concepts and the same key sequences, user efficiency is increased. ESS is completely customizable to satisfy individual desires for interface styles and code formats, and can be easily extended to support other statistical languages and data analysis packages.

References

Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The S Language; A Programming Environment for Data Analysis and Graphics*. Wadsworth & Brooks/Cole, Pacific Grove, 1988.

John M. Chambers. *Programming with Data; A Guide to the S Language*. Springer-Verlag, New York, 1998.

John M. Chambers and Trevor J. Hastie. *Statistical Models in S*. Wadsworth & Brooks/Cole, 1992.

Robert Chassell. *Programming in Emacs Lisp: An Introduction*. Free Software Foundation, `ftp://alpha.gnu.org/gnu/emacs-lisp-intro-2.00.tar.gz`, 2nd edition, 1999.

R. Dennis Cook and Sanford Weisberg. *Applied Regression Including Computing and Graphics*. John Wiley & Sons, August 1999.

Free Software Foundation. *Emacs 21*. `http://www.gnu.org/software/emacs/emacs.html`, 2001.

Paul Graham. *ANSI Common Lisp*. Prentice Hall, 1996.

Richard M. Heiberger. Emacs Speaks Statistics: One interface — many programs. In Kurt Hornik and Friedrich Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*. Technische Universität Wien, Vienna, Austria, 2001a. `http://www.ci.tuwien.ac.at/Conferences/DSC.html`, ISSN 1609-395X.

Richard M. Heiberger. ESS (Emacs Speaks Statistics) as a user interface to SAS. `http://philasug.org/Heiberger/ESS.htm`, 2001b. Presentation to Philadelphia SAS User's Group, November 12, 2001.

Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

Insightful Corp. S-Plus statistical software: Release 6.0, 2001. Seattle, WA: MathSoft.

MRC Biostatistics Unit. *Bayesian Inference Using Gibbs Sampling (BUGS)*. `http://www.mrc-bsu.cam.ac.uk/bugs/welcome.shtml`, 1996–2001.

R Development Core Team. R 1.6.0, October 2002. URL `http://www.R-project.org/`.

A.J. Rossini, Martin Mächler, Kurt Hornik, Richard M. Heiberger, and Rodney A. Sparapani. Emacs Speaks Statistics: A universal interface for statistical analysis. Technical Report 164, University of Washington Biostatistics, 2001. `http://software.biostat.washington.edu/statsoft/ess/ess-techrep.pdf`.

A.J. Rossini, Martin Mächler, Kurt Hornik, Richard M. Heiberger, and Rodney A. Sparapani. *ESS (Emacs Speaks Statistics)*. `http://software.biostat.washington.edu/statsoft/ess/`, 2002. This is the url for downloading the current release of ESS.

SAS Institute Inc. SAS® software version 8, 1999. Cary, NC, USA: SAS Institute Inc.

Matthias Schwab, Martin Karrenbach, and Jon Claerbout. Making scientific computitations reproducible. In *Stanford Exploration Porject, Report 92*, volume 92, pages 317–335. November 12 1997.

StataCorp. Stata statistical software: Release 7.0, 2000. College Station, TX: Stata Corporation.

Duncan Temple Lang. The omegahat environment: New possibilities for statistical computing. *Journal of Computational and Graphical Statistics*, 9(3), September 2000.

The XEmacs Project. *XEmacs 21*. `http://www.xemacs.org`, 2001.

Luke Tierney. *Lisp-Stat: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. John Wiley & Sons, New York, 1990.

Forrest W. Young, Richard A. Faldowski, and Mary M. McFarlane. ViSta: A visual statistics research and development testbed. In *Computing Science and Statistics. Proceedings of the 24rd Symposium on the Interface*, pages 224–233. Interface Foundation of North America (Fairfax Station, VA), 1992.

A Appendix: ESS Resources on the Internet

Latest Version. ESS is constantly in flux. New versions of statistical packages, Emacs and operating systems require new releases of ESS to support them. The latest stable version of ESS can be found on the web at `http://software.biostat.washington.edu/statsoft/ess/`. To get help with problems, send e-mail to `ess-help@stat.math.ethz.ch`. The latest development, hence unstable, version can be obtained by anonymous CVS. First type:

```
cvs -d :pserver:anoncvs@software.biostat.washington.edu:/var/anoncvs login
```

You will be prompted for a password which is "`anoncvs`". Then type:

```
cvs -d :pserver:anoncvs@software.biostat.washington.edu:/var/anoncvs co ess
```

Additional documentation. An expanded version of the present paper is in (Rossini et al., 2001). A general introduction and usage instructions can be found in (Heiberger, 2001a); in addition, one which is more focused on SAS can be found in (Heiberger, 2001b).

The documentation that comes with ESS provides details of its implementation as well as examples of its use. Start with the file `doc/ESS_intro.tex` for an overview and elementary introduction. Complete documentation is in `doc/html/ess.html` or by the `info` system with
`C-h i C-s ESS RET RET`.

List of Figures

1 Emacs detects mismatched parentheses. The top panel shows matching parentheses highlighted in the same color. The bottom panel shows mismatched parentheses, a left parenthesis and a right brace, highlighted in an attention grabbing color. In this black and white rendition, the mismatch is in reverse video. On a color display, the mismatch is in bright purple. 18
2 Enforce coding standards. The standard here is that all `PROC` statements must use the `DATA=datasetname` option. Lines that satisfy the standard are given a shaded background (green on a color screen), lines that don't are displayed in inverse video (red on a color screen). Ambiguous ones are displayed in a lighter shade of inverse video (yellow on a color screen). 19
3 Line-by-line execution of a command file. The cursor is placed on a line in the `ESS[S]` buffer and then with a single key sequence, the line is sent to the `*R*` buffer for execution. The output of the package goes directly to the editable `*R*` buffer. 20
4 Ediff of two versions of a file. Similar lines in the two files are detected. The entire line is highlighted in each file and the differences are highlighted in a contrasting color. . . . 21

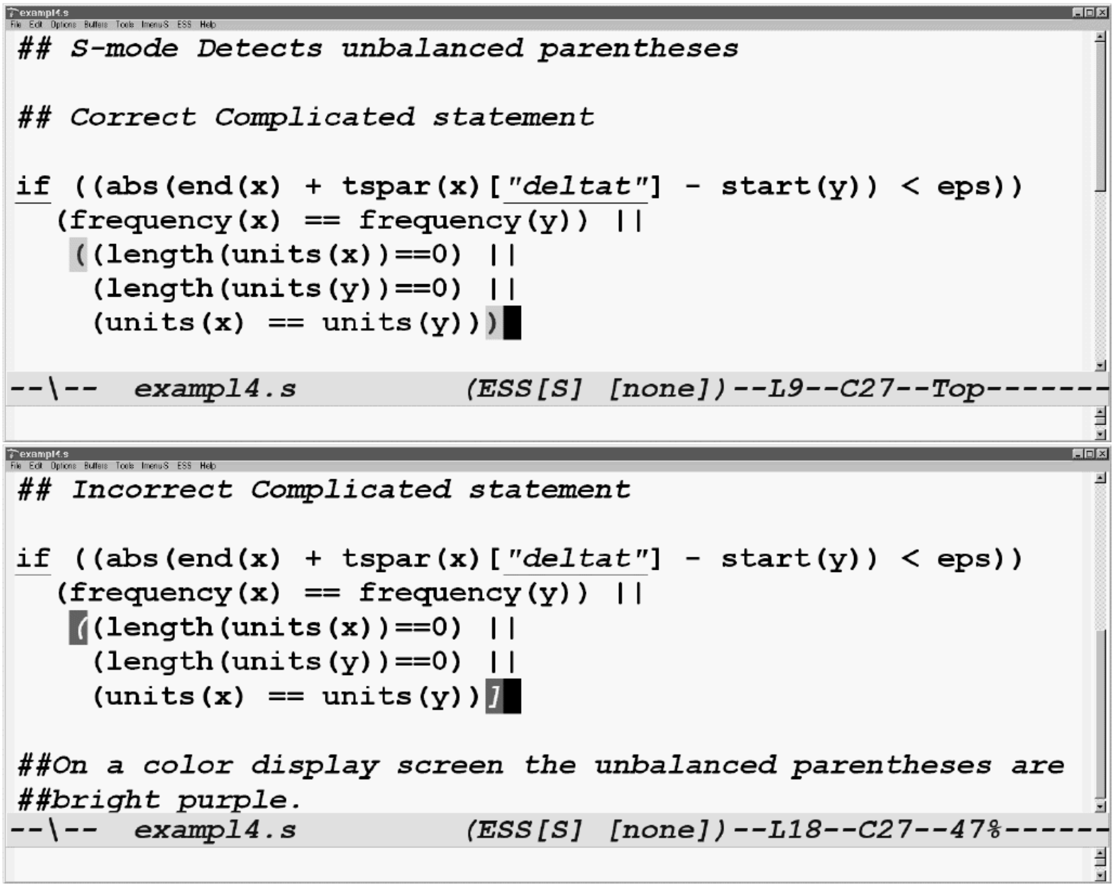

Figure 1: Emacs detects mismatched parentheses. The top panel shows matching parentheses highlighted in the same color. The bottom panel shows mismatched parentheses, a left parenthesis and a right brace, highlighted in an attention grabbing color. In this black and white rendition, the mismatch is in reverse video. On a color display, the mismatch is in bright purple.

Figure 2: Enforce coding standards. The standard here is that all `PROC` statements must use the `DATA=datasetname` option. Lines that satisfy the standard are given a shaded background (green on a color screen), lines that don't are displayed in inverse video (red on a color screen). Ambiguous ones are displayed in a lighter shade of inverse video (yellow on a color screen).

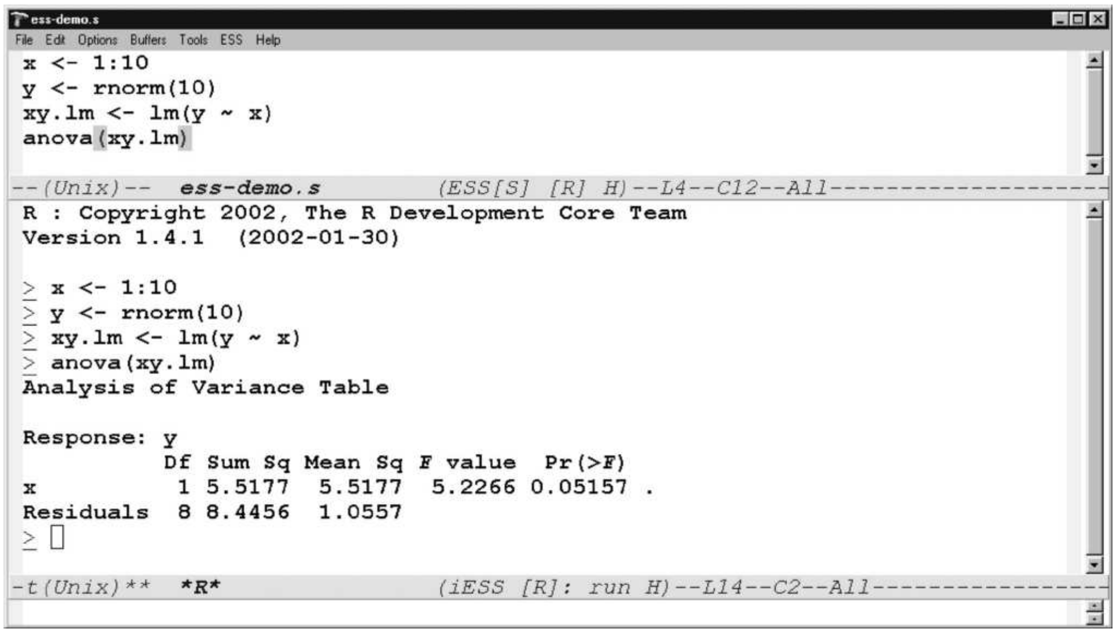

Figure 3: Line-by-line execution of a command file. The cursor is placed on a line in the `ESS[S]` buffer and then with a single key sequence, the line is sent to the `*R*` buffer for execution. The output of the package goes directly to the editable `*R*` buffer.

Figure 4: Ediff of two versions of a file. Similar lines in the two files are detected. The entire line is highlighted in each file and the differences are highlighted in a contrasting color.

List of Tables

1 History and Ancestors of ESS 23

Year				
	S-mode		SAS-mode	
1989	v.1	(GNU Emacs, Unix, S/S+)		
1990				(GNU Emacs, Unix, SAS editing)
1991	v.2	(GNU Emacs, Unix, S/S+)		
1993	v.3	(GNU Emacs, Unix, S/S+)		
1994	v.4	(GNU Emacs/XEmacs, Unix, S/S+)	v.1	(GNU Emacs, Unix, SAS batch)
1995	v.4.7	(GNU Emacs/XEmacs, Unix, S/S+/R)	v.2	(GNU Emacs/XEmacs, Unix, SAS batch)
	Emacs Speaks Statistics (ESS)			
	Emacs, Operating Systems		Additional Functionality	
1997	v.5.0	(GNU Emacs/XEmacs, Unix)		Stata, XLispStat, SAS interactive
1998	v.5.1.1	(GNU Emacs/XEmacs, Unix/Windows)		S+elsewhere; Windows: S+/R
1999	v.5.1.10	(GNU Emacs/XEmacs, Unix/Windows/Mac)		SAS batch; Omegahat
2001	v.5.1.19	(GNU Emacs/XEmacs, Unix/Windows/Mac)		Unix/DOS: BUGS batch; Mac: R
2002	v.5.2.0	(GNU Emacs/XEmacs, Unix/Windows/Mac)		Unix/DOS: BUGS interactive

Table 1: History and Ancestors of ESS