DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria
http://www.ci.tuwien.ac.at/Conferences/DSC-2001
K. Hornik & F. Leisch (eds.) *ISSN 1609-395X*

Emacs Speaks Statistics: One Interface — Many Programs

Richard M. Heiberger*

Abstract

Emacs Speaks Statistics (ESS) is a user interface for developing statistical applications and performing data analysis using any of several powerful statistical programming languages: currently, S, S-Plus, R, SAS, XLispStat, Stata. ESS provides a common editing environment, tailored to the grammar of each statistical language, and similar execution environments, in which the statistical program is run under the control of Emacs. The paper discusses the capabilities and advantages of using ESS as the primary interface for statistical programming, and the design issues addressed in providing a common interface for differently structured languages on different computational platforms.

1 Introduction

Complex statistical analyses require multiple computational tools, each targeted for a particular statistical analysis. The tools are usually designed with the assumption of a specific style of interaction with the software. When tools with conflicting styles are chosen, the interference can partially negate the efficiency gain of using appropriate tools. ESS (Emacs Speaks Statistics) [7] provides a single interface to most of the tools that a statistician is likely to use and therefore mitigates many of the incompatibility issues. ESS provides a superior editor, an interface to statistical processes, and additional tools which can be useful for both statistical software development and data analysis. ESS works with common statistical software including the S family of languages including S [1, 3, 2], S-PLUS [6], and R [5]; XLISPSTAT

*Temple University, Philadelphia, PA, USA

[12] including its extensions ARC [4] and VISTA [14]; SAS [8]; STATA [10]; Omegahat [11]; and can be extended in at least a limited way to most statistical packages which can be controlled from a command-line.

Conflicts can be simple, such as different keystrokes for editing tasks such as copy and paste, beginning of line, highlighting regions. The simple conflicts are sometimes the most aggravating because our fingers are trained for one convention and we are continually interfering with that training. ESS provides a single keyboard interface for editing files and immediately solves that specific problem.

Conflicts can be complex, for example, the coordination of several files of data and code in a single analysis. ESS, as part of Emacs, incorporates version code control systems, ftp and other protocols for transferring files across machines, a smooth interface to documentation systems such as LaTeX and html, and a myriad of housekeeping tasks.

Few statistical packages have identical command-line interfaces; this is distinct from the differences between the actual macro or programming languages. Even dialects of a language, for example R, S-PLUS, and the original S implementation, and different releases of each of these may differ in their interfaces. Code design and testing necessitate working in multiple dialects and versions; this might mean simultaneously accessing programs which run on completely different sets of machines. Reasons for this include verifying behavior on different versions of the same software, test-and-run scenarios where one process is doing long-term processing while the other is doing short-term testing, simulations, running multiple processes on multiple machines from the same place.

ESS provides in a consistent way the interactive features that people doing statistical analysis and designing statistical procedures need and want. ESS provides by far the best environment available at any price and it's free.

Section 2 discusses Statistical User Interfaces. Section 3 discusses features of the ESS development environment. Sections 4 and 5 discuss how this is realized for two different statistical languages, the S family (S, S-PLUS, and R) and SAS. This is followed by the project's history.

2 Statistical User Interfaces

Statistical packages and languages generally provide one or more user interfaces. Most packages have command-line interfaces (CLIs), which derive from text-only teletype machines that were once the standard terminal for accessing computers. Newer software uses graphical user interfaces (GUIs), systems where mouse clicks are used to activate menus and toolbars. With a GUI the screen shows multiple windows to display graphics, to enter and edit text-based commands, and to display a spreadsheet displaying a cases-by-variables view of the data set.

In addition, there have been other one-time implementation interfaces, such as the flow-chart guides, implemented as graph-based interfaces in VISTA [15]; the SAS terminal interface, which divides the terminal window into 3 or more screens; and by the coherent implementation provided by DataDesk [13], which has a tight link between data and graphics.

Command Line Interfaces. The user enters commands one line at a time. The program responds by printing tables to display the results of an analysis or by storing intermediate results. Current packages with CLIs include S-PLUS, R, STATA, and XLISPSTAT. ESS assists command-line interfaces by providing a comprehensive interface layer on top of the command-line.

Graphical User Interfaces. Many packages have adopted a data display descended from the spreadsheet data-table approach. These show a cases-by-variables spreadsheet representation of the data, along with pull-down menus and dialog boxes for data analysis activities. Software that takes this approach includes SPSS, Minitab, S-PLUS 4.x for Windows and 6.x for Unix and Windows, DataDesk, VISTA, and many others. In languages where the GUI is built on top of an underlying command language, for example S-PLUS, ESS can control the behavior of the GUI.

3 ESS

ESS (Emacs Speaks Statistics) is built on top of Emacs to provide a uniform editing and execution environment for statistical analysis. ESS builds on the extremely powerful Emacs editing capabilities and the Emacs ability to communicate directly with a running process. ESS currently supports several interactive statistical programs and interpreted statistical languages. There are various levels of support depending on the capabilities of the program and the needs of users.

ESS strongly supports the S family of languages: S, S-PLUS, and R. SAS is also well supported, but to a lesser extent. STATA and XLISPSTAT (and the XLISPSTAT extensions, ARC and VISTA) are supported with the basic functionality of syntax highlighting and process-interfacing.

3.1 Emacs

Emacs [9] is a mature, powerful, and easily extensible text editing system which is freely available under the GNU General Public License for a large number of platforms including Unix, the Apple Macintosh, and Microsoft Windows. While Emacs is not a word processor, it includes many word processor features. While Emacs is not an operating system, it shares many characteristics with operating systems; most importantly, Emacs can interact with and control other programs either as subprocesses or as cooperating processes.

Editing. Emacs provides facilities that go beyond simple insertion and deletion: viewing two or more files at once; editing formatted text; visual comparison of two similar files; and navigation in units of characters, words, lines, sentences, paragraphs, and pages. Emacs knows the syntax of each programming language. It can provide automatic indentation of programs and highlight with fonts or colors specified syntactic characteristics. Emacs is extensible, meaning new functions for editing text and interacting with programs can easily be written. Emacs provides full customization of the keyboard and mouse, meaning that the new functions can

be bound to appropriate keystrokes. The mouse-based interface, through menus and toolbars, also tries to facilitate the learning of keystroke-based short-cuts. Additional menus can be defined by the user should they be needed.

Most programming and documentation tasks fall under the realm of text editing. In addition, editor behaviors such as folding, outlining, and bookmarks can assist with maneuvering around a file. Type-setting and word-processing, which focus on the presentation of a document, are tasks that are not pure text-editing. Emacs shares many features with word-processing programs and cooperates with document preparation systems such as (LaTeX; SGML, XML, and XSLT; and Noweb).

Controlling Subprocesses. The capabilities can be extended in an orthogonal manner to include other Emacs packages for assisting with documentation as noted above; version control (RCS, CVS, SCCS, PRCS); and remote editing via FTP or secure transport mechanisms such as ssh and scp. Emacs handles the interface to both source code and transcripts contextually, providing syntax highlighting, bookmarking features, interfaces to directory structure, and command-history. Other extensions to Emacs allow it to act as a World-Wide-Web browser, a highly sophisticated mail and news reader, a shell/terminal window with history, and as an interface to other common text-based tools such as spell checking programs. ESS uses these features to run statistical programs under the control of Emacs.

3.2 ESS Features and Use

ESS extends Emacs to provide a functional, easily extensible and uniform interface for multiple statistical packages.

Syntactic Indentation and Highlighting of Source Code. The ESS interface includes a description of the syntax and grammar of each statistical language it knows about. This gives ESS the ability to edit the programming language code, often more smoothly than with editors distributed with the languages. The process of programming code is enhanced as ESS provides the user with a clear presentation of the code with syntax highlighting to denote assignment, reserved words, presence of strings, and comments. ESS has customizable automatic indentation, with the customization based on the syntactic structure of groups of expressions. ESS knows the structure of transcripts of the executing session, as well as it knows the source language, and provides syntactic highlighting here as well.

Partial Evaluations of Code. Emacs can send individual lines, entire function definitions, marked regions, and whole edited buffers from the window in which the code is displayed for editing to the statistical program/package for execution. Emacs sends the code directly to the running program and receives the printed output back from the program. Direct sending of code is a major improvement over cut-and-paste as it does not require switching buffers or windows for either the sending or receiving of information. The response is received immediately in an editable Emacs buffer.

Source Code Checking. ESS facilitates the editing of source code by providing a means for loading and error-checking of small sections of code for S, XLispStat, and SAS. This allows for one form of source-level debugging.

Interacting with the Process. 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. The output of the package goes directly to an editable text buffer in Emacs. This mode allows for command-line editing and saving history, as well as recalling and searching for previously entered commands. Filename completion is available. ESS provides object-name and function-name completion for the S languages. There is a good interface to the internal help systems for S, XLispStat, and Stata.

Interacting with Statistical Programs on Remote Computers. ESS provides the facility to edit and run programs on remote machines in the same session and with the same simplicity as if they were running on the local machine. The remote machine could be a very different platform than the local machine.

Transcription Editing and Reuse. Once a transcript log is generated, perhaps by saving an '`iESS`' buffer, transcript-mode assists with reuse of part or all of the entered commands. It permits editing and re-evaluating the commands directly from the saved transcript. This is is useful for demonstration of techniques as well as for reconstruction of data analyses. ESS can "clean" transcripts from S languages back to source code by isolating all input lines into an input file.

Help Files. ESS provides an interface for reading and displaying help files in the S language. In addition it provides an interface for writing help files for R. ESS help mode provides the ability to view and execute embedded S/R source code directly from the help file in the same manner as ESS handles code from a source file.

4 Using ESS with the S Family of Languages

ESS originated as S-mode and has historically provided strong support for the S languages. This section provides examples for editing files, communicating with the interactive S process, editing transcripts, and getting help.

4.1 Editing Files

`ESS[S]` is the mode for editing S language files. This mode handles proper indenting of multi-line code and functions (automatically when the code is entered and under user control while editing); color and font highlighting based on syntax; the ability to send the contents of an entire buffer, a highlighted region, an S function, or a single line to an inferior S process; the ability to switch between several inferior S processes (on the same or different machines); the ability to request help from an S

process for variables and functions, and to have the help results sent into a separate buffer; and completion of object names and file names.

`ESS[S]` mode is automatically turned on for files with the extensions '`.R`', '`.r`', '`.S`', '`.s`', '`.ssc`', '`.q`'. An inferior process must be running to take advantage of the interactive features such as object- and file-name completion and help.

4.2 Inferior ESS Processes

`iESS[S]` is the mode for interfacing with active S statistical processes. In effect, this mode replaces the S-PLUS or R Commands window with an active buffer running inside Emacs. This mode provides an interactive history mechanism; transcript recording and editing; the ability to resubmit the contents of a multi-line command to the executing process with a single keystroke. It also provides the same editing features as the editing mode `ESS[S]`. The statistical processes S and R are started inside an Emacs buffer by using the Emacs function call `M-x S` or `M-x R`.

`iESS[S]` uses standard input/output to communicate with the S program, except with the GUI interface for S-PLUS 4 and 6 for Windows where standard input/output is not available. ESS currently provides two options on Windows. ESS can use the DDE (Dynamic Data Exchange) protocol to communicate with the GUI. The editing features, including the sending of commands to S-PLUS, are as described above. The printed output does not come back to an Emacs buffer, but instead appears in the S-PLUS Commands Window. Transcripts must be physically copied to an Emacs buffer to get the transcript editing features. Full GUI capability is available by switching to the S-PLUS window. Or ESS can directly access Sqpe, the S-PLUS computing engine, using exactly the same procedures as are used for Unix. With this procedure interactive graphics and other GUI features are not available. In the near future, we anticipate merging these approaches for S-PLUS 6 by using the `connections` class of the new S4 engine. We will then have both interactive GUI and transcript recording and editing on the Windows platform.

ESS provides transparent execution of a statistics package on a remote machine from within the local Emacs. We open a telnet, rlogin, or ssh connection to another machine from within the '`*S+elsewhere*`' buffer in the local Emacs, and then run S on the other machine in that buffer. The interaction, including all the unique features of working with ESS, appears to the user exactly the same as if the program were running on the local machine. With X-windows on the local machine, it is possible to display remote visual displays on a Windows local machine.

4.3 Editing and Reusing Transcripts

`ESS Transcript mode` is designed for editing transcripts, constructed for example by saving an '`iESS`' buffer. This mode understands the form of S transcripts; it does color and font highlighting based on syntax; permits resubmitting multi-line commands to an active process buffer; has the ability to request help from an S process for variables and functions; and the ability to switch between processes. The S language transcripts can be "cleaned" by finding all input lines and isolating

them into a stand-alone input file that can be used as input to another S job or as a model for further program design.

4.4 Philosophies for Combining S with ESS

There are two primary philosophies for using ESS. The default is that the source code (the file '`myfile.s`', for example) is real, and objects in S are temporary realizations of the source. A project is constructed by writing S language source code for user defined functions and data objects. One or more text files containing the code are stored in a project directory or directories. The code is read into S for a temporary realization as an object. This approach allows for better portability, where the user might want to use the written code and perform the same or related analyses on different versions of the S language. This approach also permits external version control for S language source code.

The second, deprecated view, is that S objects are real. The text version of the objects (source code, although we can't use that term with this philosophy) is a temporary realization of the objects. ESS is then used to edit, but not save, the text. The text is returned to S to be saved as an object. We strongly discourage this approach. It is a natural result of assuming that the S process will always be the same, an assumption we are not willing to make.

5 Using ESS with SAS

`ESS[SAS]` mode is used for editing SAS files. It is based on the SAS mode by Cook (`ftp://ftp.biostat.wisc.edu/pub/cook/sas-mode/sas.tar.gz`). ESS provides two user interfaces for executing SAS programs. One is similar to batch processing of SAS jobs and the other to the interactive processing provided by the SAS Display Manager. In addition, SAS files can be edited in `ESS[SAS]` mode in Emacs and then pasted into the SAS Display Manager for execution. All three interfaces can be used with SAS running either locally or remotely.

5.1 Editing SAS Files

`ESS[SAS]`, the mode for editing SAS language files, is automatically turned on when editing a file with a '`.sas`' suffix. `ESS[SAS]` mode provides all the flexibility of the Emacs editor for editing and searching in each file and for simultaneously working with multiple SAS files. This mode handles: proper indenting, generated by both `[Tab]` and `[Return]`; color and font choices based on syntax; ability to send the contents of an entire buffer, a highlighted region, or a single line to an inferior SAS process; ability to switch between inferior SAS processes; ability to save and submit a file as a batch SAS process with a single keypress; single keypress navigation of '`.sas`', '`.log`' and '`.lst`' files (the '`.log`' and '`.lst`' files are automatically refreshed).

Configuration. People using ESS and SAS together come from two incompatible starting positions. Some are Emacs users who are adding SAS to their repertoire.

They expect the keyboard to behave as it does in any other Emacs mode and they expect a few new keys to interact with SAS. Others are SAS users who have discovered the power of editing their SAS source files with Emacs. They expect the keyboard to behave as it always has in their previous editor and they expect to gain some more powerful editing and interaction capabilities by using Emacs. The ESS user must configure the function keys and the `[TAB]` key for one of the positions.

Function Keys. Emacs users often customize the function keys for various editing tasks and they expect the keys to continue to behave in their customized way. SAS users who are familiar with the SAS Display Manager would like to use the same function key assignments that SAS provides. ESS therefore provides all possible options: no special function keys in SAS mode, SAS-style function keys restricted to SAS buffers; or SAS-style function keys in all buffers. The SAS-style function keys can use either the SAS Unix or SAS Windows definitions.

The `[TAB]` Key. The default for the `[TAB]` key is automatic indentation according to the definition of the SAS syntax, the same type of functionality that Emacs provides in most programming language modes. The alternate behavior makes the `[TAB]` key behave as it does on non-Emacs terminals, i.e. move the cursor to the next physical tab stop.

5.2 SAS Process Interaction

The ESS methods for interacting with SAS are based on the SAS Display Manager's procedures. Why therefore use ESS rather than the Display Manager? For two reasons: we prefer Emacs editing of source, and we want immediate access through the Emacs search and edit commands to the output in the listing file.

5.2.1 SAS Batch Processing

Batch SAS usage starts a separate instance of SAS each time a file is submitted. Keys `[f3]`–`[f8]` mimic SAS Display Manager keys. The keys are used to save a '`.sas`' file and submit it to SAS, and to switch to the '`.log`' or '`.lst`' file (refreshing if necessary). The default assumption is that SAS is run on the local machine. This can be overridden by changing `ess-sas-submit-method` to point to SAS on a remote machine. Once the connection has been made to a remote machine, the function key and file behavior is identical.

5.2.2 Inferior SAS Processes

`iESS` (inferior ESS) is the mode for interfacing with active statistical programs. A single instance of SAS is started in an '`iESS[SAS]`' buffer. Individual `DATA` and `PROC` steps are sent to this instance of SAS and the results are appended to the '`*log*`' and '`*lst*`' buffers. `iESS` mode works for SAS running on a Unix machine, either locally or remotely. It does not work for SAS running on a local Windows machine because SAS does not provide standard input/output in its Windows versions. Since `iESS`

was designed to emulate the SAS Display Manager behavior over a slow telephone line, there is really no need for it on a Windows machine running SAS locally.

5.2.3 ESS and Display Manager

It is possible to use ESS for editing SAS files with `ESS[SAS]` mode and then pick up the file or a region and drop it directly into the SAS Display Manager Program Editor window. This method gives the user access to interactive SAS graphics. With any of the other Emacs interactions, the user is limited to static graphics, for example by printing to a PostScript device.

6 History

ESS was initially designed as S-mode in 1991 (D. Bates, F. Ritter, et.al.; M. Meyer; D. Smith) for use with S and S-PLUS™; hence, this family of statistical languages currently has the most support. The extension to a language-independent generic interface was prompted by the success of R, and the need for an R-mode. This led to a merger with the SAS-mode (T. Cook), and the refactoring of the S-mode codebase to accommodate multiple languages in a flexible way.

ESS is a remarkable example of how open-source products develop beyond what their initial authors planned. S-mode was originally developed as a simple tool for editing program files for S and S-PLUS. In 1994, A.J. Rossini extended S-mode to support XEmacs, a forked version of Emacs. At the same time, Rossini extended a successful SAS-mode written by Tom Cook to work with XEmacs. In 1995, support for R was merged into a uniform S-mode for Emacs and XEmacs. During 1996 and 1997, Richard M. Heiberger incorporated SAS-mode into ESS and designed the inferior ESS mode for SAS. At the same time Rossini developed a generic means for configuring ESS to accommodate changed as well as new statistical languages.

Beginning in 1998, Heiberger provided interfaces under Windows for S-PLUS 4.x and then S-PLUS 2000 and S-PLUS 6.x. The Windows interface uses the DDE protocol and is based on an example constructed by Brian Ripley.

Beginning in 1998, Rodney Sparapani designed the SAS batch interaction mode. Sparapani and Heiberger developed SAS batch support including function key behavior that follows SAS Institute's function key definitions, developed a more comprehensive and efficient SAS syntax highlighting mechanism, provided automated error message lookup in the log file, and extended the `*ESS-elsewhere*` functionality to include SAS processes,

7 Acknowledgments

This paper is based on joint work with A.J. Rossini, Department of Biostatistics, University of Washington and Fred Hutchinson Cancer Research Center, Seattle, WA, USA; Martin Mächler, Seminar for Statistics, ETH Zurich, Zurich, Switzerland; Kurt Hornik, Technische Universität Wien, Vienna, Austria; and Rodney Sparapani, Medical College of Wisconsin, WI, USA.

References

[1] 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.

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

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

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

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

[6] MathSoft. S-plus statistical software: Release 5.1, 2000. Seattle, WA: MathSoft.

[7] A.J. Rossini, Martin Mächler, Kurt Hornik, Richard M. Heiberger, and Rodney Sparapani. ESS (emacs speaks statistics), 2001. `ess.stat.wisc.edu/pub/ESS/`.

[8] SAS Institute. SAS statistical software: Release 8.0, 2000. Cary, NC: SAS Institute.

[9] Richard M. Stallman. *GNU Emacs Manual, for Version 20.7*. Free Software Foundation, 13th edition, 2000.

[10] StataCorp. Stata statistical software: Release 6.0, 1999. College Station, TX: Stata Corporation.

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

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

[13] Paul F. Velleman and Paul Pratt. A graphical interface for data analysis. *Journal of Statistical Computation and Simulation*, 32:223–228, 1989.

[14] 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.

[15] Forrest W. Young and David J. Lubinsky. Guiding data analysts with visual statistical strategies (disc: P251–260). *Journal of Computational and Graphical Statistics*, 4:229–250, 1995.