An Introduction to ESS + XEmacs for Windows Users of R

John Fox*
McMaster University

Revised: 5 February 2006

1 Why Use ESS + XEmacs?

Emacs is a powerful and widely used programmer's editor. One of the principal attractions of Emacs is its programmability: Emacs can be adapted to provide customized support for programming languages, including such features as delimiter (e.g., parenthesis) matching, syntax highlighting, debugging, and version control. The ESS ("**E**macs **S**peaks **S**tatistics") package provides such customization for several common statistical computing packages and programming environments, including the S family of languages (R and various versions of S-PLUS), SAS, and Stata, among others. The current document introduces the use of ESS with R running under Microsoft Windows.

For some Unix/Linux users, Emacs is more a way of life than an editor: It is possible to do almost everything from within Emacs, including, of course, programming, but also writing documents in mark-up languages such as HTML and LaTeX; reading and writing email; interacting with the Unix shell; web browsing; and so on. I expect that this kind of generalized use of Emacs will not be attractive to most Windows users, who will prefer to use familiar, and specialized, tools for most of these tasks.

There are several versions of the Emacs editor, the two most common of which are GNU Emacs, a product of the Free Software Foundation, and XEmacs, an offshoot of GNU Emacs. Both of these implementations are free and are available for most computing platforms, including Windows. When I wrote the first version of this document, I held the opinion that XEmacs offers certain advantages to Windows users, such as easier installation, and so I chose to explain the use of ESS under this version of the Emacs editor. I understand that these advantages may no longer hold.

As I am writing this, there are several Windows programming editors that have been customized for use with R. The three most advanced customizations are ESS (as explained, for Emacs), R-WinEdt (for the shareware WinEdt editor; see http://www.winedt.com), and Tinn-R (for the free, open-source editor Tinn, see http://www.sciviews.org/Tinn-R).[1] In comparison to WinEdt or Tinn-R, use of ESS entails several advantages and disadvantages:

Advantages of ESS

- The combination of ESS and Emacs provides a very powerful editor, with advanced features customized to the needs of the statistical programmer and data analyst. Of course, to realize this advantage fully, it is necessary to learn to use ESS/Emacs (henceforth, "ESS") beyond the superficial level.
- If you use R on more than one platform (say, both Windows and Linux systems), then you can employ the same user interface for all platforms.

*I'm grateful to Stephen Eglin for corrections to an earlier version of this document.

[1] From version 2.0.0, R for Windows has included a simple script editor that should meet many users' needs for a basic editor, especially when R is used for data analysis rather than programming.

- If you use more than one statistical package supported by ESS (say, R and SAS), then you can employ the same user interface for all packages. Indeed, you can use more than one package (possibly running on more than one computer) simultaneously from the same ESS session.
- You may prefer the standard ESS interface — which splits a main window into two subwindows — to working with separate WinEdt and R windows: The upper XEmacs subwindow displays R source-code files[2], while the lower window shows the input and output for the R process running under XEmacs.
- XEmacs and ESS are free (but so is Tinn-R).

Disadvantages of ESS

- The editing environment provided by ESS will be somewhat unfamiliar to Windows users. To a large extent, however, the configuration file that I supply for ESS will make its behaviour more familiar (see Section 2). As well, I provide some background information pertaining to the basic use of Emacs (see Section 2.3). In sum, ESS has more the feel of a Unix editor than a Windows editor, particularly when it is used beyond the basic level.
- When you run R under ESS, you implicitly run `rterm.exe` rather than `rgui.exe` for the R process. `Rterm.exe` is used because ESS requires simple text output; on the other side of the equation, ESS in effect provides many of the basic services normally provided by the R GUI[3] (graphical user interface), and my configuration file provides menus similar to those in the R GUI. Some features are lost, however, including the ability to use the familiar Windows compiled-HTML help system. HTML help, which displays in your web browser, may be used in place of compiled-HTML help; by default, my configuration file enables HTML help for the current R session.
- There are some unresolved issues, the seriousness of which varies from system to system, associated with running R under ESS on Windows systems (see Section 3.1).

Both WinEdt and ESS require some configuration for use with R under Windows. (Section 3 describes how to install and configure ESS.) In contrast, Tinn-R works out-of-the-box. I expect that many Windows users will prefer to work with WinEdt or Tinn-R, but it is certainly worth giving ESS a try. After all, you have nothing to lose but your time!

2 Basic Use of ESS + Emacs

The purpose of this section is to get you started using R under ESS. For more information about ESS, XEmacs, and Emacs in general, consult the brief annotated bibliography in Section 4.

2.1 Preliminaries

If you have installed and configured XEmacs as I suggest in Section 3, then when you fire up the editor, the screen will appear as in Figure 1. You will see a message at the bottom of the screen (in the "echo area and minibuffer") indicating that the `.Rhistory` file cannot be read from the current directory. Don't worry about the message.

[2] Strictly speaking, the windows show "buffers," some of which are the memory image of files, rather than files themselves. On start-up, the upper window displays the `*scratch*` buffer, which is not associated with any file. By default, the `*scratch*` buffer is not used for editing R code. See Section 2.2 for more information about buffers and files.

[3] "R GUI" may well be misleading, since it does not provide a graphical user interface to the *statistical* capabilities of R. For a basic-statistics GUI, see my `Rcmdr` package, available on CRAN. The `Rcmdr` package does not run reliably under XEmacs/ESS for Windows.

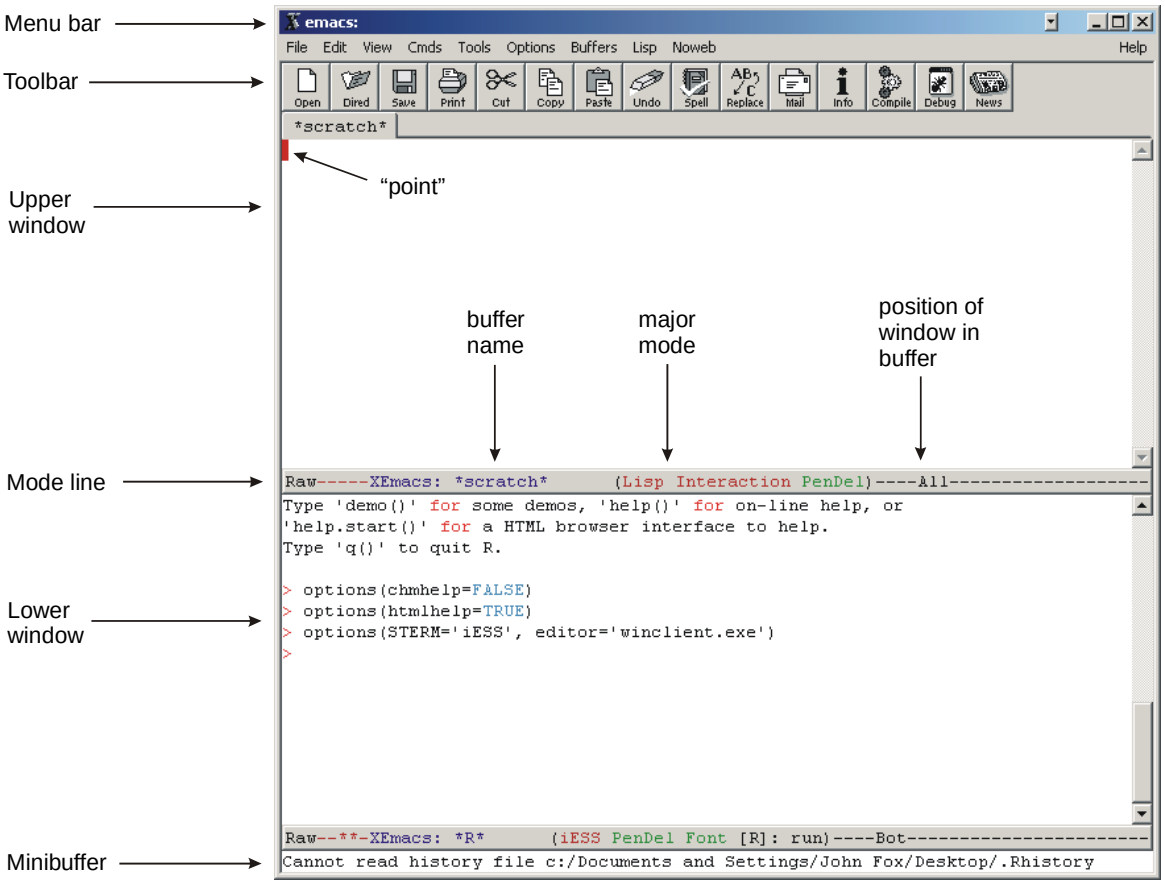

Figure 1: The XEmacs "frame" at start-up, with major components labelled. The `*scratch*` buffer appears in the upper window, the "inferior R process" in the lower window. The "point" is at the start of the `*scratch*` buffer.

- The window labelled "emacs" is called a *frame* in Emacs jargon. Although XEmacs can run in several frames simultaneously, the standard ESS configuration uses a single frame divided into two subwindows, henceforth called simply *windows*.

- At the top of the frame are a *menu bar* and a *toolbar*. These are interface elements familiar to Windows users. Some of the menus and toolbar buttons will also be familiar. The content of the menu bar and toolbar changes with the *major mode* of the *buffer* (see below) in the current window. ESS provides a toolbar customized for use with R[4] (which appears when a file with extension `.R` is in the active buffer). When you start to use ESS, take some time to explore the various menus.

- Each Emacs window displays the contents of a *buffer,* which is a region of the computer's memory containing text.

  - The upper window displays the `*scratch*` buffer. As its name implies, the `*scratch*` buffer is intended to be impermanent, although its contents can be saved to a file (e.g., by pressing the *Save* button in the toolbar). By default, the `*scratch*` buffer is not used to edit R code, so your first step should be to open a file with extension `.R` in the upper window; this can be an existing file or a new one. For example, I pressed the Open button in the toolbar and entered the name of the new file `scratch.R`; the result appears in Figure 2. Note that the upper buffer is now in `ESS[S]` mode (see the discussion of modes below), for entering R commands.
  - The lower window displays the output produced by the *"inferior" R process.* I will resist the temptation to say something humourous, and simply explain that the process is termed "inferior" because it runs under the control of Emacs. You can also type R commands at the > prompt in this buffer.

- The R toolbar provides buttons for interacting with the R process.

  - The first few buttons (*Open*, *Save*, etc.) are familiar and likely do not require explanation.
  - The buttons for R and S-PLUS are for starting R and S-PLUS processes, respectively. Because my configuration file starts up an R process, you can disregard these buttons.[5]
  - The next four buttons can be used to send commands from the upper R script window to the lower R process window — the current line; a selected region; the entire buffer (silently); or a function definition.
  - The last button switches the focus to the lower window.

- Below each window is a *mode line*, which shows status information for the buffer displayed in the corresponding window, including the name of the buffer; the *major mode* (and, if applicable, *minor modes* — see below) of the buffer; and the position of the window in the buffer. For example, the upper window in Figure 1, which displays the `*scratch*` buffer, is in `Lisp Interaction` major mode, while the upper window in Figure 2, which displays the buffer for `scratch.R`, is in `ESS[S]` major mode, and the window shows `All` of the current content of the buffer (which is empty). Similarly, the lower window (in both cases) contains the `*R*` buffer, which is in `iESS` (inferior ESS) major mode, and is currently positioned at the bottom (`Bot`) of the buffer.

[4] Actually, the toolbar is more generally for implementations of S, including R and S-PLUS.
[5] Although ESS can handle several simultaneously executing statistical processes, my configuration cannot.

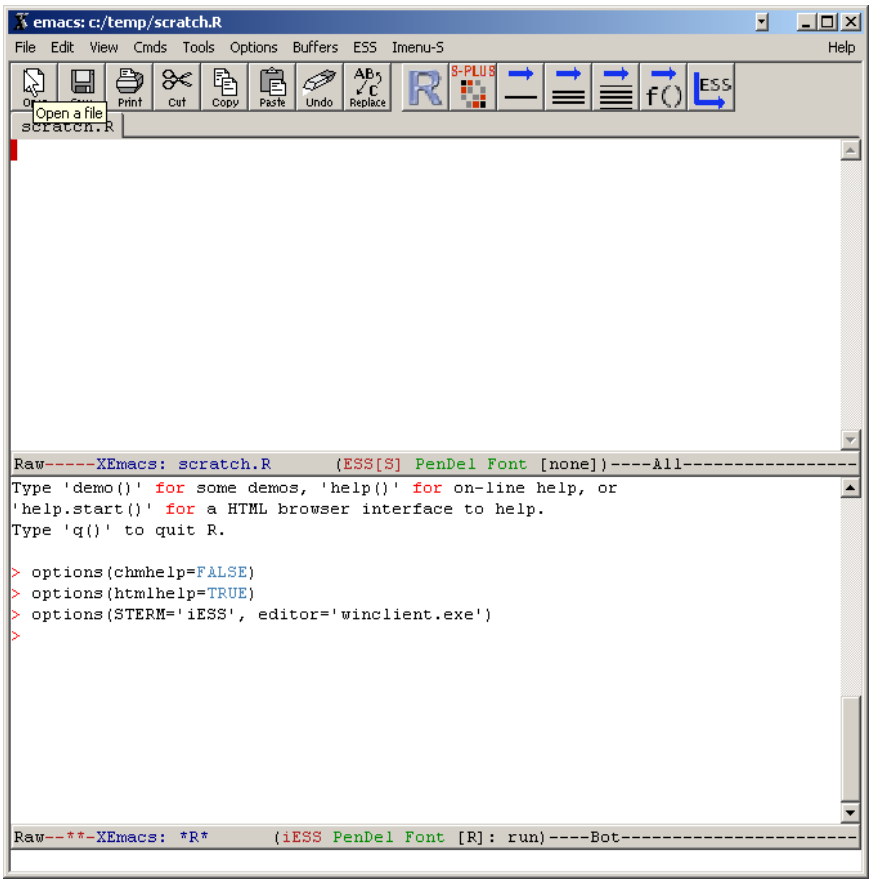

Figure 2: The new file `scratch.R` has been opened in the upper buffer, which is now in `ESS[S]` mode.

- The *point* is the Emacs cursor. It appears in the active window — initially, in the `*scratch*` buffer. By and large, the point behaves in a familiar manner, but it sometimes helps to remember that most activity takes place to the *left* of the point.[6] In particular:
  - When you type, characters are entered to the left of the point, moving existing text to the right.
  - After you mark a region of text — most intuitively by placing the mouse cursor to the left of the first character in the region and dragging over the region (i.e., clicking and holding down the left mouse button) — the point should be positioned immediately to the right of the last character in the region. In other words, if after marking a region the point is over a character, that character is *not* included in the marked region.
- At the bottom of the frame is a small one-line window called the *echo area and minibuffer*. The echo area/minibuffer displays messages (such as the initial message about the missing history file). As well, when you enter Emacs commands using the keyboard, the commands appear in the echo area/minibuffer. Finally, you will occasionally type in the minibuffer when you execute a command.
  - For example, to search for (or search for and replace) text in the current buffer, you may press the *Replace* button in the toolbar. When you do so, Emacs prompts in the echo area/minibuffer for the search and replacement strings; press the *Enter* key after typing each string. Emacs proceeds to find the first matching entry in the buffer below the point. Type `y` to effect the replacement, `n` to keep the current text. Emacs continues to search until the last occurrence of the search string is located. Alternatively, click in a window to terminate the search.
  - Note, as well, that you can search for (or search for and replace) text via the *Edit* menu, or by entering one of many Emacs commands (see the references in Section 4).

2.2 Buffers, Files, Modes, and Windows

As mentioned, Emacs buffers reside in memory and contain text. In many, but not all, cases, the contents of a buffer are read from a disk file; for example, you can load a file into a buffer by using the *Open* button on the toolbar, navigating to and selecting the file in the standard Windows file dialog box that appears. As I did in the preceding section, one can type the name of a nonexistent file; the file is not created until the buffer is saved.

Once read, the contents of the buffer are independent of the contents of the corresponding file, in the sense that changes to the buffer are not automatically reflected in the file. To make changes to the buffer permanent, save the file (e.g., using the *Save* button on the toolbar).

At any given time, each buffer is in one, and only one, *major mode*; for buffers loaded from files, the major mode is determined by the file extension. For example, buffers loaded from files with extensions `.r`, `.R`, `.s`, `.S`, and `.q` are in major mode `ESS[S]`, for S-language source code. The behaviour of Emacs in a particular buffer is governed to a certain extent by the major mode of the buffer. For example, major modes may be associated with specialized menus and key-bindings, and `ESS[S]` mode provides syntax highlighting, delimiter-matching, line indentation, and certain other features appropriate to S-language code.

A buffer may also be in any number of *minor modes,* which provide additional special features. For example, syntax highlighting in ESS is implemented through the `font-lock` minor mode.

[6] I have redefined the *Delete* key in ESS mode so that it deletes the character under (technically, to the right of) the point. In contrast, the *Backspace* key deletes the character to the left of the point. This is the usual Windows convention.

An Emacs window is literally a window into a buffer. Depending upon the size of the buffer, the window may display the entire contents of the buffer or just part of the buffer. In familiar Windows fashion, you can use the scroll bar at the right to move a window up and down within its buffer.

2.3 A Simple Modus Operandi

For purposes of simplicity, I recommend that you proceed as follows, at least initially:

- Use the upper window for buffers containing R source code. For example, in Figure 3, I have loaded the script file for Chapter 1 of Fox (2002) into the upper buffer. Notice that XEmacs provides tabs, immediately below the toolbar, for switching between the `Ch1-script.R` and previously created `scratch.R` buffers; this feature makes it easy to work simultaneously with several R source buffers in the upper window.
- Send commands and function definitions from the upper window to the lower window for execution using the buttons in the toolbar.
- The first time that you try to execute commands in a particular source buffer, ESS prints the following query in the echo area/minibuffer: `Process to load into: R` (see Figure 3). Simply press *Enter* to send this and subsequent commands from the buffer to the R process. The question is posed only once for each source buffer.
- Output appears in the lower window, as shown in Figure 4. Use the scroll bar to move through the output.
- As mentioned, you can also enter commands directly in the lower window by typing them at the > command prompt.
- I have added an *R* submenu to the *ESS* menu, which is displayed when the point is in an R source-code buffer. The *R* submenu includes an item to stop runaway computations along with three submenus, *File*, *Packages*, and *Misc.*, with the following menu items (which perform functions similar to those provided by the menus in the R GUI):[7]
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *File* $\Longrightarrow$ *Source R code ...*: Opens a dialog box to select a file to be "sourced" into the R process.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *File* $\Longrightarrow$ *Save workspace ...*: Save the R workspace to a file.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *File* $\Longrightarrow$ *Load workspace ...*: Load a file containing a saved R workspace.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *File* $\Longrightarrow$ *Change R directory ...*: Change the working directory for the R process (but not for R source files displayed in the upper window).
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Load package ...*: Load an R package, attaching it to the search path.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Install packages from CRAN ...*: Select, download, and install packages from CRAN (requires an active Internet connection).
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Install packages from Bioconductor ...*: Select, download, and install packages from Bioconductor (requires an active Internet connection).
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Install packages from local zip files ...*: Select and install a package for which the zip file resides on your local computer or network.

[7] Some of the menu items may not function correctly on many Windows systems and therefore are disabled by default. See Section 3.1 for more information.

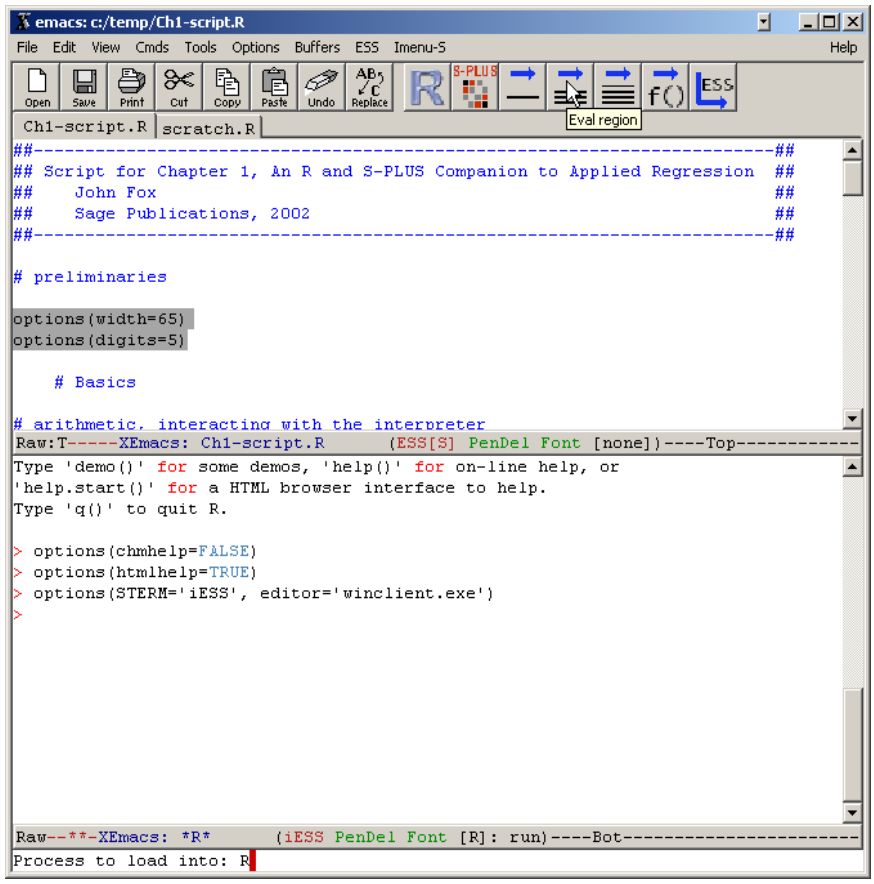

Figure 3: The buffer for the file `Ch1-script.R` is shown in the upper window. The first time that commands are executed, answer the query in the minibuffer by pressing the *Enter* key.

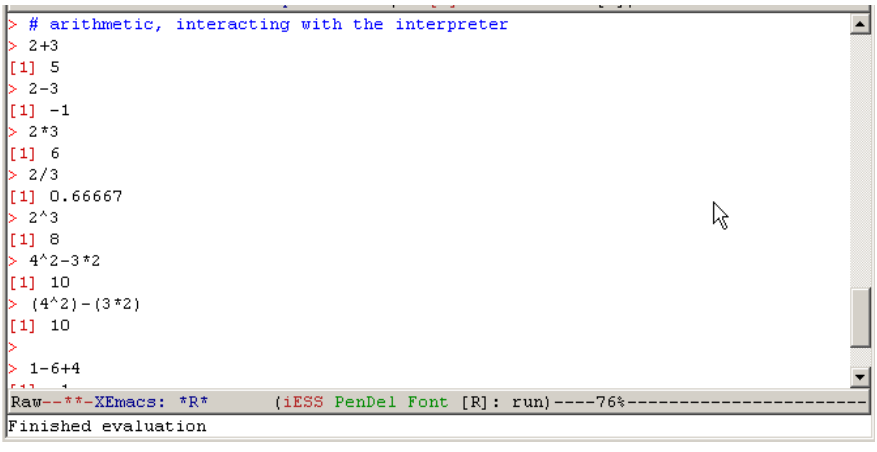

Figure 4: The lower window with R output.

  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Update packages from CRAN*: Look for newer versions of all installed packages, download them from CRAN, and install them (requires an active Internet connection).
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Packages* $\Longrightarrow$ *Update packages from Bioconductor*: Look for newer versions of all installed packages, download them from Bioconductor, and install them (requires an active Internet connection).
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Misc.* $\Longrightarrow$ *Remove all objects*: Delete all objects currently in the R workspace.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Misc.* $\Longrightarrow$ *List objects*: List objects in the R workspace.
  - *ESS* $\Longrightarrow$ *R* $\Longrightarrow$ *Misc.* $\Longrightarrow$ *Display path*: List the R search path.

- I have also added an *Exit R/XEmacs* submenu to the *File* menu, with the following menu items:

  - *File* $\Longrightarrow$ *Exit R/XEmacs* $\Longrightarrow$ *Quit saving R workspace*: The workspace is saved in the current R directory, and you are given an opportunity to save modified buffers (with the exception of the `*scratch*` buffer).
  - *File* $\Longrightarrow$ *Exit R/XEmacs* $\Longrightarrow$ *Just quit*: Terminate the R process without saving the workspace; you are still asked whether or not to save modified buffers.

- I have optionally redefined the key-combinations *Control*-`x`, *Control*-`c`, and *Control*-`v` so that they perform the familiar Windows editing functions (i.e., cut, copy, and paste marked text) in R source-code buffers and in the R-process buffer. Unfortunately, *Control*-`x` and *Control*-`c` are often used as "prefixes" for Emacs multi-key editing commands, and these redefinitions effectively disable those commands in R code and process buffers. The standard Windows behaviour of these keys is therefore turned off by default.[8] You can alternatively use the standard Windows chords *Control-Insert* for "copy," *Shift-Insert* for "paste," and *Shift-Delete* for "cut." In addition, I have made some other small changes meant to make Emacs's behaviour more "Windows-like"; for example, I redefined the *Delete* key so that it deletes the character under the point (actually, to the right of the point), which is the standard Windows convention, rather than to the left of the point. This and other similar small changes seem to me innocuous — for example, the *Backspace* key still deletes to the left — and therefore are turned on by default in my configuration file.[9]

2.4 Obtaining Help

By default, my configuration enables HTML help.

- Entering `help(`*`object`*`)` or `?`*`object`* in the upper window, and pasting this command into the lower window opens your default web browser with the HTML help page for *`object`*.

- Entering `help(`*`object`*`)` or `?`*`object`* directly at the command prompt in the lower window opens the help page for *`object`* in a help buffer in the upper window. This is useful, for example, for pasting examples from the help page into the R process. You can close the help buffer by selecting *Delete Buffer* from the *Buffer* menu.

[8] In any event, standard Emacs control keys are still in effect by default in the `*scratch*` buffer.

[9] If you wish, you can edit my `init.el` file to provide the editing behaviour that you want. Simple instructions are near the beginning of the file (and are discussed in Section 3.1).

- Entering `help.start()` from either window starts your web browser with the initial page of the HTML help system. From this page, you can search the help system; navigate to help pages from all installed packages; and examine HTML versions of the R manuals. A reasonable procedure is to include `utils::help.start()` in your `Rprofile.site` initialization file.

Help on XEmacs is also available, most conveniently through the *Help* menu, at the right of the menu bar. Select *Help* $\Longrightarrow$ *Tutorial* $\Longrightarrow$ *English* for an interactive Emacs tutorial (in English, of course).

2.5 Graphics Output

When working in ESS, you can start a graphics device in the normal manner, explicitly by entering an appropriate command — for example, `windows()` or `trellis.device()` (from the `lattice` package) — or implicitly by calling a high-level plotting function, such a `plot`. The graphics device opens in its own window, and includes the usual menus associated with R graphics devices in Windows (e.g., for saving graphs to files or copying them to the clipboard).[10]

2.6 More on Emacs

Beneath the veneer of menus, dialogs, toolbars, and the mouse, Emacs is fundamentally a keyboard-oriented editor. This keyboard orientation partly reflects the origin of Emacs as an editor for use on character-based terminals (indeed, Emacs provides a primitive windowing system for such terminals), but it also reflects the opinion of most serious Emacs users that it is desirable to access editor commands without removing one's hands from the keyboard.

Almost all Emacs editing commands are implemented as programs written in Emacs Lisp, a dialect of the Lisp programming language (see Cameron, Rosenblatt, and Raymond, 1996: Ch. 13). Most commands are accessible through simple key-combinations. Moreover, the Emacs keyboard is redefinable, and most major editing modes, such as for ESS, include special key definitions. Likewise, menu selections and toolbar buttons work by calling Emacs Lisp commands.

The sources in the annotated bibliography (Section 4) include detailed information on Emacs and ESS editing keys, so I will restrict myself here to orienting information and a few examples. Emacs keyboard commands typically use special key-combinations (or "chords") involving either the *Control* key (hereafter, `C-`) or the *Meta* key (`M-`). There are literally dozens of such chords: To see the "key bindings" in effect in the active buffer, enter the chord `C-h b` (i.e., *Control*-`h` followed by `b`).

- *Control*–key chords are entered by simultaneously pressing the *Control* key and some other key. For example, the chord `C-f` moves the point forward one character (invoking the command `forward-char`). (You can also use the arrow keys to move the point left, right, up, or down.)

- On the PC keyboard, *Meta*-key chords are entered either by simultaneously pressing the *Alt* key and some other key, or by pressing the *Esc* (escape) key and some other key in sequence (i.e., without holding down *Esc*). For example, the chord `M-f` moves the point forward one word (invoking the command `forward-word`).

- Some key commands require a sequence of key presses. For example, `C-x u` (i.e., *Control*-`x` followed by `u`) invokes the `undo` command. Note that this key sequence will not work in R source-code buffers and in the R process buffer if you uese my optional editing-key redefinition of `C-x` (i.e, to cut text). You can, however, switch to the `*scratch*` buffer in the upper window to restore the standard Emacs behaviour for `C-x`.

[10] You may find that some of these menus don't work properly when R is run under ESS.

- Other key commands require that information be entered into the echo area/minibuffer. For example, the chord `C-s` initiates an "incremental search" (via the command `isearch-forward`): As you type successive characters in the echo area/minibuffer, the point forward moves to the first instance of those characters in the buffer.
- The chord `M-x` permits you to enter Emacs commands in the echo area/minibuffer; after the command is complete, press the *Enter* key. For example, `M-x isearch-forward` followed by *Enter* begins an incremental search (and is equivalent to `C-s`).

2.7 Terminating the Session

You may end your ESS session by selecting *Exit XEmacs* from the *File* menu, or by clicking on the standard-Windows ☒ button at the top-right of the *emacs* frame; XEmacs will give you a chance to save any modified buffers. Before you exit, however, you should terminate the R process by entering `q()` at the command prompt in the lower window, or by selecting *Quit S* from the *iESS* menu (which is displayed when the R process buffer has the focus); R will ask you whether you want to save the workspace. Simply killing the R process does not terminate it normally.

As mentioned in Section 2.3, the simplest way to terminate the session properly is to select *Exit R/XEmacs* from the *File* menu.

3 Installing and Configuring XEmacs + ESS

I assume that you want to configure XEmacs primarily for use with ESS and R. If you want to use XEmacs more generally as a programming and text editor, then the configuration that I suggest should probably be adjusted.

An installer for XEmacs can downloaded from the Internet at

<http://www.xemacs.org/Download/win32/#InnoSetup-Download>

Run the installer, installing (unless you have a compelling reason for doing otherwise) XEmacs to the default location, and requesting that a desktop icon be created. These instructions have been checked using the `XEmacs Setup 21.4.19.exe` installer. Earlier version of XEmacs will not work properly with my `init.el` file (described below).

Download the latest version of the zip file for ESS from

<http://ess.r-project.org/downloads/ess/>

At the time of writing, this is `ess-5.2.11.zip`. Unzip the contents of this zip archive in `c:\Program Files\XEmacs\site-packages\` (or modify to reflect where you installed XEmacs), being sure to preserve the directory structure. This should create a subdirectory named `ess-x.y.z`, where `x.y.z` is the ESS version number (e.g., `5.2.11`).

I assume as well that R is installed in the default location,

`c:\Program Files\R\R-`*`x.y.z`*`\`

(where `x.y.z` represents the R version number — e.g., `2.2.1`)[11], that XEmacs is installed in

`c:\Program Files\XEmacs\`

and that there is an icon for XEmacs on the desktop. If this is not the case, adjust the directions below accordingly.

[11] R for Windows is available at <http://cran.r-project.org/bin/windows/base/>, or at one of the R mirror sites <http://cran.r-project.org/mirrors.html>.

1. Determine whether you have a "home" directory and, if so, where it is:

   - In Windows 9x or ME, open an *MS/DOS Prompt* window and enter `set` at the command prompt. Look at the resulting list of environment variables and their values. If the environment variable `HOME` exists, note the directory to which it points. Alternatively, you can enter the command `echo %HOME%`. If `HOME` does not exist, create it by entering the line `set HOME=c:\` in your `autoexec.bat` file. The environment variable will be created when you reboot the computer.[12]
   - In Windows NT, 2000, and XP, open a *Command Prompt* (DOS) window. Enter the command `echo %HOMEDRIVE%%HOMEPATH%` or the command `set HOME` to discover the location of your home directory.[13]

2. Create the subdirectory `.xemacs` in your home directory. Note that (as far as I am aware) *Windows Explorer* does not permit you to rename a directory to a name beginning with a period. Instead, open an *MS/DOS* or *Command Prompt* window; make sure that you are in your home directory; and issue the command `mkdir .xemacs`.

3. Copy my configuration file `init.el` to the `.xemacs` directory. This file can be obtained from

   <http://socserv.socsci.mcmaster.ca/jfox/Books/Companion/ESS/>

   - If you want to use XEmacs for purposes other than running R, you can maintain your present `init.el` file and rename my configuration as, say, `Rinit.el` (still in the `.xemacs` subdirectory of your home directory). Make a second desktop icon for XEmacs (see step 6, below). When you are editing the icon's properties, add `-q -l "~\.xemacs\Rinit.el"` to the end of the *Target* field.[14]

4. If it is not already there, add

   `c:\Program Files\R\R-`*`x.y.z`*`\bin`

   to the Windows search path:

   - In Windows NT, 2000, or XP, go to *System* in the Windows Control Panel. On the *Advanced* tab, press the *Environment Variables ...* button. In the resulting dialog box, click on the *Path* variable under either *User variables* or *System variables*; click the *Edit* button; add

     `;c:\Program Files\R\R-`*`x.y.z`*`\bin`

     to the end of the path; and back out of the dialog boxes by clicking *OK* repeatedly.
   - In Windows 9x or ME, add the following line to the end of your `autoexec.bat` file:

     `path=%path%;"c:\Program Files\R\R-`*`x.y.z`*`\bin"`

     (In either case, remember that *`x.y.z`* represents the R version number.)

5. As an *alternative* to modifying the search path (in step 4 above), edit the `init.el` file in a plain-text editor such as Windows Notepad (or XEmacs!) to tell ESS the location of the `rterm.exe` program; simply set the variable `inferior-R-program-name` as decribed near the beginning of my `init.el` file.

[12] If you prefer, you can define a different directory as `HOME`.

[13] Windows users who have roaming profiles should set the `HOME` environment variable to `%USERPROFILE%\Application Data` , so that they can get their own profiles regardless of where they log on. I'm grateful to Henric Nilsson for pointing this out to me.

[14] I'm grateful to Brian Lopes for this suggestion.

6. Right-click on the *XEmacs* desktop icon; click on *Properties* in the pop-up menu. Specify a *Start in* directory for XEmacs on the *Shortcut* tab — for example, `c:\temp`. An alternative is to create different *XEmacs* icons for different projects, and to start each in the directory for the corresponding project: Begin by right-clicking and dragging the original *XEmacs* icon to make a copy of it. If you always want to open a particular file when XEmacs starts up — for example, a file containing R source code for the project — then add the name of this file to the end of the *Target* field (preceded by one or more spaces, following the closing double-quote around the path to `xemacs.exe`)[15]; rename the icon on the *General* tab to reflect the project.

7. If you wish, edit the `Rprofile.site` file in

   `c:\Program Files\R\R-`*x.y.z*`\etc`

   adding the line

   ```
   utils::help.start()
   ```

Once these configuration steps have been performed, double-clicking on the *XEmacs* icon should start up XEmacs, ESS, and R, much as in Figure 1.

3.1 Further Configuration and Trouble-Shooting

There are unresolved problems in the communication between ESS and the inferior-R process on Windows systems. The problems manifest themselves in graphical-interface elements not functioning correctly; for example:

- Calling up a dialog box (e.g., through a menu selection) may have no effect; issuing an interrupt (via the menus, for example) may return control to the command prompt. In an extreme case, the R process may hang.
- Some menus in graphics-device windows (e.g., the *File* $\Longrightarrow$ *Save as* menu) may not function properly.
- You may find it impossible to exit from the `identify` function via a right-mouse click or menu selection, and may have to close the graphics window instead.
- Dialog boxes may not automatically rise to the front and may be hidden by the XEmacs window. In this event, click on the dialog in the Windows taskbar to bring it to the front.

Partly to cope with ESS/R communications problems, I have provided several configuration options that can be conveniently set by editing my `init.el` configuration file, using a text editor such as Windows Notepad (or XEmacs!). These options appear near the top of the file, which also contains detailed information about each option and recommended settings. I suggest that you read these options before using the configuration file.

4 Brief Annotated Bibliography

1. D. Cameron, B. Rosenblatt, and E. Raymond (1996). *Learning GNU Emacs*. Sebastopol CA: O'Reilly. A broad introduction to Emacs, for those who really want to get serious about the subject. The book includes some information on programming in Emacs Lisp, the language used to extend the capabilities of the editor, but does not specifically cover XEmacs or the use of Emacs on Windows systems.

[15] For example, I use the file `$scratch$.R`, which I normally don't save.

2. J. Fox (2002). *An R and S-PLUS Companion to Applied Regression.* Thousand Oaks CA: Sage. (Referenced in the text.)

3. A. J. Rossini, R. M. Heiberger, K. Hornik, M. Maechler, R. A. Sparapani, and S. J. Eglen (2004). *ESS — Emacs Speaks Statistics, ESS version 5.2.12.* <http://ess.r-project.org/>. This is a clear and well written, but not-quite-complete, manual for ESS, in HTML and PDF form. As far as I know, this manual is the only source that describes the use of ESS in any detail.

4. A. J. Rossini, M. Mächler, K. Hornik, R. M. Heiberger, and R. A. Sparapani, (2001). Emacs Speaks Statistics: A Universal Interface for Statistical Analysis. <http://stat.ethz.ch/ESS/Techrep.pdf>.
   One of several similar unpublished papers and conference presentations that primarily describe the rationale for and general design of ESS. Some information on usage is provided as well.

5. R. Stallman and R. Goyal (1994). *Getting Started With XEmacs.* One of a complete set of manuals for XEmacs, all available at <http://www.xemacs.org/Documentation/index.html>. It is a good place to start learning more about the general use of Emacs, as well as specific information concerning XEmacs.