

Emacs Has No Learning Curve

Emacs and ESS

Paul E. Johnson¹ ²

¹Department of Political Science

²Center for Research Methods and Data Analysis, University of Kansas

2015

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy
- 3 No Learning Curve
- 4 ESS
- 5 Conclusion

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy
- 3 No Learning Curve
- 4 ESS
- 5 Conclusion

Why Use Emacs? Multidimensional Availability

- All Platforms (Windows, Mac, Linux, Unix, Atari,...)
- Across Time: Will Always Exist (because eager fanatics maintain it)
- Free & Open Source

Why Use Emacs? Super Tools

- Emacs has “major modes” for most languages and types of files (C, C++, Lisp, R, SAS, Stata, \LaTeX , English, ...).
- Incredibly powerful text management tools
 - “compare” documents or buffers
 - Regular expression search and replace (even across many files)
 - Copy, paste, insert columns
- Enormous power to edit very large files
- At some point in the future, you may find that the only editor that is capable for a particular project is Emacs. Prepare for that time by using Emacs for other projects as well!

Check the R FAQ

- R FAQ, by Kurt Hornik (Version 2.15.2012-09-19):

6.2 Should I run R from within Emacs?

Yes, definitely. Inferior R mode provides a readline/history mechanism, object name completion, and syntax-based highlighting of the interaction buffer using Font Lock mode, as well as a very convenient interface to the R help system.

Of course, it also integrates nicely with the mechanisms for editing R source using Emacs. One can write code in one Emacs buffer and send whole or parts of it for execution to R; this is helpful for both data analysis and

Check the R FAQ ...

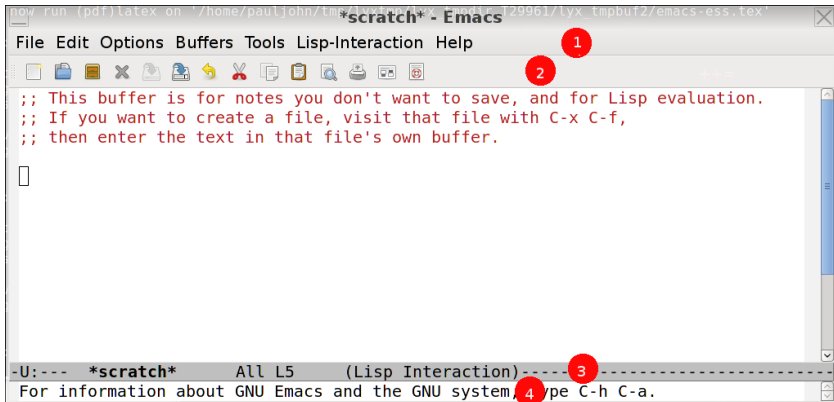
programming. One can also seamlessly integrate with a revision control system, in order to maintain a log of changes in your programs and data, as well as to allow for the retrieval of past versions of the code.

In addition, it allows you to keep a record of your session, which can also be used for error recovery through the use of the transcript mode.

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy**
- 3 No Learning Curve
- 4 ESS
- 5 Conclusion

Start Emacs. Here's What I See



1: Pull Down Menus 2. Button Bar 3. Status Bar 4. Mini-buffer

Terminology

- Frame: That “whole thing”.
- Window: The content display area inside a frame
- Buffer: A chunk of “content,” (letters and words) that can be shown inside a window.
- Check for yourself.
 - Chose File -> Split Window. You can have 2 “windows” showing same “buffer”
 - Choose File -> New Frame
 - Choose Menu Buffers to select a buffer to display in the currently focused Window

Emacs "Major Mode" system

- We want the editor to customize itself to the language we are working with.
- A display of R code should differ from Lisp or Java
- Emacs will guess the mode you want from the file extension (if you open a file)
- Or you can specify the mode while inside the session

Emacs "Major Mode" system

- Emacs automatically chooses the mode for you, from the file name you opened.
- Here's how to manually change the mode, if you want to. (M is the Meta key, usually "Alt")
 - M-x text-mode
 - M-x c-mode
 - M-x tex-mode
 - M-x R-mode

Major Modes try to help with Formatting

- The TAB key should indent a line.
- Position the cursor ANYWHERE in a line and hit tab (Not necessary to be at start of line).
- SHOULD Indent properly.
- Usually, languages want a certain number of spaces, and NOT the literal TAB to be inserted. Emacs handles that.
- The shortcut paragraph re-shape key, M-q, can be configured to work very carefully, to re-shape long lines. It works very well in the major modes for LaTeX, for example. Not so well in ESS at the current time.

Comment Characters

- Programming languages differ in the comment symbol
- Major modes generally know and can insert and arrange them for you.
- Hit TAB to force re-indentation
- Comment and un-comment blocks.
 - 1 Highlight a block of text
 - 2 Hit the keystroke M-;.
 - 3 In an R file, that inserts two pound signs at front of lines
 - 4 If you insert one pound sign, text will get pushed to the right when you hit TAB.
 - 5 If you insert 3 pound signs, text will get pushed flush left

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy
- 3 No Learning Curve**
- 4 ESS
- 5 Conclusion

Understand the History of Emacs

- Emacs was created in the days before
 - mice
 - multi-windowed “desktop” environments
 - Windows or Macintosh computers
- Can still be used in a “keyboard” only world
 - Combinations using (C) Control, (S) Shift (S), and (M) Meta key (usually “Alt”)
 - These are featured in the Emacs tutorial
 - Note the prevalence of “Emacs cheat sheets” floating about on the Internet

Properly Understood, the Emacs Learning Curve is NOT STEEP

- If you have a
 - a mouse, and
 - Windows, Mac, Linux, or Unixthen Emacs will run with pull down menus.
- It is tremendously programmable, most people can use Emacs comfortably even if they don't customize it.
- Settings that seem unusual can usually be changed, and I'll handle most of that for you.

Don't Listen To People Who Want You To Memorize C-x C-g C-h whatever...

- You can learn key-stroke combinations later, if you need them.
- But you don't need most of them now.
- What's easier to remember?
 - Click the Edit Menu, Choose select all, or
 - Type C-x h (Control-x, release both keys, then the letter "h")
 - I use the first way. (Why "h" for select all?)
- What's easier to remember?
 - Hit the "page down" key on the keyboard for a few seconds and watch a file fly by, or
 - Type "M->" (Alt, Shift and the greater-than sign) to go to the bottom

Don't Listen To People Who Want You To Memorize C-x C-g C-h whatever... ..

- I usually just hit “page down” for a moment. Yesterday I had R output that was 150,000 lines and M-> would have been useful.

Example of Previous

- Here is an example from “Experienced Programmers Introduction to Emacs”
<http://weatherall.4all2u.com/work/emacs.htm>

```
Now to a more conventional learning order
C-a   move to beginning of line
C-e   move to end of line
M-<   move to the beginning of the file
M->   move to the end of the tutorial.
C-k   delete rest of line
```

```
<Delete> delete the character just before the
           cursor
C-d      delete the next character after the
           cursor
```

Example of Previous ...

- To do those things, I use the mouse, delete key, and the arrow keys in the obvious ways.
- I don't use Control key combinations that are unique to Emacs very often. If I did, I'd feel helpless when I had to use some other editor. The HOME and END keys work fine. Or arrow keys. I would only need C-a if I had no arrow keys, and no mouse!
- I don't deny the keystrokes might be faster, *if you remember them*.
- New users should not become preoccupied with memorizing C, M and S sequences. Note frequent needs, then learn those keystrokes.

Another Intimidating Example

Useful Emacs bits 'n' bobs

<http://www.insectnation.org/howto/emacs-tips>

Here's a few handy key combinations I wish I'd been told about when I started using emacs:

C-g or ESC ESC ESC: cancel minibrowser session

C-s: search for text

C-r: search for text backwards

C-%: replace text (press space to okay each suggested instance)

C-M-%: replace regex (press space to okay each suggested instance)

C-[space]: place mark (I don't use this...see above re. region selection)

C-w: cut region (std emacs)

Another Intimidating Example ...

```
M-w: copy region (std emacs)
C-k: cut line
C-y: paste (yank)
C-l: recentre buffer window vertically around
      active line
C-x C-f: open (find) file in buffer
C-x C-s: save buffer to file
C-x C-w: save buffer to new file
C-x C-z: stop (pause) emacs process (re-start
        with shell fg)
C-x k: kill buffer
C-x b: change active buffer
C-x 1: display only current buffer window
C-x 2: split buffer window
C-x 0: kill current buffer pane
C-x 4 f: open file in new window
```

Another Intimidating Example ...

```
C-x 4 b: open buffer in new window
C-x 5 2: open new frame
C-x 5 0: kill current frame
C-x 5 f: open file in new frame
C-x 5 b: open buffer in new frame
C-x o: switch between active sub-window
C-x D: enter the very nifty dired-mode
M-x global-font-lock-mode: toggle syntax
    highlighting
M-x perl-mode: a lot of code highlights fairly
    well in Perl mode
M-x goto-line: jump to line
F10: access menus in text mode (via the
    minibuffer)
M-x byte-compile-file .emacs: compile .emacs or
    another elisp file for faster operation
```


Avoid the Emacs Tutorial Until Later

- Emacs provides a nearly complete desktop environment.
- The Emacs tutorial emphasizes memorization of keystrokes that are not truly necessary with current Emacs.
- Instead, *listen to me!* Use my init file. Make Emacs behave in a more familiar way.
- We don't have to sacrifice on any of the other very powerful features of Emacs
- Emacs enthusiasts are able to remember hundreds of key combinations, but I can't.

CUA Mode: Good Fix for Common "Problem"

- People are accustomed to cut and paste keys (from MS or Macintosh):

keystroke	result
C-x	cut
C-c	copy
C-v	paste
C-z	undo

- Emacs default keystrokes are

keystroke	result
C-w	cut ("wipe" in Emacs-speak)
M-w	copy
C-y	paste ("yank" in Emacs-speak)
C-_	undo

CUA Mode: Good Fix for Common "Problem" ...

- Fix. Turn on "CUA mode" in Emacs. Will make C-x, C-v, C-c, C-z do what you expect.
- I turn on CUA mode in my init file, but now it can also be done in a pull down menu. Look under Options.
- Potential problem: Emacs has many keystroke combinations that use C and M and anything we do to "override" them may cause trouble. But most of these CUA bugs have been solved.
- As an example of a bug, try to read the Emacs Tutorial with CUA mode on. (Click the Emacs Help menu, choose Tutorial). The tutorial presumes you can type C-v to go to the next "page" (in the old-fashioned terminal sense of displaying the next "screen sized chunk"). It is necessary to turn off CUA mode to make any progress inside the Emacs tutorial.

CUA Mode: Good Fix for Common "Problem" ...

- Of course, the fact that the first thing in the Emacs tutorial is how to see the "next screen" with C-v illustrates my previous point, that the Emacs tutorial is addressed to a much different user audience than this presentation.

CUA mode Rectangular Selection Super-Power

- Sometimes, you need to highlight a rectangular column of text and copy and paste it.
- This is a secret power of CUA mode! Very convenient
- Here is the CUA way.
 - Move the cursor “top left” of a rectangle you want to select
 - Hit “C-return”
 - Use arrow keys to expand rectangle. Highlight will display a rectangle.
- Cut, copy, paste *just work!*

Customizations I Recommend

- When Emacs starts, it reads a series of configuration files
 - Some are “system-wide”
 - Some are user specific (in the directory `/.emacs.d`)
- I customize the settings thusly.
`http://pj.freefaculty.org/Software/Emacs/init.el`
- An administrator can drop that file in the Emacs install folder under `site-lisp/site.start.d`
- A user should create a folder `.emacs.d` in their home folder and drop `init.el` in there
 - Emacs will create “`/.emacs.d`” folder for you if you use pull down menu “Options -> Save Options”
 - `.emacs.d` is a “hidden” folder, a file manager may not want to show it to you. Try harder.

Here's my Emacs init file (on my laptop 2015-02-04)

```
1  ;; Paul Johnson
2
3  ;; UPDATE: Because Emacs ESS changes made this MUCH easier, my
4  ;; re-work gets shorter :=)
5
6  ;; INSTRUCTIONS. Copy this file into the
7  ;; folder ~/emacs.d, so its name would be ~/emacs.d/init.el
8  ;; or copy it into the Emacs site-start.d folder.
9
10 ;; R USER PREVIEW.
11 ;; Here are my special features related to ESS with R.
12
13 ;; 1. Shift+Enter will send the current line to R, and it will start R
14 ;; if it is not running. ESS 13-05 chose instead CTL+Enter, which I
15 ;; DO NOT want because it conflicts with CUA mode.
16 ;; 2. R will assume the current working directory is the document directory.
17 ;; 3. R runs in its own "frame"
18 ;; 4. Emacs help pops up in its own frame.
19
20
21 ;; JUSTIFICATION. The intention is to make Emacs work more like a
22 ;; "modern" GUI editor.
23 ;; See my companion lecture
24 ;; "Emacs Has No Learning Curve"
25 ;; http://pj.freefaculty.org/guides/Rcourse
26
27
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
28 ;; Paul Johnson <pauljohn@ku.edu>
29 ;; 2012-11-24
30 ;;
31 ;; Conflicts between Emacs-ESS and SAS usage forced me to make
32 ;; some changes. And for no benefit, as SAS mode still not great.
33 ;; I had to cut out a lot of framepop stuff.
34 ;;
35
36
37 ;; 2013-07-10 TODO: Find out if this is still necessary, or for
38 ;; which versions of windows.
39
40 ;; Section I. Windows OS work-arounds
41 ;; 20141222: seems unnecessary now
42 ;;(if (eq system-type 'windows-nt)
43 ;;    (setq use-file-dialog nil))
44 ;; There's been a chronic problem with file selection dialogs on Windows
45 ;; Maybe you comment previous out and see if your Windows is fixed.
46
47
48 ;; Org-mode with R doesn't work without this
49 ;; http://orgmode.org/worg/org-contrib/babel/how-to-use-Org-Babel-for-R.html
50 (custom-set-variables
51 '(ansi-color-names-vector ["#2e3436" "#a40000" "#4e9a06" "#c4a000" "#204a87" "
52                               #5c3566" "#729fcf" "#eeeeec"]))
53 '(cua-auto-tabify-rectangles nil)
54 '(custom-enabled-themes (quote (whiteboard))))
```


Here's my Emacs init file (on my laptop 2015-02-04 ...

```
54 '(org-babel-load-languages (quote ((emacs-lisp . t) (R . t))))
55 '(org-confirm-babel-evaluate nil)
56 '(tab-stop-list (quote (8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 4))))
57
58 ;; Section II. Keyboard and mouse customization
59
60 ;; IIA: make mouse selection work in the usual Mac/Windows way
61
62 (delete-selection-mode t) ; delete seleted text when typing
63
64 ;; In Linux, I see weirdness in Emacs 24 with paste and clipboard. Confusing!
65 ;; http://stackoverflow.com/questions/13036155/
66     how-to-to-combine-emacs-primary-clipboard-copy-and-paste-behavior-on-ms-windows
67
68 (setq select-active-regions t)
69 (global-set-key [mouse-2] 'mouse-yank-primary) ; make mouse middle-click only
70     paste from primary X11 selection, not clipboard and kill ring.
71
72
73 ;; highlight does not alter KILL ring
74 (setq mouse-drag-copy-region nil)
75
76
77 ;; IIB: keyboard customization
78
79 ;; CUA mode is helpful not only for copy and paste, but also C-Enter is
80     rectangle select
81 (cua-mode t) ; windows style binding C-x, C-v, C-c, C-z
```

Here's my Emacs init file (on my laptop 2015-02-04 ...)

```
77 (setq cua-auto-tabify-rectangles nil) ;; Don't tabify after rectangle commands
78 ;;20130717(setq cua-keep-region-after-copy t) ;; Selection remains after C-c
79
80 ;; write line numbers on left of window
81 ;; Caution: if you do this, it makes Emacs much slower!
82 ;;(global-linum-mode 1) ; always show line numbers
83
84
85 ;; Section III. Programming conveniences:
86 (show-paren-mode t) ; light-up matching parens
87 (global-font-lock-mode t) ; turn on syntax highlighting
88 (setq text-mode-hook (quote (turn-on-auto-fill text-mode-hook-identify)))
89
90 ;; Section IV. ESS Emacs Statistics
91
92 ;; start R in current working directory, don't let R ask user
93 (setq ess-ask-for-ess-directory nil)
94
95 ;; ESS 13.05 default C-Ret conflicts with CUA mode rectangular selection.
96 ;; Change shortcut to use Shift-Return
97 ;; Suggested by Vitalie Spinu ESS-help email 2013-05-15
98 ;; Revision suggested 2013-09-30 to co-exist with Windows Emacs
99 ;; and the load order which has ess after user init file.
100 (eval-after-load "ess-mode"
101 '(progn
102   (define-key ess-mode-map [(control return)] nil)
103   (define-key ess-mode-map [(shift return)] 'ess-eval-region-or-line-and-step)))
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
104 )
105
106 ;; create a new frame for each help instance
107 ;; (setq ess-help-own-frame t)
108 ;; If you want all help buffers to go into one frame do:
109 (setq ess-help-own-frame 'one)
110
111 ;; I want the *R* process in its own frame
112 ;; This was a broken feature in ESS, fixed now. Helps massively!
113 (setq inferior-ess-own-frame t)
114 ;;(setq inferior-ess-same-window nil)
115
116
117 ;; In Spring 2012, we noticed ESS SAS mode doesn't work well
118 ;; at all on Windows, that lead to removal of lots of stuff
119 ;; I really liked. Even then, we couldn't get much satisfaction.
120 ;;
121 ;; Following was needed for that, otherwise, it is not needed
122 ;; (load "ess-site")
123 ;; (ess-sas-global-unix-keys)
124
125
126 ;; ; ; ; Section V. Customize the use of Frames. Try to make new content
127 ;; ; ; ; appear in wholly new frames on screen.
128 ;; ; ; ;
129 ;; ; ; ; V.A: Discourage Emacs from splitting "frames", encourage it to pop up
      new
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
130 ;; ;;; frames for new content.
131 ;; ;;; see: http://www.gnu.org/software/emacs/elisp/html\_node/Choosing-Window.html
132 ;; (setq pop-up-frames t)
133 ;; (setq special-display-popup-frame t)
134 (setq split-window-preferred-function nil) ;discourage horizontal splits
135 ;; (setq pop-up-windows nil)
136
137
138 ;; V.C: Make files opened from the menu bar appear in their own
139 ;; frames. This overrides the default menu bar settings. Opening an
140 ;; existing file and creating new one in a new frame are the exact
141 ;; same operations. adapted from Emacs menu-bar.el
142 (defun menu-find-existing ()
143   "Edit the existing file FILENAME."
144   (interactive)
145   (let* ((mustmatch (not (and (fboundp 'x-uses-old-gtk-dialog)
146                               (x-uses-old-gtk-dialog))))
147          (filename (car (find-file-read-args "Find file: " mustmatch))))
148     (if mustmatch
149         (find-file-other-frame filename)
150         (find-file filename)))
151 (define-key menu-bar-file-menu [new-file]
152   '(menu-item "Open/Create" find-file-other-frame
153             :enable (menu-bar-non-minibuffer-window-p)
154             :help "Create a new file"))
155 (define-key menu-bar-file-menu [open-file]
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
156 '(menu-item ,(purecopy "Open File ...") menu-find-existing
157             :enable (menu-bar-non-minibuffer-window-p)
158             :help ,(purecopy "Read an existing file into an Emacs buffer")))
159
160
161 ;; V.D Open directory list in new frame.
162 (define-key menu-bar-file-menu [dired]
163   '(menu-item "Open Directory..." dired-other-frame
164             :help "Read a directory; operate on its files (Dired)"
165             :enable (not (window-minibuffer-p (frame-selected-window
166                                               menu-updating-frame))))))
167
168 ;; Section VI: Miscellaneous convenience
169
170 ;; Remove Emacs "splash screen"
171 ;; http://fuhm.livejournal.com/
172 (defadvice command-line-normalize-file-name
173   (before kill-stupid-startup-screen activate)
174   (setq inhibit-startup-screen t))
175 (setq inhibit-splash-screen t)
176
177 ;; Show file name in title bar
178 ;; http://www.thetechrepo.com/main-articles/549
179 (setq frame-title-format "%b - Emacs")
180
181 ;; I'm right handed, need scroll bar on right (like other programs)
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
182 ;;(setq scroll-bar-mode-explicit t)
183 ;;(set-scroll-bar-mode `right)
184
185 ;; Make Emacs scroll smoothly with down arrow key.
186 ;; 2011-10-14
187 ;; faq 5.45 http://www.gnu.org/s/emacs/emacs-faq.html#
      Modifying-pull_002ddown-menus
188 (setq scroll-conservatively most-positive-fixnum)
189
190 ;; adjust the size of the frames, uncomment this, adjust values
191 ;;(setq default-frame-alist '((width . 90) (height . 65)))
192
193
194 ;; Remember password when connected to remote sites via Tramp
195 ;; http://stackoverflow.com/questions/840279/
      passwords-in-emacs-tramp-mode-editing
196 ;; Emacs "tramp" service (ssh connection) constantly
197 ;; asks for the log in password without this
198 (setq password-cache-expiry nil)
199
200 ;; Section : Emacs shells work better
201 ;; http://snarfed.org/why\_i\_run\_shells\_inside\_emacs
202 (setq ansi-color-for-comint-mode 'filter)
203 (setq comint-prompt-read-only t)
204 (setq comint-scroll-to-bottom-on-input t)
205 (setq comint-scroll-to-bottom-on-output t)
206 (setq comint-move-point-for-output t)
```

Here's my Emacs init file (on my laptop 2015-02-04 ...

```
207  
208  
209 (custom-set-faces  
210   ;; custom-set-faces was added by Custom.  
211   ;; If you edit it by hand, you could mess it up, so be careful.  
212   ;; Your init file should contain only one such instance.  
213   ;; If there is more than one, they won't work right.  
214 )
```

Selection of text

- Default emacs keyboard uses C-space to begin a selection, followed by movement keys.
- A more modern sort of selection can be used.
- This has changed in Emacs-23, so now it behaves in a more usual way. Currently, it seems to work “correctly” if I include only this in my init file.

```
(delete-selection-mode t) ; delete  
selected text when typing
```


Shortcuts I do remember

- There are some keyboard shortcuts I do remember
 - C-s: I search frequently, this is easier than the menu Edit -> Search (C-r searches in reverse)
 - M-%: Query replace
 - C-M-%: Regular-Expression Query replace (can't live without them)
 - M-x: execute commands.
 - C-g : get out of jail free. If you mistakenly start something in the minibuffer, C-g gets out of it.
 - M-q : re-shapes an ill-formed paragraph
 - M-; : for a selected region, will "comment" and "uncomment" all lines.
- But for things that I only do once every 6 months, it is easier to find them in a menu or type them by name.

Using M-x

- M-x tells the minibuffer to get ready for a command
- TAB completion works inside there, so if you remember the first few letters of a command, this is an easy way to do things.
- Example: indent code in a region
 - Highlight some text and type: M-x indent-region
 - After that, the indentation occurs, but Emacs minibuffer reminds me I could have used a short cut:

```
You can run the command '
  indent-region' with C-M-\
```

- Repeat previous using TAB completion feature of minibuffer

Using M-x ...

- Highlight some text and type: M-x indent
Stop typing, hit the tab key once or twice. A set of legal completions pops up, one of which is “indent-region”.
 - Middle-click the one you want to choose it
 - Hit enter
- Example: accidentally open a read only file. Some file saved from R sessions are marked “read only” automatically. Tedious! I want to edit!. Emacs can try to make it writable.
- Run M-x toggle-read-only
 - Emacs minibuffer pops up a reminder

You can run the command “
toggle-read-only” with C-x C-q

- (thanks very much. Which is more memorable. “toggle-read-only” or C-x C-q?)

Using M-x ...

- Maybe the joke's on me here.
 - A reader of the first draft of this presentation pointed out that in the Emacs status bar, there is a click-able thing to toggle "read only."
 - But it is hidden. Look in the left side of the status bar, for two percent signs %%
 - hover the mouse over the first percent sign, and pops up "buffer is read-only. Mouse-1 toggles"
 - It works fine, *as long as you know its there*. (Reminds me of the secret doors in *DOOM*)

Tools -> Compare. Example of a "power feature" in Emacs

- What's the difference between two text files?
- On my website, there is a copy of the Emacs config file I share to people.
`http://pj.freefaculty.org/Software/Emacs/50emacs-ess-ku.el`
- That may not match my laptop. Perhaps I've tried some new settings. My laptop has some features turned on that are commented out in the version on the website.
- I wonder, how far out-of-whack is my website with my reality?
 - 1 Download `50emacs-ess-ku.el`, open in Emacs.
 - 2 Open `~/.emacs.d/init.el` (the current version of the same config file).
 - 3 In Emacs Tools menu, choose Compare, then 2 buffers.

Tools -> Compare. Example of a "power feature" in Emacs

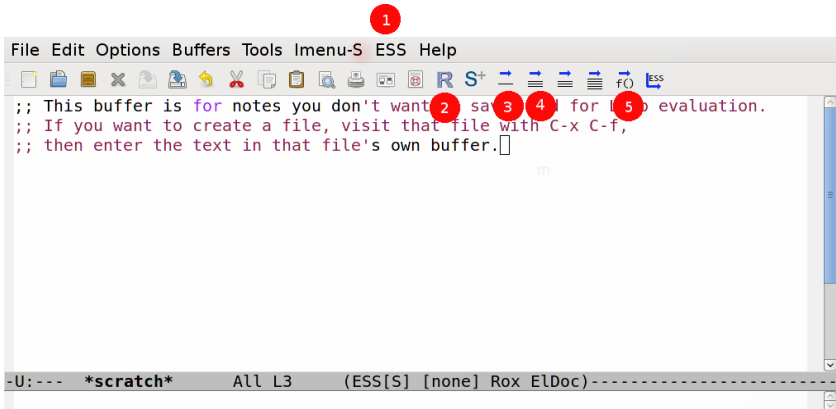
...

- 2 things happen.
 - The different parts of the 2 buffers are color highlighted
 - A small window pops up in which I can Navigate the differences.
 - Type "n" to step to the next difference between the buffers, or
 - "p" for previous.
- Don't forget to hit "q" to close the compare setup, and then it makes you type "yes" in full to escape.

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy
- 3 No Learning Curve
- 4 ESS**
- 5 Conclusion

R-mode is provided by the ESS package

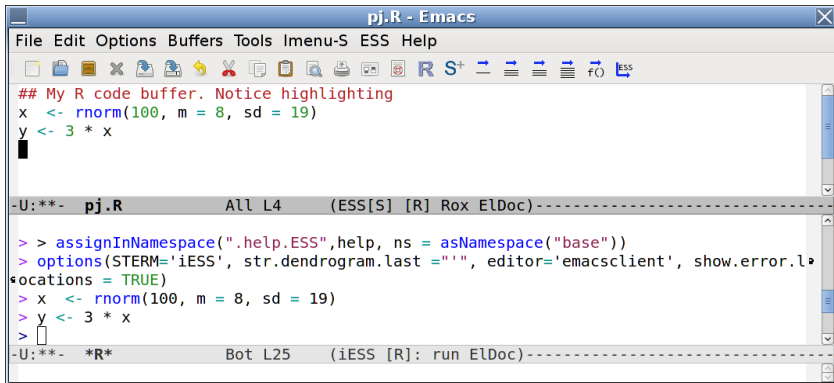


1. ESS menu
2. Blue R starter (same as M-x R)
3. Send one line to R
4. Send selection to R
5. Send current function to R.

Start an R Session Within Emacs

- Hit the big blue R button
- Hopefully, R starts inside a buffer, which Emacs refers to in the buffer list as `*R*`
- If Emacs can't find R in your system, you need to do some configuration work so that the R/bin folder is added to your system path (I have instructions for that in the `crmda` computing documentation)

Some People Prefer "Splits" with Code and *R* like so



The screenshot shows the Emacs editor window titled "pj.R - Emacs". The top pane contains R code with syntax highlighting:

```
## My R code buffer. Notice highlighting
x <- rnorm(100, m = 8, sd = 19)
y <- 3 * x
```

The bottom pane is the ESS terminal, showing the execution of the R code:

```
> > assignInNamespace(".help.ESS",help, ns = asNamespace("base"))
> options(STERM='iESS', str.dendrogram.last = "", editor='emacsclient', show.error.l
ocations = TRUE)
> x <- rnorm(100, m = 8, sd = 19)
> y <- 3 * x
>
```

The terminal status bar indicates the current buffer is "*R*" and the cursor is at line 25 (Bot L25).

The bottom part is the ESS "R inferior mode", a terminal in which R is running

I prefer Separate Frames with R

The image shows two Emacs frames side-by-side. The left frame, titled 'pj.R - Emacs', contains R code with syntax highlighting: a comment line starting with '##' and two lines of code: 'x <- rnorm(100, m = 8, sd = 19)' and 'y <- 3 * x'. The right frame, titled '+R+ - Emacs', displays the R help page for the 'license()' function, including text about warranty, contributors, and help options. The status bars of both frames show the current buffer and mode.

```
## My R code buffer. Notice highlighting
x <- rnorm(100, m = 8, sd = 19)
y <- 3 * x
```

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

```
> > assignInNamespace(".help.ESS",help, ns = asNamespace("base"))
> options(STERM='iESS', str.dendrogram.last="", editor='emacscli
w.error.locations = TRUE)
> x <- rnorm(100, m = 8, sd = 19)
> y <- 3 * x
>
```



Start Emacs within the Desired Working Directory

- Don't Start Emacs from an applications menu: It won't know where to look for files.
- Make a directory structure, and run Emacs from within it. (In Linux, just type `emacs newFile.R` to start).
- On many OS, the easiest way to do that is to copy an R file into the desired directory, and then open that file with Emacs.
- Open an R file, Emacs automatically knows to turn on R-mode (ESS Menu & Buttons)

I'm Willing to Fight For Frames

- I want separate Frames!
- Spawning more Frames is very un-Emacs-like because it goes against the decades-old tradition of Emacs (during which time one could only have one Frame on one terminal).
- Much of my Emacs init file is aimed at forcing Emacs to Start Frames with new content, rather than simply starting new buffers that over-write windows I'm using.
- This is an area where reasonable people can disagree, I'm just telling you what I like. I don't want Emacs to be my window manager, I want the OS to handle that.

Using ESS button bar

- Some of the very eager Emacs-ESS users say they remove the button bar and the menus because they are distracting. I can't imagine...
- I often use the big blue R button  to start R.
 - Sometimes I use M-x R, just to prove I still can. (There was a time when Emacs for Windows had no button bar, so the M-x R was the only way.)
- After starting R within Emacs, run “getwd()” to make sure the working directory is correct.
- The function evaluator  is really handy. While revising a function in code, hit that button and Emacs sends the whole function to *R*.

ESS Menu

ESS Help	
What is this? (beta)	
Load file	C-c C-l
Eval func/para & step	C-c C-c
Enter expression	C-c C-t
Eval and Go	>
ESS Eval	>
Motion...	>
ESS list...	>
ESS Edit	>
Roxygen	>
Start Process	>
Switch Process	C-c C-s
Describe	C-h m
About editing	
Read ESS info	

- The ESS menu teaches you the keyboard shortcuts. You can decide if you like them.
- Eval versus Eval-and-go.
 - Eval commands send instructions to *R*.
 - Eval-and-go send the instructions and transfer the focus to *R*

Roxygen

- Roxygen is a framework for generating R documentation files
- In the “olden days” (last year), an R package would have separate files for R functions and the help files that went with them.
- Problem: programmers found it tedious to maintain the separate help files
- The package roxygen2 (by Hadley Wickham, Peter Danenberg, and Manuel Eugster) addresses that by
 - creating a “language” for writing documentation inside R code files
 - providing functions to translate the result into documentation files

Roxygen ...

- Even if you aren't writing a package, the Roxygen style might be a nice way to prepare your documents.
- Suppose a function is declared like so

```
myFabFunc <- function(x, y, z, a){  
  x * y * z * a  
}
```

- Click on the first line and from the ESS menu, choose Roxygen and "Update/generate template".
- Observe a skeleton is created in which the function can be described

Roxygen ...

```
##' .. content for \description{} (no empty
    lines) ..
##'
##' .. content for \details{} ..
##' @title
##' @param x
##' @param y
##' @param z
##' @param a
##' @return
##' @author pauljohn
myFabFunc <- function(x, y, z, a){
  x * y * z * a
}
```

Roxygen ...

- If we then fill in the sections, we will have pretty good documentation for the function and it will be package-ready (more or less).

ESS Magic I Don't use Menus and Buttons For

- My shortcut: Shift+Enter sends the current line of code to *R*. I think that's convenient.

- Instead, you could :

- Hit the ESS button with one arrow:



- Or type: C-x C-n.

- Edit an existing function object:

- C-c C-d

The mini buffer answers, Object to Edit:

- type the name of some function, such as “lm”
- Emacs opens a buffer called “username.lm.R”, which shows the code for linear models
- We can edit that file, and then re-load it into R with
- C-c C-l

ESS Magic I Don't use Menus and Buttons For ...

- After that, when you run functions that access `lm`, your new version of that function will be used.
- To test this, run `C-c C-d lm` and in the beginning insert `"browser()"`.
- Then run `lm`, or simply `"example(lm)"`. When the computations come to the line where you put in `"browser()"`, the *R* terminal will stop and give you a chance to interact with the inside of that function.
- To me, that is a fun feature. However, I just got an email that indicates it is controversial. (See *ESS- Emacs Speaks Statistics* Version 12.04-4, Section 3.4: Philosophies for using ESS. The current ESS core team believes that the R code files are the "real" stuff on which we should be working, not fiddling about with functions in memory.)

ESS Help Mode

- While reading a help file in Emacs, it is possible to run the example code
- Example. In *R* window, ask for help on lm

```
> ?lm
```

- Emacs will open the help file and the menu at the top will have an ESS-help menu
- In the help file, move the cursor to one of the examples at the end of the file.
- Click the ESS-help menu and choose “Eval Line” or one of the other options.

Transcript Mode. Record Keeping.

- In the perfect world, here is what happens.
 - 1 Save the R code file.
 - 2 Close R. Then Re-start R (so we have a clean canvas).
 - 3 Step through your R code file line-by-line to be sure everything runs properly
 - 4 Save the output file with the extension *.Rout (that means it becomes a “transcript” file).

Some people use the extension *.Rt for transcript files, but for me it doesn't work (don't know why, yet)
- As long as you have the R code and the transcript file, you have all the records you need for future reference.
- What if you are working on an R code file, but you type some commands into the *R* buffer interactively?

Transcript Mode. Record Keeping. ...

- I do that all the time. While I'm testing code, I experiment in the *R* buffer, then copy commands back to the R code file.
- I should save the *R* buffer in a file *.Rout. Let's suppose I did.
- The following problem may arise.
 - I open a new R session
 - I run the R code file that seemed good, but then it fails!
 - How can that be? It worked before?
- Obviously, something I typed interactively in the *R* buffer did not get copied into the code file properly.
- Solution strategy.
 - Open the *.Rout file in Emacs.
 - Emacs recognizes that this is a transcript file

Transcript Mode. Record Keeping. ...

- Emacs won't let me edit that transcript because it is marked "read only", so I either have to copy it to a new file or M-x toggle-read-only.
- Then
 - 1 Select All of the text (either Menu Edit -> Select All, or type "C-x h")
 - 2 In the ESS-trans menu, choose "Clean".
- Viola! That will erase all the output from that file, and leave behind only the commands that were run.
- Along the same lines, from a reviewer of these slides, I received an email about an ESS feature. Since ESS 12.03, there is an R automatic output "scrub and paste." The key sequence is C-u C-u C-y.

Transcript Mode. Record Keeping. ...

- Highlight some material in an *R* output window. It should include commands and output.
- Copy that selection (To copy, use either 1) M-w or 2) C-c from CUA or 3) Edit menu).
- Move the focus to an R code file where you want to paste in only the R commands, no output.
 - If you do the ordinary paste, with C-y, you get all that messy output.
 - But this key sequence will “clean” and paste the resulting R commands.

```
C-u C-u C-y
```

- Note, this requires the traditional C-y (“yank”) to paste, not the CUA C-v.

Outline

- 1 Why Use Emacs?
- 2 Emacs Anatomy
- 3 No Learning Curve
- 4 ESS
- 5 Conclusion**

Emacs in Retrospect

- I think the documentation and publicity that goes with Emacs does it great dis-service.
 - Bill Venables, is credited with the quip: “The first five years of Emacs are the worst; after that, it’s just difficult.”
- When I first tried Emacs, I looked through the tutorial and I thought “this really is trouble.”
- I’ve told people that “Emacs is like Democracy,” adapting the comment that Winston Churchill made to the House of Commons, 1947-11-11, “Democracy is the worst form of government, except for all those other forms that have been tried from time to time.”
- Almost all of the really smart people I know use Emacs.

Do I Really Believe There is No Learning Curve?

- Of course not.
- But I'll get more hits in Google than with my original title, "Emacs Learning Curve 77% Less Steep Than Previously Believed" 😊
- I honestly believe that if people are willing to try Emacs "my way," they can use it comfortably and benefit from many powerful features.
- It is not necessary to memorize a raft of key sequences, even though many Emacs experts do so.

Emacs is Extensible

- I'm not denying the fact that
 - Customizing Emacs is possible, enriching, and frustrating (all at the same time)
- Some parts of Emacs are still difficult for me
 - The help menus are still as confusing to me as ever.
 - I find the Customization menus still very difficult.
- If a person is willing to use Emacs with the init file I provide, using Emacs can be a very reasonable experience.

Useful Websites

Introduction to Emacs: http://cs.earlham.edu/~psg/tutorials/vtvm_emacs/part2.html