Emacs Has No Learning Curve

Emacs and ESS

Paul E. Johnson[1] [2]

[1]Department of Political Science

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

2012

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?

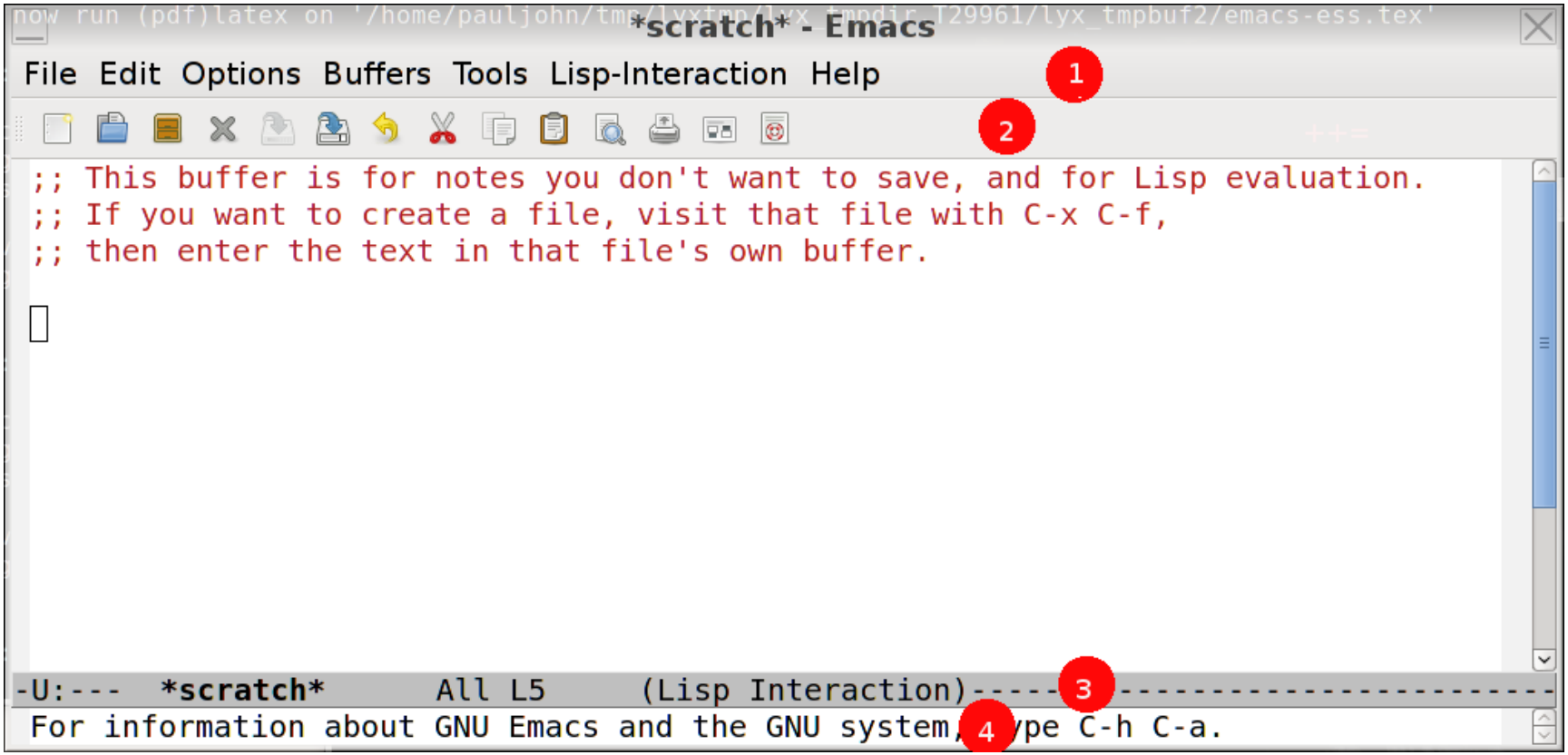
2 Emacs Anatomy

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," the collection of 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
- Type M-x ("meta" (usually the Alt key) and the letter x at the same time) and then enter a valid mode.
- (Hit Alt with x, release both keys. Then type a command and hit return)

Emacs "Major Mode" system

- Test some major modes for yourself.
  - M-x text-mode
  - M-x c-mode
  - M-x tex-mode
  - M-x R-mode

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
- Emacs still has ability to work in a no-picture, no-mouse "terminal" mode
  - There are many key board combinations using (C) Control, (S) Shift (S), and (M) Meta key (usually "Alt")
  - These are are featured in the Emacs tutorial, and make Emacs seem difficult to learn
  - Hence the prevalence of "Emacs cheat sheets" floating about on the Internet

Properly Understood, the Emacs Learning Curve is NOT STEEP

- Assume the user has the version of Emacs for Windows, Mac, or Linux system with X11 display.
- Emacs will run with pull down menus, more-or-less like other Editors.
- Its just a text editor with lots of great features (most of which you will never need).
- 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, because Emacs now has menus and buttons.
- 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! (as it was in 1982, in grad school)
- 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 that most of us don't need.
- The Emacs tutorial emphasizes memorization of keystrokes that are not truly necessary to use Emacs. So don't.
- Instead, *listen to me!* Use my init file. Make Emacs behave in a more familiar way.
  - scroll up and down with the mouse or arrow keys
  - copy, cut and paste regions with the mouse or the usual keyboard shortcuts
- 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.

Avoid the Emacs Tutorial Until Later ...

- Some enthusiasts write Lisp code for a living, but I think Lisp stands for “lots of irritating, stupid parentheses.”

Example of Easily Fixed "Problem"

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

keystroke	result
C-x	cut
s 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

Example of Easily Fixed "Problem" ...

- Fix. Turn on "CUA mode" in Emacs. Will make C-x, C-v, C-c, C-z do what you expect.
- 3 ways to turn on CUA mode.
  - In most recent Emacs, Click Options and choose CUA, or
  - M-x cua-mode, or
  - Make it permanent: Add this in your Emacs startup configuration

```
(cua-mode t)
```

- Potential problem to keep in mind: Emacs has many keystroke combinations that use C and M and anything we do to "override" them may cause trouble. But most of these bugs have been solved.

Example of Easily Fixed "Problem" ...

- 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.
- 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

- There are just a few times when it would be really handy to highlight a rectangular column of text and copy and paste it.
- This is a secret power of CUA mode. Emacs has an elaborate system to select "rectangular" sections, but it is difficult to use
- 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.
- That rectangular block can be copied, cut, or pasted. Sometimes, this can be very handy, especially when revising text that includes tables or columns.

Customizations I Recommend

- When Emacs starts, it reads a series of configuration files
  - Some are "system-wide"
  - Some are user specific
- On systems I administer, I generally customize the system-wide settings with my personal favorite settings `http://pj.freefaculty.org/Software/Emacs/50emacs-ess-ku.el`
- On systems I don't administer, I urge my users to install my configuration settings file. They they can share in my hard-found victory
- The user settings can be stored in the user's home folder, either in
  - .emacs, or
  - .emacs.d/init.el

Here's my Emacs init file (on my laptop 2012-08-23)

```
;; This is a testing version, will go up to
;; http://pj.freefaculty.org/Software/Emacs/50emacs-ess-ku.el
;; Nothing is new here,  I'm just trying to tidy up
;; Paul Johnson <pauljohn@ku.edu>
;; 2012-08-23
;;
;; This is my .emacs startup file. It makes
;; Emacs easy for me to use. If you try this file, I believe you can
;; use Emacs without knowing any Lisp and without even reading the
;; Emacs tutorial. The intention is to make Emacs work more like a
;; "modern" GUI editor.  I agree with the idea behind "oneonone Emacs"
;; (http://www.emacswiki.org/emacs/OneOnOneEmacs), but think it is too
;; difficult to implement and maintain.

;; Either you should put this file in your Emacs site-start.d
;; folder or put it in your home directory and call it
;; ".emacs". Or re-name it "init.el" and copy it into a new
;; directory in your Home folder called .emacs.d.

;; If you are using this because you use Emacs to use ESS (Emacs
;; Speaks Statistics) with R, here are my special features.

;; 1. Indentation policy follows Programming R Extensions Manual
;; 2. Shift+Enter will send the current line to R, and it will start R
;; if it is not running.
;; 3. R will start in the current working directory, without stopping
;; to ask the user about a working directory.
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
28      ;; 4. R runs in its own "frame"
29      ;; 5. Emacs help pops up in its own frame.
30
31
32
33      ;; Section I. Windows OS work-arounds
34
35      (if (eq system-type 'windows-nt)
36      (setq use-file-dialog nil))
37      ;; There's a problem with file selection dialogs on Windows
38
39
40      ;; Section II. Keyboard and mouse customization
41
42      ;; IIA: make mouse selection work in the usual Mac/Windows way
43      ;;(require 'pc-select)
44
45      (setq shift-select-mode t) ; is default in Emacs 23+, replaces pc-select
46      (transient-mark-mode t) ; highlight text selection
47      (delete-selection-mode t) ; delete seleted text when typing
48
49      ;; IIB: keyboard customization
50      (cua-mode t) ; windows style binding C-x, C-v, C-c, C-z, cut paste
51      (setq cua-auto-tabify-rectangles nil) ;; Don't tabify after rectangle
            commands
52      (setq cua-keep-region-after-copy t) ;; Selection remains after C-c
53
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
54      ;; write line numbers on left of window
55      ;; (global-linum-mode 1) ; always show line numbers
56
57
58
59      ;; Section III. Programming conveniences:
60      (show-paren-mode t) ; light-up matching parens
61      (global-font-lock-mode t) ; turn on syntax highlighting
62      (setq text-mode-hook (quote (turn-on-auto-fill text-mode-hook-identify)))
63
64
65
66      ;; Section IV. ESS Emacs Statistics
67
68      ;;(setq inferior-ess-own-frame t)
69      (setq inferior-ess-same-window nil)
70
71      ;; create a new frame for each help instance
72      ;; (setq ess-help-own-frame t)
73      ;; If you want all help buffers to go into one frame do:
74      (setq ess-help-own-frame 'one)
75
76      ;; minibuffer tips: will work in future
77      ;;(require 'ess-eldoc)
78      ;; html help
79      ; (setq inferior-ess-r-help-command "help(\"%s\", help_type=\"html\")\n")
80
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
81      ;;start R in current working directory, don't ask user
82      (setq ess-ask-for-ess-directory nil)
83      (setq ess-local-process-name "R")
84
85      ;; cause "Shift+Enter" to send the current line to *R*
86      (defun my-ess-eval ()
87      (interactive)
88      (if (and transient-mark-mode mark-active)
89      (call-interactively 'ess-eval-region)
90      (call-interactively 'ess-eval-line-and-step)))
91
92      (add-hook 'ess-mode-hook
93      '(lambda()
94      (local-set-key [(shift return)] 'my-ess-eval)))
95
96
97      ;; PJ 2012-03-21 Follow advice in Programming R Extensions: Use
98      ;; indentation in C++ style
99
100     (add-hook 'ess-mode-hook
101     (lambda ()
102     (ess-set-style 'C++ 'quiet)
103     (add-hook 'local-write-file-hooks
104     (lambda ()
105     (ess-nuke-trailing-whitespace)))))
106     ;;(setq ess-nuke-trailing-whitespace-p 'ask)
107     ;; or even
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
108     (setq ess-nuke-trailing-whitespace-p t)
109     ;;; Perl
110     (add-hook 'perl-mode-hook
111     (lambda () (setq perl-indent-level 4)))
112
113     ;;; End ESS
114
115
116
117     ;; Section V. Customize the use of Frames. Try to make new content
118     ;; appear in wholly new frames on screen.
119     ;;
120     ;; V.A: Discourage Emacs from splitting "frames", encourage it to
121     ;; pop up new frames for new content. see:
122     ;; http://www.gnu.org/software/emacs/elisp/html_node/Choosing-Window.html
123     (setq pop-up-frames t)
124     (setq special-display-popup-frame t)
125     (setq split-window-preferred-function nil) ;discourage horizontal splits
126     (setq pop-up-windows nil)
127
128
129     ;; V.B: Use "framepop" to pop up small frame messages. Some users don't
130     ;; have "framepop". I try to make this check to see if the feature is
131     ;; installed.
132
133     ;; Note that "pop-up-frames" is different from "framepop".
134     ;; Crazy!  The framepop package can "catch" some special small buffers
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
135     ;; and divert them off to a specially configured frame.  I made it pink!
136     ;; The framepop packages is in emacs-goodies on Ubuntu.
137
138     (when (require 'advice)
139     (when window-system
140     (when (require 'framepop nil 'noerror)
141     (framepop-enable))))
142
143     ;; Even if you don't have framepop, it is OK to leave these.
144     ;;
145     (setq framepop-frame-parameters
146     '((name . nil)                          ; use buffer name
147     (unsplittable . t)                      ; always include this
148     (menu-bar-lines . 0)                    ; no menu bar
149     (minibuffer . nil)                      ;    or minubuffer
150     (left . -1)                             ; top left corner of screen,
151     (top . 30)                              ;    away from my main frame
152     (width . 71)                            ; narrower, so it fits nicely
153     (background-color . "MistyRose")        ; for October.
154     (tool-bar-lines . 0)
155     (minibuffer)))
156
157     (setq framepop-min-frame-size 20)
158     (setq framepop-use-advice (quote automatic))
159     (setq framepop-auto-resize t)
160     ;; Stops framepop from fiddling with a few specific buffer types.
161     (setq special-display-buffer-names
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
162     '("*Help*" "*shell*" "*Completions*" "*grep*" "*tex-shell*"))
163
164
165     ;; V.C: Make files opened from the menu bar appear in their own
166     ;; frames. This overrides the default menu bar settings.  Opening an
167     ;; existing file and creating new one in a new frame are the exact
168     ;; same operations.  adapted from Emacs menu-bar.el
169     (defun menu-find-existing ()
170     "Edit the existing file FILENAME."
171     (interactive)
172     (let* ((mustmatch (not (and (fboundp 'x-uses-old-gtk-dialog)
173     (x-uses-old-gtk-dialog))))
174     (filename (car (find-file-read-args "Find file: " mustmatch))))
175     (if mustmatch
176     (find-file-other-frame filename)
177     (find-file filename))))
178     (define-key menu-bar-file-menu [new-file]
179     '(menu-item "Open/Create" find-file-other-frame
180     :enable (menu-bar-non-minibuffer-window-p)
181     :help "Create a new file"))
182     (define-key menu-bar-file-menu [open-file]
183     '(menu-item ,(purecopy "Open File...") menu-find-existing
184     :enable (menu-bar-non-minibuffer-window-p)
185     :help ,(purecopy "Read an existing file into an Emacs buffer")))
186
187     ;;(define-key menu-bar-file-menu [open-file]
188     ;;  '(menu-item "Open File..." find-file-other-frame
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
189     ;;          :enable (menu-bar-non-minibuffer-window-p)
190     ;;          :help "Open Existing File"))
191
192     ;; V.D  Open directory list in new frame.
193     (define-key menu-bar-file-menu [dired]
194     '(menu-item "Open Directory..." dired-other-frame
195     :help "Read a directory; operate on its files (Dired)"
196     :enable (not (window-minibuffer-p (frame-selected-window menu-updating-frame
            )))))
197
198
199
200
201     ;; Section VI: Miscellaneous convenience
202
203     ;; Remove Emacs "splash screen"
204     ;; http://fuhm.livejournal.com/
205     (defadvice command-line-normalize-file-name
206     (before kill-stupid-startup-screen activate)
207     (setq inhibit-startup-screen t))
208     (setq inhibit-splash-screen t)
209
210
211     ;; Show file name in title bar
212     ;; http://www.thetechrepo.com/main-articles/549
213     (setq frame-title-format "%b - Emacs")
214
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
215
216     ;; I'm right handed, need scroll bar on right (like other programs)
217     (setq scroll-bar-mode-explicit t)
218     (set-scroll-bar-mode `right)
219
220
221     ;; Make Emacs scroll smoothly with down arrow key.
222     ;;2011-10-14
223     ;;testing faq 5.45 http://www.gnu.org/s/emacs/emacs-faq.html#
            Modifying-pull_002ddown-menus
224     (setq scroll-conservatively most-positive-fixnum)
225
226
227     ;; If you want to adjust the size of the frames, uncomment this, adjust
            values
228     ;;(setq default-frame-alist '((width . 90) (height . 65)))
229
230
231     ;; Remember password when connected to remote sites via Tramp
232     ;; http://stackoverflow.com/questions/840279/
            passwords-in-emacs-tramp-mode-editing
233     ;; Emacs "tramp" service (ssh connection) constantly
234     ;; asks for the log in password without this
235     (setq password-cache-expiry nil)
236
237
238     ;; PJ 2011-05-15
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
239     ;;(setq ess-swv-pdflatex-commands '("pdflatex" "make"))
240     ;;(custom-set-variables
241
242
243
244     ;; Section : Emacs shells work better
245     ;;http://snarfed.org/why_i_run_shells_inside_emacs
246     (setq ansi-color-for-comint-mode 'filter)
247     (setq comint-prompt-read-only t)
248     (setq comint-scroll-to-bottom-on-input t)
249     (setq comint-scroll-to-bottom-on-output t)
250     (setq comint-move-point-for-output t)
251
252
253     ;;
254
255     ;; ================================================================
256     ;; From Marc Schwartz
257     ;; Set keys for 'windmove', built into Emacs  ==================
258     ;; Keyboard accelerator M-direction will "scroll" through
259     ;; buffers. Easier to remember than other ways.
260     ;; ================================================================
261
262     (windmove-default-keybindings 'meta)
263
264     (global-set-key (kbd "M-<up>") 'windmove-up)
265     (global-set-key (kbd "M-<down>") 'windmove-down)
```

Here's my Emacs init file (on my laptop 2012-08-23) ...

```
266     (global-set-key (kbd "M-<right>") 'windmove-right)
267     (global-set-key (kbd "M-<left>") 'windmove-left)
```

Another Easily Fixed "Problem"

- Selection of text in base Emacs used to be very different from other editors
- Several settings can be used to make selection work in a way that is more familiar to users (Emacs defaults are migrating in that direction).
- These lines in the startup configuration can (basically) completely solve the problem

```
(setq shift-select-mode t)
(transient-mark-mode t) ; highlight text
    selection
(delete-selection-mode t) ; delete
    seleted text when typing
```

As far as I can tell, the first two became defaults in Emacs-23, so I'm just "making sure" they are still set that way.

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 inden
  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?

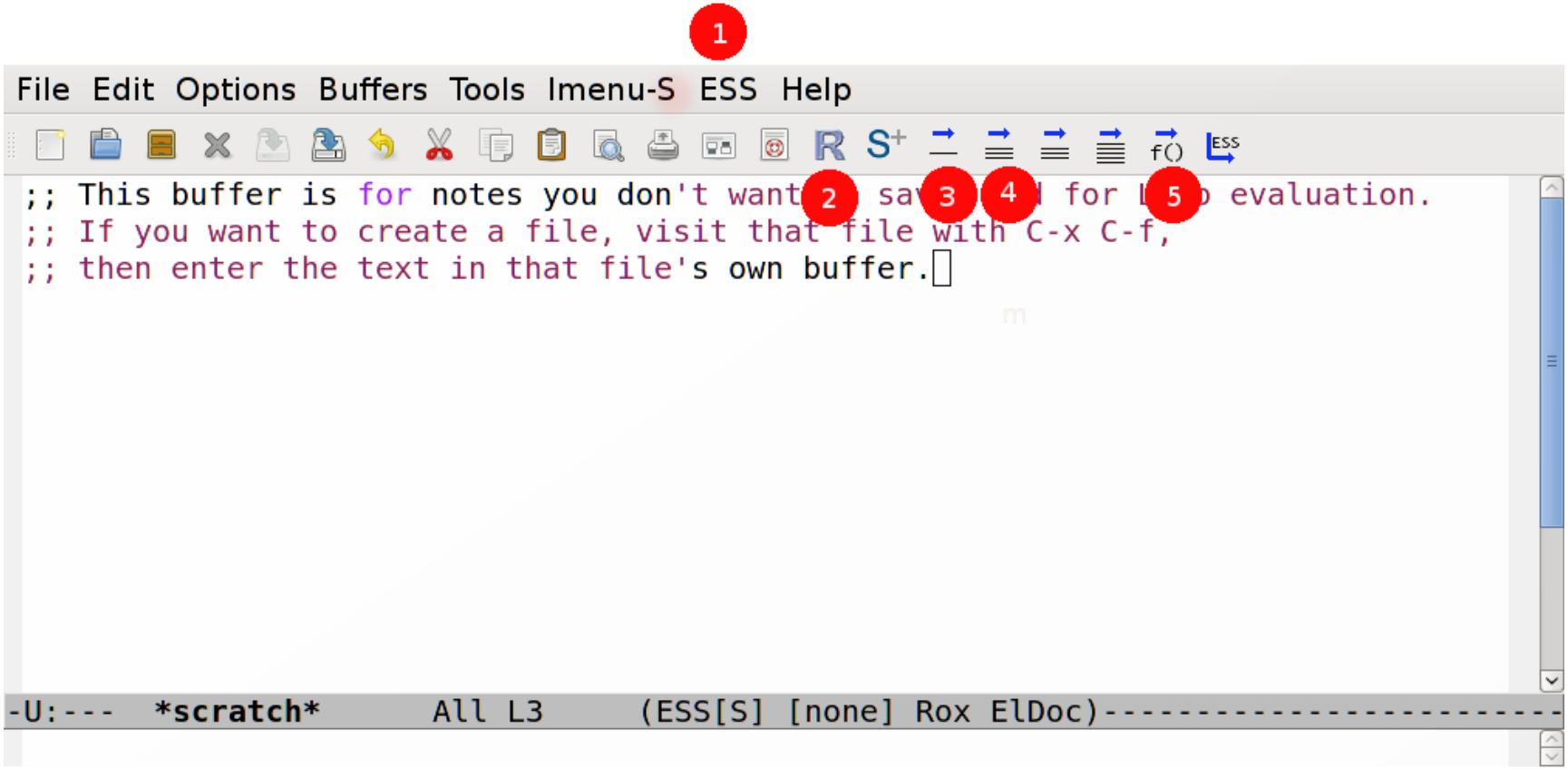
2 Emacs Anatomy

3 No Learning Curve

4 ESS

5 Conclusion

R-mode is provided by the ESS package

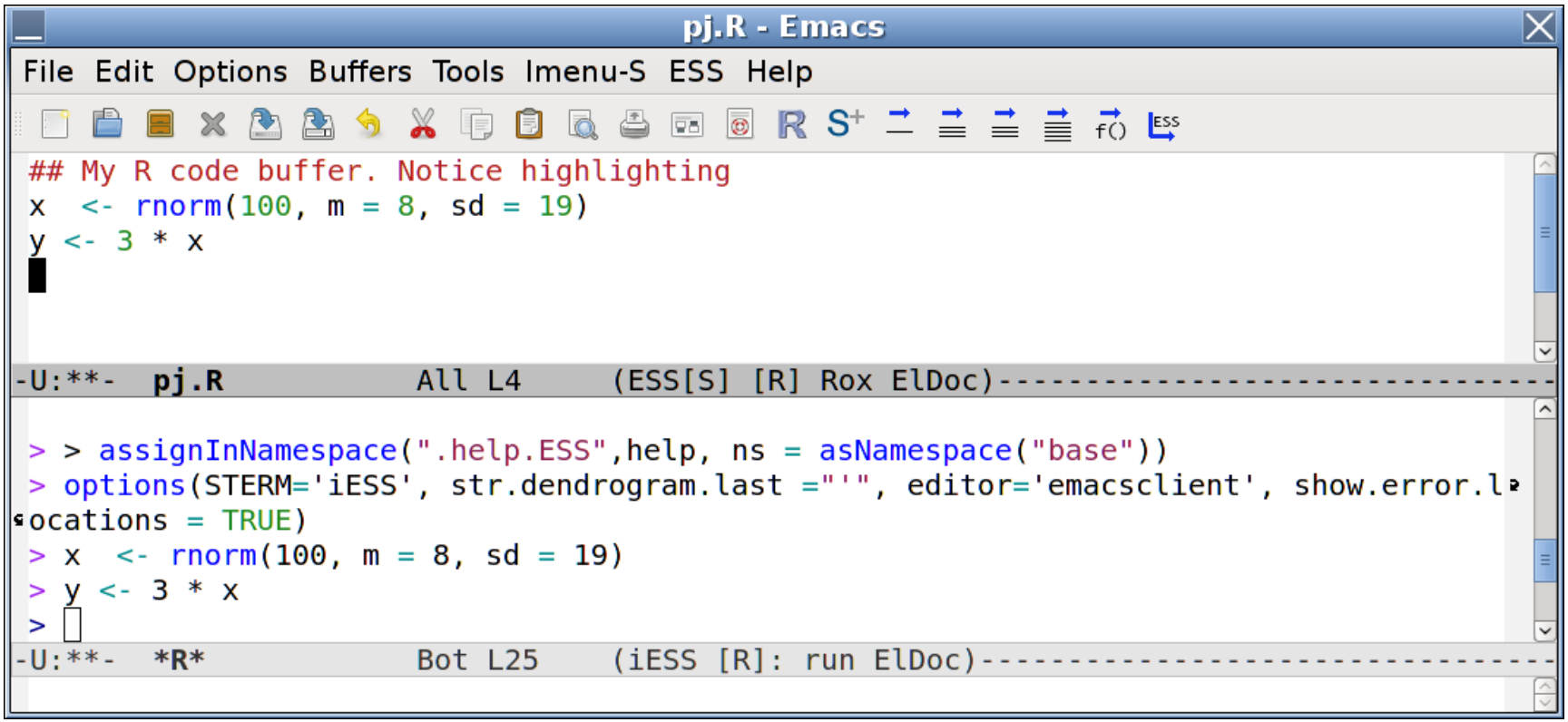


1. ESS menu 2. Blue R starter (same as M-x R) 3. Send one line to R 4. Send selection to R 5. Send current function to R.

Start an R Session Within Emacs

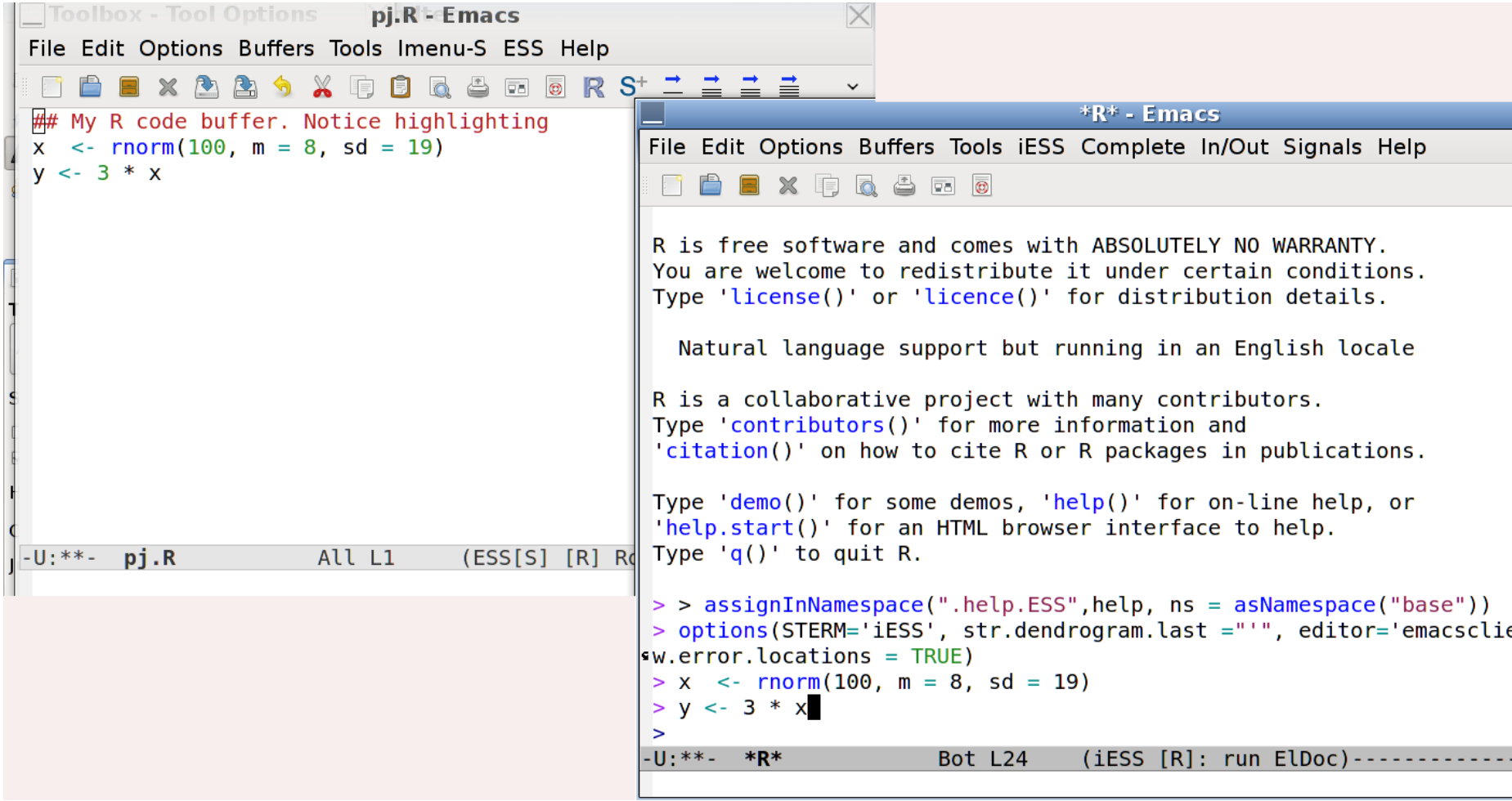
- Hit the big blue R button
- Hopefully, R starts inside a buffer, which Emacs refers to in the buffer list as *R*
- If Emacs can't find R in your system, you need to do some configuration work so that the R/bin folder is added to your system path (I have instructions for that in the crmda computing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

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



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

I prefer Separate Frames with R



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.

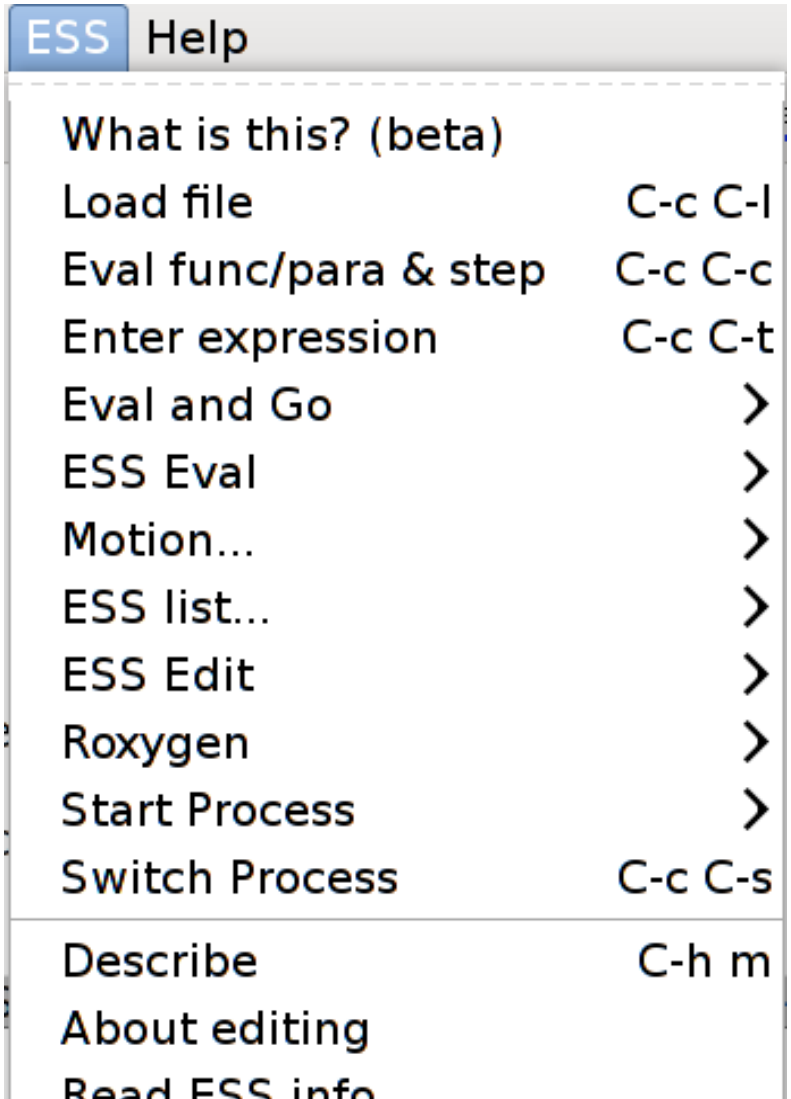
Using ESS button bar

- 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



- 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/vtwm_emacs/part2.html`