

Formas de utilización de R y uso de EMACS + ESS

F. Tusell*

Curso 2.005-2.006

Índice

1. Introducción	1
2. Requisitos	2
3. Escenarios posibles	2
3.1. Tecleo directo en la ventana interactiva de R	2
3.2. Uso de R en “batch”	2
3.2.1. En Unix/Linux	3
3.2.2. En Windows	3
3.2.3. Ventajas e inconvenientes.	3
3.3. Uso de <code>source()</code> y <code>sink()</code>	4
3.4. Uso de Emacs + ESS	4
3.5. Otras formas de utilizar Emacs + ESS	10
3.6. Mandatos útiles al editar utilizando ESS	10
3.7. Uso de Emacs + R + Sweave	10
4. Si quieres saber más . . .	12
A. Lista de comandos usuales de ESS	13
B. Lista de comandos usuales de Emacs	14

1. Introducción

R puede ser utilizado con bastante comodidad en forma interactiva. Comandos erróneos pueden ser reinvocados, corregidos y reejecutados. Sin embargo, a medida que el volumen del trabajo crece y, sobre todo, cuando queremos una transcripción

* Actualización del día 18 de octubre de 2005. La última versión de este documento, posiblemente más reciente, puede obtenerse de <http://www.et.bs.ehu.es/~etptupaf>

“limpia” y editada del mismo (por ejemplo, para documentar un informe o una tarea de clase), el modo simple interactivo que ofrece R se torna insuficiente.

Emacs es un editor extremadamente potente y adaptable que, entre otras cosas, puede servir como interface a R. Su uso es la opción recomendada, aunque en lo que sigue se mencionan otras que pueden ser de utilidad en diferentes situaciones.

2. Requisitos

Se presupone una instalación correcta y funcionando de R, Emacs y el “addon” de Emacs llamado ESS (Emacs Speaks Statistics)¹. Los tres componentes han sido integrados y pueden instalarse con mínimo esfuerzo sobre Windows 9x/NT/XP utilizando el CD ROM repartido a los alumnos de **Estadística: Modelos Lineales** y **Estadística: Análisis Multivariante**. Los mismos componentes son completamente “standard” en Unix/Linux y forman parte de bastantes distribuciones.

3. Escenarios posibles

3.1. Tecleo directo en la ventana interactiva de R

Es el escenario más elemental y simple. Abrimos una sesión² de R y tecleamos los mandatos deseados.

Si deseamos una transcripción de la sesión, marcamos con el ratón la parte deseada y la cortamos y pegamos en, por ejemplo, un documento Word (en Windows) o un fichero de texto ordinario o de un procesador de texto en Unix/Linux.

Los *objetos* que creemos persisten de una sesión a otra (si los salvamos), pero *la información acerca de cómo se crearon, no*.

3.2. Uso de R en “batch”

El “batch” o proceso por lotes consiste en someter a un programa un conjunto de instrucciones para ser ejecutadas secuencialmente. Era el modo de trabajo único en los albores de la informática. En general, es ventajoso poder trabajar interactivamente, viendo el resultado de ejecutar una instrucción antes de emitir la siguiente.

No obstante, procesos ya depurados pueden ventajosamente ejecutarse en “batch”, sin intervención del usuario, que no tiene que actuar ni esperar (si el proceso es muy

¹Las recomendaciones que siguen han sido probadas en Windows XP Professional Edition con R versión 2.0.0 y ESS versión 5.2.3 y en Linux (Debian 3.1 sobre Pentium) con R versión 1.9.1 y ESS versión 5.2.2, sin diferencias aparentes. Versiones diferentes, no muy antiguas, funcionarán seguramente del mismo modo.

²Pinchando sobre su icono en Windows, tecleando R en un terminal o pinchando sobre su icono, si lo hay, en Unix/Linux.

largo). Adicionalmente, en máquinas multiusuario es posible programar la ejecución en “horas valle”, cuando menos congestión existe en la máquina donde se ejecutan.

3.2.1. En Unix/Linux

Utilizaremos el editor de nuestra elección (*vi*, *pico*, *emacs*, ...) para generar un fichero de texto ASCII conteniendo las instrucciones a ejecutar. Lo salvaremos con el nombre que deseemos, preferiblemente con la extensión `.R`; por ejemplo `programa.R`. Desde una ventana de terminal teclearemos:

```
R CMD BATCH programa.R
```

Las instrucciones en `programa.R` se ejecutarán una tras otra y tanto las instrucciones como su resultado aparecerá en el fichero de nueva creación `programa.Rout`

3.2.2. En Windows

Utilizaremos el editor de nuestra elección para generar un fichero de texto ASCII conteniendo las instrucciones `R`. ¡Atención! Procesadores de texto como Word y similares incluyen caracteres no ASCII de formateo. Si empleamos Word, debemos asegurarnos de salvar el resultado como ASCII puro. Otra alternativa es Notepad. Desde luego, *emacs* será la opción preferible si está disponible.

A continuación, desde una ventana MS-DOS³ teclearemos:

```
R CMD BATCH programa.R
```

obteniendo como resultado el fichero de nueva creación `programa.Rout`. Este fichero quedará en la carpeta donde `R` se haya iniciado.

3.2.3. Ventajas e inconvenientes.

Tanto en Windows como Unix/Linux, la ventaja esencial del proceso en “batch” es que tenemos un listado de las instrucciones que generan un resultado. El resultado es así totalmente reproducible. Podemos además incluir los comentarios que queramos en `programa.R` y aparecerán en `programa.Rout`

¡Nunca añadiremos comentarios, etc. editando directamente `programa.Rout`! Si lo hiciéramos y constatáramos después que son precisas modificaciones, al volver a ejecutar `programa.R` obtendríamos un nuevo `programa.Rout` sin los comentarios añadidos: tendríamos que volver a repetir dicho trabajo de comentario. Nos conviene incluir todos los comentarios en `programa.R`, pues así reaparecerán en `programa.Rout` junto con los resultados modificados si hemos de cambiar datos o instrucciones.

³Que obtendremos pinchando el icono correspondiente. En Windows XP está un poco escondido: en la carpeta `C:\Windows\system32` buscaremos el fichero `cmd`. Nos convendrá copiarlo al Escritorio para su utilización frecuente.

Tener un “script” como `programa.R`, además de constituir una buena documentación que nos ayuda a saber qué hicimos para obtener un resultado, permite borrarlo y ahorrar sitio en disco. Por ejemplo, podemos escribir un `programa.R` que a partir de un fichero de datos genera multitud de tabulaciones cruzadas. Si conservamos tanto el fichero de datos como `programa.R`, podemos eliminar todas las tablas creadas con la tranquilidad de que las podemos recrear en cualquier momento⁴.

3.3. Uso de `source()` y `sink()`.

El uso de `source()` y `sink()` es similar bajo Windows y bajo Unix/Linux. Son dos funciones de R que desvían respectivamente la entrada y salida. Si desde dentro de una sesión R tecleamos

```
source("fichero")
```

instrucciones contenidas en el mismo. Las diferencias con `R CMD BATCH...` son:

1. Se trata, en principio, de una función silenciosa: se ejecutan las instrucciones sin producir ninguna salida⁵. Esto puede ser conveniente si `fichero` contiene, por ejemplo, instrucciones de inicialización que queremos ejecutar frecuentemente al comenzar o en algún momento de la sesión, sin necesidad de ver todos los resultados en la salida.
2. No crea un fichero externo `.Rout`. La salida, si se produjera alguna por emplear la opción `echo=TRUE`, la vemos en la pantalla.

`sink("fichero")` por su parte, como se ha indicado, redirige la salida de R al fichero de nombre `fichero`. Se puede obtener la salida en la pantalla y en `fichero` a la vez invocando `source` con la opción `split=TRUE`.

3.4. Uso de Emacs + ESS

Será, en general, la opción preferida, al darnos simultáneamente las ventajas de la interactividad y del proceso por lotes, con completa reproducibilidad del trabajo efectuado. A continuación, un ejemplo de sesión mostrando pantallas seleccionadas.

⁴Si, entre tanto, el fichero de datos de partida se modifica, las tablas que recreáramos harían uso de los datos más modernos, evitándose la situación indeseable de tener tablas creadas en diferentes momentos inconsistentes entre sí.

⁵Aunque esto se puede modificar invocando `source` con la opción `echo=TRUE`.

1. Inicia el editor Emacs. Teclea ESC x R para abrir una sesión interactiva con R. Verás,

```

emacs@RNPXR7
File Edit Options Buffers Tools ESS Complete In/Out Signals Help

R : Copyright 2004, The R Foundation for Statistical Computing
Version 2.0.0 (2004-10-04), ISBN 3-900051-07-0

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.

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 a HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> options(STERM='iESS', editor='gnuclient.exe')
>
**<ESPX82> *R* c:/Documents and Settings/etptupaf/ (iESS [R]: run)--L 20--
Type C-h m for help on ESS version 5.2.3
    
```

2. Parte la ventana en dos, para tener una ventana mostrando tu sesión interactiva R y otra donde podrás editar. Utiliza el mandato CTRL-x 3.

```

emacs@RNPXR7
File Edit Options Buffers Tools ESS Complete In/Out Signals Help

R : Copyright 2004, The R Foundation for Statistical Computing
Version 2.0.0 (2004-10-04), ISBN 3-900051-07-0

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.

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 a HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> options(STERM='iESS', editor='gnuclient.exe')
>
**<ESPX82> *R* c:/Documents and Settings/etptupaf/ (iESS [R]: run)--L 20--
Type C-h m for help on ESS version 5.2.3
    
```

- En una de las dos ventanas, edita un fichero (mediante *File* → *Open file*) y teclea las instrucciones de R que deseas ejecutar.

The screenshot shows the Emacs editor interface. The left pane displays the R console output, including the R startup message and the command `options(STERN='iESS', editor='gnuclie`. The right pane shows the contents of the file `prueba.R`, which contains the following R code:

```
library(MASS)
data(UScrime)
attach(UScrime)
lafit(cbind(laeq, Prob), y)
```

The status bar at the bottom indicates the current file is `prueba.R` located at `c:/Documents and Settings/BSPXB2/...`.

- Señala el fragmento de texto que deseas ejecutar arrastrando sobre él el ratón con el botón izquierdo apretado. Lo verás resaltado con fondo en color.

This screenshot is identical to the previous one, but with a yellow highlight box around the R code in the right pane:

```
library(MASS)
data(UScrime)
attach(UScrime)
lafit(cbind(laeq, Prob), y)
```

The status bar at the bottom remains the same, showing the file `prueba.R`.

5. Teclea CTRL-c CTRL-r. Te preguntará a qué proceso deseas transferir tu selección (fíjate en la línea inferior de la pantalla). Acepta que se transfiera a R.

```

Emacs@BSPX62
File Edit Options Buffers Tools Minibuf Help

R : Copyright 2004, The R Foundation fo
Version 2.0.0 (2004-10-04), ISBN 3-900

R is free software and comes with ABSOL
You are welcome to redistribute it unde
Type 'license()' or 'licence()' for dis

R is a collaborative project with many
Type 'contributors()' for more informat
'citation()' on how to cite R or R pack

Type 'demo()' for some demos, 'help()'
'help.start()' for a HTML browser inter
Type 'q()' to quit R.

[Previously saved workspace restored]

> options(STERN='iESS', editor='gnuclie
> 
**<BSPX62> *R* c:/Documents and S**<BSPX62> prueba.R c:/Documents and
Process to load into: R
    
```

6. Las instrucciones transferidas se ejecutan en tu sesión interactiva.

```

Emacs@BSPX62
File Edit Options Buffers Tools Minibuf Help

> options(STERN='iESS', editor='gnuclie
> library(MASS)
> data(UScrime)
> attach(UScrime)
> lsfit(cbind(Ineq, Prob), y)
$coefficients
      Intercept          Ineq          Pr
1209.2324768      0.2457999 -7471.27241

$residuals
 [1] 149.696339  599.224535 -69.570866
 [7]  27.013709  594.723297 123.68916
[13] -410.403673 -194.462765  40.14127
[19] -349.721631  234.972491 -344.81844
[25] -217.419619 1057.943056 -631.69320
[31] -571.606486 -176.289471 175.31666
[37] -102.979080 -295.103686 -93.03879
[43] -31.103856   -9.109305 -395.53619

$intercept
[1] TRUE

$qr
$qt
**<BSPX62> *R* c:/Documents and S**<BSPX62> prueba.R c:/Documents and
    
```

- Si los resultados no son de tu interés, puedes desecharlos con CTRL-c CTRL-o.

The screenshot shows an Emacs window titled 'emacs@B3PX62'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'ESS', 'Complete', 'In/Out', 'Signals', and 'Help'. The left pane contains the text '*** output flushed ***' followed by a prompt '>' and a cursor. The right pane contains the following R code:

```
library(MASS)
data(UScrime)
attach(UScrime)
lfit(cbind(Ineq, Prob), y)
[]
```

The status bar at the bottom shows the file path 'c:/Documents and Settings/B3PX62/prueba.R'.

- A continuación, puedes continuar editando en la ventana de la derecha las modificaciones que creas oportunas...

The screenshot shows the same Emacs window. The left pane now contains the prompt '>' followed by a cursor. The right pane contains the following R code, with a modification on the last line:

```
library(MASS)
data(UScrime)
attach(UScrime)
result <- lfit(cbind(Ineq, Prob), y)
```

The status bar at the bottom remains the same, showing the file path 'c:/Documents and Settings/B3PX62/prueba.R'.

9. ... y re-ejecutarlas en R (mediante marcado más CTRL-c CTRL-r).

The screenshot shows an Emacs window titled 'emacs@B3PX62'. The left pane contains the following R code:

```

*** output flushed ***
> result <- lsfit(cbind(Ineq, Prob), y)
>

```

The right pane contains the following R code:

```

library(MASS)
data(UScrime)
attach(UScrime)
result <- lsfit(cbind(Ineq, Prob), y)

```

The status bar at the bottom shows the file path: 'c:/Documents and Settings/B3PX62/prueba.R'.

10. Puedes también teclear directamente cosas en tu sesión interactiva. Si los resultados te convencen, puedes copiar el mandato ensayado en la ventana de la derecha.

The screenshot shows the same Emacs window. The left pane now displays the output of the R code:

```

*** output flushed ***
> result <- lsfit(cbind(Ineq, Prob), y)
> result$coefficients

```

Intercept	Ineq	Pr
1209.2324768	0.2457999	-7471.272411

The right pane contains the same R code as in the previous screenshot:

```

library(MASS)
data(UScrime)
attach(UScrime)
result <- lsfit(cbind(Ineq, Prob), y)

```

The status bar at the bottom shows the file path: 'c:/Documents and Settings/B3PX62/prueba.R'.

Las notas anteriores muestran un modo muy simple y efectivo de trabajar, que reúne lo mejor de la interactividad y el “batch”. Cuando la ventana de edición tenga ya un código suficientemente ensayado y depurado, al que habrás añadido los comentarios pertinentes, puedes salvarlo y re-ejecutarlo una última vez mediante `R CMD BATCH` (ver más arriba en la sección 3.2).

3.5. Otras formas de utilizar Emacs + ESS

Hay muchas, y cada cuál puede adoptar la que mejor se adapte a su forma de trabajo. Una posibilidad que encuentro útil es la de trabajar interactivamente y guardar copia de los mandatos empleados.

Si esto se hace simplemente guardando la transcripción de la sesión, nos encontraremos con mandatos y salidas entreveradas; y los mandatos están precedidos del “prompt” `>`. En consecuencia, no podemos simplemente “batchear” la transcripción de la sesión guardada tal cual.

ESS ofrece la opción de guardar una transcripción “limpia”. Para ello, señalaremos la parte de la sesión que desemos limpiar (arrastrando el ratón sobre ella con el botón izquierdo apretado) y teclearemos:

```
ESC x ess-transcript-clean-region RET
```

lo que dejará sólo los mandatos. Podremos así guardar el contenido de la sesión y emplearlo como entrada de un `R CMD BATCH` o de cualquier otro modo.

3.6. Mandatos útiles al editar utilizando ESS

`ESC-;` Comentar (precediendo cada línea de un `#`) la región señalada.

`ESC-x` `uncomment-region` Descomentar la región señalada.

`C-c C-v` Ayuda sobre alguna cuestión. `h` Situándose sobre un link, nos lleva a la página correspondiente.

3.7. Uso de Emacs + R + Sweave

En ocasiones interesa escribir un documento en $\text{\LaTeX}2\text{e}$ que incorpore resultados de cálculos realizados en R. En este caso, el paquete⁶ de nombre Sweave facilita sobre manera el trabajo.

La idea es simple: se escribe un documento \LaTeX ordinario, con extensión `.Rnw`, incluyendo entre marcas adecuadas código en R. Este documento se preprocesa mediante una función de R que ejecuta el código entre las marcas indicadas y sustituye su output en el mismo lugar, produciendo como salida un documento con extensión `.tex`. Este documento se “`latex`ea” como de ordinario.

Las marcas a que se hace referencia son

⁶Escrito por Friedrich Leisch, y disponible en <http://www.ci.tuwien.ac.at/~leisch/Sweave/>. Puede encontrarse documentación y la forma de instalarlo en dicha dirección.

```
<<>>=
```

para comenzar una sección de código y

```
@
```

para finalizarla. Por ejemplo, procesando el fichero a continuación, que suponemos nombrado `example1.Rnw`,

```
\documentclass[a4paper]{article}
\title{Sweave Example 1}
\author{Friedrich Leisch}
\begin{document}
\maketitle
```

In this example we embed parts of the examples from the `\texttt{kruskal.test}` help page into a `\LaTeX{}` document:

```
<<>>=
```

```
data(airquality)
library(ctest)
kruskal.test(Ozone ~ Month, data = airquality)
```

```
@
```

which shows that the location parameter of the Ozone distribution varies significantly from month to month. Finally we include a boxplot of the data:

```
\begin{center}
<<fig=TRUE,echo=FALSE>>=
boxplot(Ozone ~ Month, data = airquality)
@
\end{center}
\end{document}
```

obtendríamos un fichero de nombre `example1.tex` que podríamos formatear mediante \LaTeX . Obsérvese que es posible incluso incluir figuras. Obsérvese que no es preciso insertar manualmente fragmentos en el fichero `.tex`; el preprocesador lo hace por nosotros.

Para preprocesar el fichero `.Rnw` y convertirlo en su homólogo `.tex`, debemos cargar en R el package `tools`,

```
library("tools")
```

y ejecutar:

```
Sweave("example1.Rnw")
```

que producirá `example1.tex` en el directorio en que estamos trabajando.

Nada de lo anterior requiere Emacs ni ESS; no obstante, el proceso es más cómodo si tenemos el fichero `.Rnw` en un *buffer* de Emacs y una sesión R bajo ESS en otro. Podemos incluso programar las divertas operaciones (procesar, el `.Rnw`, etc.) de manera que las ejecutemos con un sólo golpe de tecla, en lugar de tener que teclear `Sweave("example1.Rnw")` cada vez⁷.

4. Si quieres saber más. . .

Hay bastantes referencias, muchas de ellas disponibles en Internet. Si has instalado Emacs y ESS sobre Windows desde el CD ROM facilitado a los estudiantes de Cuantitativa, en la carpeta `C:/ESS/doc` dispones de los ficheros `ess.pdf` y `ess-intro.pdf` (entre otros), que te interesará leer.

Cuanta información puedas necesitar sobre ESS la encontrarás en `http://stat.ethz.ch/ESS/`; de ahí procede, por ejemplo, la lista de comandos habituales reproducida en el Apéndice A.

En cuanto a Emacs, referencias de utilidad incluyen [1] y [2]. Este último está también disponible *on-line* en `http://www.delorie.com/gnu/docs/emacs/emacs_toc.html`.

Otras fuentes de información sobre Emacs en Internet incluyen `http://www.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html` y `http://www.gnu.org/software/emacs/windows/ntemacs.html`. En las páginas Web del Departamento (`http://www.et.bs.ehu.es`) en *Recursos* → *Software* → *Manuales* está disponible el manual de Emacs en PDF y en HTML, así como alguna información adicional.

⁷Si empleas la distribución de \LaTeX y el Emacs que se entrega en el CD ROM distribuido en clase, esta configuración está ya hecha, y puedes re-compilear un documento tecleando en sucesión F5, F6 y F7.

A. Lista de comandos usuales de ESS

ESS [Emacs Speaks Statistics] Reference Card for S and R

updated for ESS 5.x

Date: 2003/10/28 17:50:07 — as of October 28, 2003

1. NOTA BENE: S is the *language*, R is one *dialect*!
2. This is a list of the more widely used **key - shortcuts**. Many more are available, and most are accessible from the Emacs **Menus** such as iESS, ESS, etc.

Interacting with the S process

For use in a process buffer ‘*R*’ (inferior-ess-mode):

<RET>	Send a command
<TAB>	Complete S object name
C-c C-c	Break
C-g	interrupt Emacs' waiting for S
C-a / C-e	Beginning / End of command
C-c C-u	Delete this command
C-c C-w	Delete last word

Command history (part of Menu ‘In/Out’)

M-p	Previous command
M-n	Next command
C-c C-l	List command history (& choose!)
C-c M-r	Previous similar command
C-c M-s	Next similar command
C-c<RET>	Copy current input
C-c C-r	Top of last output
C-c C-o	Delete last output

Hot keys

C-c C-v	Help for S object
C-c C-l	Load source file (+ error check!)
C-c C-x	List objects
C-c C-s	Display search list
C-c C-a	Attach a directory
C-c C-d	Edit an object (dump to file)

Others

C-c ‘	Jump to error after C-c C-l
C-c C-q	Quit from S
C-c C-z	Kill the S process

Inside S Transcripts (I + O)

Inside ESS transcript buffers, (*.Rout files):

<RET>	Send and Move
C-c C-n	Next prompt
C-c C-p	Previous prompt
C-c C-w	Clean Region (→ input only)

Editing S source

For use in `ess-mode` edit buffers, (*.R files):

<TAB>	Indent this line
C-c<TAB>	Complete S object name
M-<TAB>	Complete file- / path- name
M-C-a	Beginning of function
M-C-e	End of function
M-C-q	Indent this expression (use at ‘{’)
M-C-h	Mark this function

Evaluation commands (Prefix C-u: *visibly*)

C-c C-l	Load this buffer – detect errors !
C-c C-n	Step through code – line by line
C-c C-e	Evaluate an expression
C-c C-j	Evaluate this line
C-c M-j	Evaluate this line and go
M-C-x	Evaluate this function
C-c C-f	Evaluate this function
C-c M-f	Evaluate this function and go
C-c C-r	Evaluate this region
C-c M-r	Evaluate this region and go
C-c C-b	Evaluate this buffer
C-c M-b	Evaluate this buffer and go

Others

C-c C-v	Help for S object
C-c C-d	“dump” – Edit another object
C-c C-z	Return to S process (at prompt)

At Sfs, or activated by M-x `ess-add-MM-keys`

C-c f	insert function() definition outline
-------	--------------------------------------

Reading help files

For use in ‘*help[R](...)*’ help buffers:

SPC	Next page
DEL	Previous page
b	Previous page (‘back’)
/	Search forwards
n	Next section
p	Previous section
s	Skip (‘jump’) to a named section
s e	e.g., skip to “Examples:”
l	Evaluate one ‘Example’ line
r	Evaluate current region
h	Help on another object
?	Help for this mode
q	Return to S process (‘quit’)
x	Kill this buffer and return (‘exit’)

B. Lista de comandos usuales de Emacs

GNU Emacs Reference Card

(for version 21)

Starting Emacs

To enter GNU Emacs 21, just type its name: `emacs`

To read in a file to edit, see Files, below.

Leaving Emacs

suspend Emacs (or iconify it under X)	<code>C-z</code>
exit Emacs permanently	<code>C-x C-c</code>

Files

read a file into Emacs	<code>C-x C-f</code>
save a file back to disk	<code>C-x C-s</code>
save all files	<code>C-x s</code>
insert contents of another file into this buffer	<code>C-x i</code>
replace this file with the file you really want	<code>C-x C-v</code>
write buffer to a specified file	<code>C-x C-w</code>
version control checkin/checkout	<code>C-x C-q</code>

Getting Help

The help system is simple. Type `C-h` (or `F1`) and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**.

remove help window	<code>C-x 1</code>
scroll help window	<code>C-M-v</code>
apropos: show commands matching a string	<code>C-h a</code>
show the function a key runs	<code>C-h c</code>
describe a function	<code>C-h f</code>
get mode-specific information	<code>C-h m</code>

Error Recovery

abort partially typed or executing command	<code>C-g</code>
recover a file lost by a system crash	<code>M-x recover-file</code>
undo an unwanted change	<code>C-x u</code> or <code>C-_</code>
restore a buffer to its original contents	<code>M-x revert-buffer</code>
redraw garbaged screen	<code>C-l</code>

Incremental Search

search forward	<code>C-s</code>
search backward	<code>C-r</code>
regular expression search	<code>C-M-s</code>
reverse regular expression search	<code>C-M-r</code>
select previous search string	<code>M-p</code>
select next later search string	<code>M-n</code>
exit incremental search	<code>RET</code>
undo effect of last character	<code>DEL</code>
abort current search	<code>C-g</code>

Use `C-s` or `C-r` again to repeat the search in either direction. If Emacs is still searching, `C-g` cancels only the part not done.

Motion

entity to move over	backward	forward
character	C-b	C-f
word	M-b	M-f
line	C-p	C-n
go to line beginning (or end)	C-a	C-e
sentence	M-a	M-e
paragraph	M-{	M-}
page	C-x [C-x]
sexp	C-M-b	C-M-f
function	C-M-a	C-M-e
go to buffer beginning (or end)	M-<	M->
scroll to next screen		C-v
scroll to previous screen		M-v
scroll left		C-x <
scroll right		C-x >
scroll current line to center of screen		C-u C-l

Killing and Deleting

entity to kill	backward	forward
character (delete, not kill)	DEL	C-d
word	M-DEL	M-d
line (to end of)	M-O C-k	C-k
sentence	C-x DEL	M-k
sexp	M-- C-M-k	C-M-k
kill region		C-w
copy region to kill ring		M-w
kill through next occurrence of <i>char</i>		M-z <i>char</i>
yank back last thing killed		C-y
replace last yank with previous kill		M-y

Marking

set mark here	C-@ or C-SPC
exchange point and mark	C-x C-x
set mark <i>arg</i> words away	M-@
mark paragraph	M-h
mark page	C-x C-p
mark sexp	C-M-@
mark function	C-M-h
mark entire buffer	C-x h

Query Replace

interactively replace a text string	M-%
using regular expressions	M-x query-replace-regexp
Valid responses in query-replace mode are	
replace this one, go on to next	SPC
replace this one, don't move	,
skip to next without replacing	DEL
replace all remaining matches	!
back up to the previous match	^
exit query-replace	RET
enter recursive edit (C-M-c to exit)	C-r

Multiple Windows

When two commands are shown, the second is for “other frame.”

delete all other windows	C-x 1	
split window, above and below	C-x 2	C-x 5 2
delete this window	C-x 0	C-x 5 0
split window, side by side	C-x 3	
scroll other window	C-M-v	
switch cursor to another window	C-x o	C-x 5 o
select buffer in other window	C-x 4 b	C-x 5 b
display buffer in other window	C-x 4 C-o	C-x 5 C-o
find file in other window	C-x 4 f	C-x 5 f
find file read-only in other window	C-x 4 r	C-x 5 r
run Dired in other window	C-x 4 d	C-x 5 d
find tag in other window	C-x 4 .	C-x 5 .
grow window taller	C-x ^	
shrink window narrower	C-x {	
grow window wider	C-x }	

Formatting

indent current line (mode-dependent)	TAB
indent region (mode-dependent)	C-M-\
indent sexp (mode-dependent)	C-M-q
indent region rigidly <i>arg</i> columns	C-x TAB
insert newline after point	C-o
move rest of line vertically down	C-M-o
delete blank lines around point	C-x C-o
join line with previous (with <i>arg</i> , next)	M-^
delete all white space around point	M-\
put exactly one space at point	M-SPC
fill paragraph	M-q
set fill column	C-x f
set prefix each line starts with	C-x .
set face	M-g

Case Change

uppercase word	M-u
lowercase word	M-l
capitalize word	M-c
uppercase region	C-x C-u
lowercase region	C-x C-l

The Minibuffer

The following keys are defined in the minibuffer.

complete as much as possible	TAB
complete up to one word	SPC
complete and execute	RET
show possible completions	?
fetch previous minibuffer input	M-p
fetch later minibuffer input or default	M-n
regexp search backward through history	M-r
regexp search forward through history	M-s
abort command	C-g

Type C-x ESC ESC to edit and repeat the last command that used the minibuffer. Type F10 to activate the menu bar using the minibuffer.

GNU Emacs Reference Card

Buffers

select another buffer	C-x b
list all buffers	C-x C-b
kill a buffer	C-x k

Transposing

transpose characters	C-t
transpose words	M-t
transpose lines	C-x C-t
transpose sexps	C-M-t

Spelling Check

check spelling of current word	M-\$
check spelling of all words in region	M-x ispell-region
check spelling of entire buffer	M-x ispell-buffer

Tags

find a tag (a definition)	M-.
find next occurrence of tag	C-u M-.
specify a new tags file	M-x visit-tags-table
regexp search on all files in tags table	M-x tags-search
run query-replace on all the files	M-x tags-query-replace
continue last tags search or query-replace	M-,

Shells

execute a shell command	M-!
run a shell command on the region	M-
filter region through a shell command	C-u M-
start a shell in window *shell*	M-x shell

Rectangles

copy rectangle to register	C-x r r
kill rectangle	C-x r k
yank rectangle	C-x r y
open rectangle, shifting text right	C-x r o
blank out rectangle	C-x r c
prefix each line with a string	C-x r t

Abbrevs

add global abbrev	C-x a g
add mode-local abbrev	C-x a l
add global expansion for this abbrev	C-x a i g
add mode-local expansion for this abbrev	C-x a i l
explicitly expand abbrev	C-x a e
expand previous word dynamically	M-/

Regular Expressions

any single character except a newline	.	(dot)
zero or more repeats	*	
one or more repeats	+	
zero or one repeat	?	
quote regular expression special character <i>c</i>	\	<i>c</i>
alternative ("or")		
grouping	(...)
same text as <i>n</i> th group	\	<i>n</i>
at word break	\	b
not at word break	\	B
entity	match start	match end
line	^	\$
word	\<	\>
buffer	\'	\'
class of characters	match these	match others
explicit set	[...]	[^ ...]
word-syntax character	\w	\W
character with syntax <i>c</i>	\s <i>c</i>	\S <i>c</i>

International Character Sets

specify principal language	M-x	set-language-environment
show all input methods	M-x	list-input-methods
enable or disable input method	C-\	
set coding system for next command	C-x	RET <i>c</i>
show all coding systems	M-x	list-coding-systems
choose preferred coding system	M-x	prefer-coding-system

Info

enter the Info documentation reader	C-h	i
find specified function or variable in Info	C-h	C-i
Moving within a node:		
scroll forward	SPC	
scroll reverse	DEL	
beginning of node	.	(dot)
Moving between nodes:		
next node	n	
previous node	p	
move up	u	
select menu item by name	m	
select <i>n</i> th menu item by number (1-9)	<i>n</i>	
follow cross reference (return with 1)	f	
return to last node you saw	l	
return to directory node	d	
go to any node by name	g	
Other:		
run Info tutorial	h	
quit Info	q	
search nodes for regexp	M-s	

Registers

save region in register	C-x r s
insert register contents into buffer	C-x r i
save value of point in register	C-x r SPC
jump to point saved in register	C-x r j

Keyboard Macros

start defining a keyboard macro	C-x (
end keyboard macro definition	C-x)
execute last-defined keyboard macro	C-x e
append to last keyboard macro	C-u C-x (
name last keyboard macro	M-x name-last-kbd-macro
insert Lisp definition in buffer	M-x insert-kbd-macro

Commands Dealing with Emacs Lisp

eval sexp before point	C-x C-e
eval current defun	C-M-x
eval region	M-x eval-region
read and eval minibuffer	M-:
load from standard system directory	M-x load-library

Simple Customization

customize variables and faces M-x customize

Making global key bindings in Emacs Lisp (examples):

```
(global-set-key "\C-cg" 'goto-line)
(global-set-key "\M-#" 'query-replace-regexp)
```

Writing Commands

```
(defun command-name (args)
  "documentation" (interactive "template")
  body)
```

An example:

```
(defun this-line-to-top-of-window (line)
  "Reposition line point is on to top of window.
With ARG, put point on line ARG."
  (interactive "P")
  (recenter (if (null line)
                0
                (prefix-numeric-value line))))
```

The `interactive` spec says how to read arguments interactively. Type C-h f `interactive` for more details.

Copyright © 1997 Free Software Foundation, Inc.
v2.2 for GNU Emacs version 21, 1997
designed by Stephen Gildea

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.

For copies of the GNU Emacs manual, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Referencias

- [1] D. Cameron and B. Rosenblatt. *Learning GNU Emacs*. O'Reilly and Assoc., 1991.
- [2] R.M. Stallman. *GNU Emacs Manual*. Free Software Foundation, 1997.