Octave Quick Reference Octave Version 1.1.1

Starting Octave

`octave`	start interactive Octave session
`octave` *file*	run Octave on commands in *file*
`octave --help`	describe command line options

Stopping Octave

`quit` or `exit`	exit Octave
`INTERRUPT`	(*e.g.* `C-c`) terminate current command and return to top-level prompt

Getting Help

`help`	list all commands and built-in variables
`help` *command*	briefly describe *command*
`help -i`	use Info to browse Octave manual
`help -i` *command*	search for *command* in Octave manual

Motion in Info

`SPC` or `C-v`	scroll forward one screenful
`DEL` or `M-v`	scroll backward one screenful
`C-l`	redraw the display

Node Selection in Info

`n`	select the next node
`p`	select the previous node
`u`	select the 'up' node
`t`	select the 'top' node
`d`	select the directory node
`<`	select the first node in the current file
`>`	select the last node in the current file
`g`	reads the name of a node and selects it
`C-x k`	kills the current node

Searching in Info

`s`	search for a string
`C-s`	search forward incrementally
`C-r`	search backward incrementally
`i`	search index & go to corresponding node
`,`	go to next match from last 'i' command

Command-Line Cursor Motion

`C-b`	move back one character
`C-f`	move forward one character
`C-a`	move the the start of the line
`C-e`	move to the end of the line
`M-f`	move forward a word
`M-b`	move backward a word
`C-l`	clear screen, reprinting current line at top

Inserting or Changing Text

`M-TAB`	insert a tab character
`DEL`	delete character to the left of the cursor
`C-d`	delete character under the cursor
`C-v`	add the next character verbatim
`C-t`	transpose characters at the point
`M-t`	transpose words at the point

[ ] surround optional arguments ... show one or more arguments

Killing and Yanking

`C-k`	kill to the end of the line
`C-y`	yank the most recently killed text
`M-d`	kill to the end of the current word
`M-DEL`	kill the word behind the cursor
`M-y`	rotate the kill ring and yank the new top

Command Completion and History

`TAB`	complete a command or variable name
`M-?`	list possible completions
`RET`	enter the current line
`C-p`	move 'up' through the history list
`C-n`	move 'down' through the history list
`M-<`	move to the first line in the history
`M->`	move to the last line in the history
`C-r`	search backward in the history list
`C-s`	search forward in the history list
`history` [-q] [*N*]	list *N* previous history lines, omitting history numbers if `-q`
`history -w` [*file*]	write history to *file* (`~/.octave_hist` if no *file* argument)
`history -r` [*file*]	read history from *file* (`~/.octave_hist` if no *file* argument)
`edit_history` *lines*	edit and then run previous commands from the history list
[*beg*] [*end*]	Specify the first and last history commands to edit or run.

If *beg* is greater than *end*, reverse the list of commands before editing. If *end* is omitted, select commands from *beg* to the end of the history list. If both arguments are omitted, edit the previous item in the history list.

`run_history` *lines* run previous commands from the history list

Shell Commands

`cd` *dir*	change working directory to *dir*
`pwd`	print working directory
`ls` [*options*]	print directory listing
`getenv` (*string*)	return value of named environment variable
`system` (*cmd*)	execute arbitrary shell command string

Matrices

Square brackets delimit literal matrices. Commas separate elements on the same row. Semicolons separate rows. Commas may be replaced by spaces, and semicolons may be replaced by one or more newlines. Elements of a matrix may be arbitrary expressions, provided that all the dimensions agree.

`[` *x*`,` *y*`, ... ]`	enter a row vector
`[` *x*`;` *y*`; ... ]`	enter a column vector
`[` *w*`,` *x*`;` *y*`,` *z* `]`	enter a 2×2 matrix

Ranges

base `:` *limit*
base `:` *incr* `:` *limit*

Specify a range of values beginning with *base* with no elements greater than *limit*. If it is omitted, the default value of *incr* is 1. Negative increments are permitted.

Strings and Co

A *string constant* co
enclosed in either dou

`\\`	a
`\"`	a
`\'`	a
`\n`	n
`\t`	h

Index Expressio

var (*idx*)	s
var (*idx1*, *idx2*)	s
scalar	s
vector	s
range	s
`:`	s

Global Variable

`global` *var1 ...* D

Global variables m
function without h
parameter list prov
within the function

Selected Built-i

`EDITOR`	e
`Inf`, `NaN`	I
`LOADPATH`	p
`PAGER`	p
`ans`	la
`eps`	n
`pi`	π
`realmax`	n
`realmin`	n

`automatic_replot`
`do_fortran_indexing`
`implicit_str_to_num`
`output_max_field_wi`
`output_precision`
`page_screen_output`
`prefer_column_vecto`
`resize_on_range_err`
`save_precision`
`silent_functions`
`warn_divide_by_zero`

`commas_in_literal_m`
control handling of

`ignore_function_tim`
ignore changes in f

`ok_to_lose_imaginar`
allow complex to r

`prefer_zero_one_ind`
if ambiguous, prefe

Statements

`for` *identifier* `=` *expr* *stmt-list* `endfor`

Execute *stmt-list* once for each column of *expr*. The variable *identifier* is set to the value of the current column during each iteration.

`while` (*condition*) *stmt-list* `endwhile`

Execute *stmt-list* while *condition* is true.

`break`	exit innermost loop
`continue`	go to beginning of innermost loop
`return`	return to calling function

`if` (*condition*) *if-body* [`else` *else-body*] `endif`

Execute *if-body* if *condition* is true, otherwise execute *else-body*.

`if` (*condition*) *if-body* [`elseif` (*condition*) *elseif-body*] `endif`

Execute *if-body* if *condition* is true, otherwise execute the *elseif-body* corresponding to the first `elseif` condition that is true, otherwise execute *else-body*.

Any number of `elseif` clauses may appear in an `if` statement.

`unwind_protect` *body* `unwind_protect_cleanup` *cleanup* `end`

Execute *body*. Execute *cleanup* no matter how control exits *body*.

Defining Functions

`function` [*ret-list*] *function-name* [(*arg-list*)]
function-body
`endfunction`

ret-list may be a single identifier or a comma-separated list of identifiers delimited by square-brackets.

arg-list is a comma-separated list of identifiers and may be empty.

Basic Matrix Manipulations

`rows` (*a*)	return number of rows of *a*
`columns` (*a*)	return number of columns of *a*
`all` (*a*)	check if all elements of *a* nonzero
`any` (*a*)	check if any elements of *a* nonzero
`find` (*a*)	return indices of nonzero elements
`sort` (*a*)	order elements in each column of *a*
`sum` (*a*)	sum elements in columns of *a*
`prod` (*a*)	product of elements in columns of *a*
`min` (*args*)	find minimum values
`max` (*args*)	find maximum values
`rem` (*x*, *y*)	find remainder of *x*/*y*
`reshape` (*a*, *m*, *n*)	reformat *a* to be *m* by *n*
`diag` (*v*, *k*)	create diagonal matrices
`linspace` (*b*, *l*, *n*)	create vector of linearly-spaced elements
`logspace` (*b*, *l*, *n*)	create vector of log-spaced elements
`eye` (*n*, *m*)	create *n* by *m* identity matrix
`ones` (*n*, *m*)	create *n* by *m* matrix of ones
`zeros` (*n*, *m*)	create *n* by *m* matrix of zeros
`rand` (*n*, *m*)	create *n* by *m* matrix of random values

Linear Algebra

`chol` (*a*)	Cholesky factorization
`det` (*a*)	compute the determinant of a matrix
`eig` (*a*)	eigenvalues and eigenvectors
`expm` (*a*)	compute the exponential of a matrix
`hess` (*a*)	compute Hessenberg decomposition
`inverse` (*a*)	invert a square matrix
`norm` (*a*, *p*)	compute the *p*-norm of a matrix
`pinv` (*a*)	compute pseudoinverse of *a*
`qr` (*a*)	compute the QR factorization of a matrix
`rank` (*a*)	matrix rank
`schur` (*a*)	Schur decomposition of a matrix
`svd` (*a*)	singular value decomposition
`syl` (*a*, *b*, *c*)	solve the Sylvester equation

Equations, ODEs, DAEs, Quadrature

`*fsolve`	solve nonlinear algebraic equations
`*lsode`	integrate nonlinear ODEs
`*dassl`	integrate nonlinear DAEs
`*quad`	integrate nonlinear functions

`perror` (*nm*, *code*) for functions that return numeric codes, print error message for named function and given error code

`*` See the on-line or printed manual for the complete list of arguments for these functions.

Signal Processing

`fft` (*a*)	Fast Fourier Transform using FFTPACK
`ifft` (*a*)	inverse FFT using FFTPACK
`freqz` (*args*)	FIR filter frequency response
`sinc` (*x*)	returns `sin` (π `x`)/(π `x`)

Image Processing

`colormap` (*map*)	set the current colormap
`gray2ind` (*i*, *n*)	convert gray scale to Octave image
`image` (*img*, *zoom*)	display an Octave image matrix
`imagesc` (*img*, *zoom*)	display scaled matrix as image
`imshow` (*img*, *map*)	display Octave image
`imshow` (*i*, *n*)	display gray scale image
`imshow` (*r*, *g*, *b*)	display RGB image
`ind2gray` (*img*, *map*)	convert Octave image to gray scale
`ind2rgb` (*img*, *map*)	convert indexed image to RGB
`loadimage` (*file*)	load an image file
`rgb2ind` (*r*, *g*, *b*)	convert RGB to Octave image
`saveimage` (*file*, *img*, *fmt*, *map*)	save a matrix to *file*

Sets

`create_set` (*a*, *b*)	create row vector of unique values
`complement` (*a*, *b*)	elements of *b* not in *a*
`intersection` (*a*, *b*)	intersection of sets *a* and *b*
`union` (*a*, *b*)	union of sets *a* and *b*

Strings

`strcmp` (*s*, *t*)	compare strings
`strcat` (*s*, *t*, ...)	concatenate strings

C-style Input a

`fopen` (*name*, *mode*)
`fclose` (*file*)
`printf` (*fmt*, ...)
`fprintf` (*file*, *fmt*,
`sprintf` (*fmt*, ...)
`scanf` (*fmt*)
`fscanf` (*file*, *fmt*)
`sscanf` (*str*, *fmt*)
`fgets` (*file*, *len*)
`fflush` (*file*)
`ftell` (*file*)
`frewind` (*file*)
`freport`
`fread` (*file*, *size*, *pr*
`fwrite` (*file*, *size*, *p*
`feof` (*file*)

A file may be referen
returned from `fopen`.
Octave starts: `stdin`,

Other Input an

`save` *file var* ... s
`load` *file* l
`disp` (*var*) d

Miscellaneous

`eval` (*str*) e
`feval` (*str*, ...) e

`error` (*message*) p

`clear` *pattern* c
`exist` (*str*) c
`who` li

Polynomials

`compan` (*p*) c
`conv` (*a*, *b*) c
`deconv` (*a*, *b*) d
`poly` (*a*) c
`polyderiv` (*p*) d
`polyreduce` (*p*) i
`polyval` (*p*, *x*) v
`polyvalm` (*p*, *x*) v
`roots` (*p*) p
`residue` (*a*, *b*) p

Statistics

`corrcoef` (*x*, *y*) c
`cov` (*x*, *y*) c
`mean` (*a*) m
`median` (*a*) m
`std` (*a*) s
`var` (*a*) v