${ m MATLAB\,{}_{ m R}}$ / R Reference

David Hiebeler
Dept. of Mathematics and Statistics
University of Maine
Orono, ME 04469-5752
http://www.math.umaine.edu/~hiebeler

I wrote the first version of this reference during the Spring 2007 semester, as I learned R while teaching my Modeling & Simulation course at the University of Maine. The course covers population and epidemiological modeling, including deterministic and stochastic models in discrete and continuous time, along with spatial models. Half of the class meetings are in a regular classroom, and half are in a computer lab where students work through modeling & simulation exercises. When I taught earlier versions of the course, it was based on MATLAB only. In Spring 2007, some biology graduate students in the class who had learned R in statistics courses asked if they could use R in my class as well, and I said yes. My colleague Bill Halteman was a great help as I frantically learned R to stay ahead of the class. As I went, every time I learned how to do something in R for the course, I added it to this reference, so that I wouldn't forget it later. Some items took a huge amount of time searching for a simple way to do what I wanted, but at the end of the semester, I was pleasantly surprised that almost everything I do in MATLAB had an equivalent in R. I was also inspired to do this after seeing the "R for Octave Users" reference written by Robin Hankin. I've continued to add to the document, with many additions based on topics that came up while teaching courses on Advanced Linear Algebra and Numerical Analysis.

This reference is organized into general categories. There is also a MATLAB index and an R index at the end, which should make it easy to look up a command you know in one of the languages and learn how to do it in the other (or if you're trying to read code in whichever language is unfamiliar to you, allow you to translate back to the one you are more familiar with). The index entries refer to the item numbers in the first column of the reference document, rather than page numbers.

Any corrections, suggested improvements, or even just notification that the reference has been useful are appreciated. I hope all the time I spent on this will prove useful for others in addition to myself and my students. Note that sometimes I don't necessarily do things in what you may consider the "best" way in a particular language. I often tried to do things in a similar way in both languages, and where possible I've avoided the use of MATLAB toolboxes or R packages which are not part of the core distributions. But if you believe you have a "better" way (either simpler, or more computationally efficient) to do something, feel free to let me know.

Acknowledgements: Thanks to Alan Cobo-Lewis and Isaac Michaud for correcting some errors; and Robert Bryce, Thomas Clerc, Richard Cotton, Stephen Eglen, Andreas Handel, Niels Richard Hansen, David Khabie-Zeitoune, Michael Kiparsky, Andy Moody, Ben Morin, Lee Pang, Manas A. Pathak, Rachel Rier, Rune Schjellerup Philosof, and Corey Yanofsky for contributions.

Permission is granted to make and distribute verbatim copies of this manual provided this permission notice is preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Free Software Foundation.

Contents

1	Help	3
2	Entering/building/indexing matrices 2.1 Cell arrays and lists	3 6 6
3	Computations 3.1 Basic computations 3.2 Complex numbers 3.3 Matrix/vector computations 3.4 Root-finding 3.5 Function optimization/minimization 3.6 Numerical integration / quadrature 3.7 Curve fitting	6 7 8 14 14 15 16
4	Conditionals, control structure, loops	17
5	Functions, ODEs	21
6	Probability and random values	23
7	Graphics 7.1 Various types of plotting	27 27 35 36
8	Working with files	37
9	Miscellaneous 9.1 Variables	38 38 39
10	Spatial Modeling	42
In	dex of MATLAB commands and concepts	43
In	dex of R commands and concepts	48

1 Help

No.	Description	Matlab	R
1	Show help for a function (e.g.	help sqrt, or helpwin sqrt to see	help(sqrt) or ?sqrt
	$\mathbf{sqrt})$	it in a separate window	
2	Show help for a built-in key-	help for	help('for') or ?'for'
	word (e.g. for)		
3	General list of many help top-	help	library() to see available libraries,
	ics		or library(help='base') for very
			long list of stuff in base package which
			you can see help for
4	Explore main documentation	doc or helpbrowser (previously it	help.start()
	in browser	was helpdesk, which is now being	
		phased out)	
5	Search documentation for	lookfor binomial	help.search('binomial')
	keyword or partial keyword		
	(e.g. functions which refer to		
	"binomial")		

2 Entering/building/indexing matrices

No.	Description	Matlab	R
6	Enter a row vector \vec{v} =	v=[1 2 3 4]	v=c(1,2,3,4) or alternatively
	$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$		v=scan() then enter "1 2 3 4" and
			press Enter twice (the blank line
			terminates input)
7	Enter a column vector $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$	[1; 2; 3; 4]	c(1,2,3,4)
			(R does not distinguish between row and column vectors.)
8	Enter a matrix $ \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} $	[1 2 3 ; 4 5 6]	To enter values by row: matrix(c(1,2,3,4,5,6), nrow=2, byrow=TRUE) To enter values by column: matrix(c(1,4,2,5,3,6), nrow=2)
9	Access an element of vector ${f v}$	v(3)	v[3]
10	Access an element of matrix A	A(2,3)	A[2,3]
11	Access an element of matrix A using a single index: indices count down the first column, then down the second column, etc.	A(5)	A[5]
12	Build the vector [2 3 4 5 6 7]	2:7	2:7
13	Build the vector [7 6 5 4 3 2]	7:-1:2	7:2
14	Build the vector [2 5 8 11 14]	2:3:14	seq(2,14,3)

N.T.	D : ::	34	D
No.	Description	MATLAB	R
15	Build a vector containing	linspace(a,b,n)	seq(a,b,length.out=n) or just
	n equally-spaced values be-		seq(a,b,len=n)
	tween a and b inclusive		
16	Build a vector containing	logspace(a,b,n)	10^seq(a,b,len=n)
	n logarithmically equally-		_
	spaced values between 10^a		
	and 10^b inclusive		
17	Build a vector of length k	zeros(k,1) (for a column vector) or	rep(0,k)
	containing all zeros	zeros(1,k) (for a row vector)	2007(0,11)
18	Build a vector of length k	j*ones(k,1) (for a column vector)	rep(j,k)
10	containing the value j in all	or j*ones(1,k) (for a row vector)	[Tep(J,k)
	positions the value j in an	or j*ones(1,k) (for a row vector)	
10	_		
19	Build an $m \times n$ matrix of zeros	zeros(m,n)	matrix(0,nrow=m,ncol=n) or just
			matrix(0,m,n)
20	Build an $m \times n$ matrix con-	j*ones(m,n)	matrix(j,nrow=m,ncol=n) or just
	taining j in all positions		matrix(j,m,n)
21	$n \times n$ identity matrix I_n	eye(n)	diag(n)
22	Build diagonal matrix A us-	diag(v)	diag(v,nrow=length(v)) (Note: if
	ing elements of vector \mathbf{v} as di-		you are sure the length of vector \mathbf{v} is 2
	agonal entries		or more, you can simply say diag(v).)
23	Extract diagonal elements of	v=diag(A)	v=diag(A)
	\mid matrix A		
24	"Glue" two matrices a1 and	[a1 a2]	cbind(a1,a2)
	a2 (with the same number of	[41 42]	
	rows) side-by-side		
25	"Stack" two matrices a1 and	[a1; a2]	rbind(a1,a2)
25		[a1; a2]	rbind(a1,a2)
	a2 (with the same number of		
	columns) on top of each other		
26	Given vectors \mathbf{x} and \mathbf{y} of	<pre>[X,Y]=meshgrid(x,y)</pre>	
	lengths m and n respectively,		<pre>m=length(x); n=length(y);</pre>
	build $n \times m$ matrices X whose		<pre>X=matrix(rep(x,each=n),nrow=n);</pre>
	rows are copies of \mathbf{x} and \mathbf{Y}		Y=matrix(rep(y,m),nrow=n)
	whose columns are copies of		- mass = 1 (2 op (y , m) , 1 o m = 1.)
	y		
27	Reverse the order of elements	v(end:-1:1)	rev(v)
	in vector v		
28	Column 2 of matrix A	A(:,2)	A[,2] Note: that gives the result as a
			vector. To make the result a $m \times 1$ ma-
			trix instead, do A[,2,drop=FALSE]
29	Row 7 of matrix A	A(7,:)	A[7,] Note: that gives the result as a
-5			vector. To make the result a $1 \times n$ ma-
			trix instead, do A[7,,drop=FALSE]
30	All elements of A as a vector,	A(:) (gives a column vector)	c(A)
30	column-by-column	1 (81,000 a columni Accion)	
91	Rows 2–4, columns 6–10 of A	A(2.4 6.10)	A [0.4 6.10]
31		A(2:4,6:10)	A[2:4,6:10]
6.2	(this is a 3×5 matrix)	. (5	
32	A 3×2 matrix consisting of	A([7 7 6], [2 1])	A[c(7,7,6),c(2,1)]
	rows 7, 7, and 6 and columns		
	2 and 1 of A (in that order)		
33	Circularly shift the rows of	circshift(A, [s1 s2])	No simple way, but modulo arithmetic
	matrix A down by s_1 ele-		on indices will work: m=dim(A)[1];
	ments, and right by s_2 ele-		n=dim(A)[2]; A[(1:m-s1-1)%/m+1,
	ments		(1:n-s2-1)%%n+1]
	l .	l .	

No.	Description	Matlab	R
34	Flip the order of elements in	fliplr(A)	t(apply(A,1,rev))
	each row of matrix A		
35	Flip the order of elements in	flipud(A)	apply(A,2,rev)
	each column of matrix A	-	
36	Given a single index ind into		
	an $m \times n$ matrix A , compute	[m ol = indOgub(gigo(A) ind)	n = ((ind-1) % m) + 1
	the row \mathbf{r} and column \mathbf{c} of	[r,c] = ind2sub(size(A), ind)	r = ((ind-1) %% m) + 1 c = floor((ind-1) / m) + 1
	that position (also works if		C = 1100r((1nd-1) / m) + 1
	ind is a vector)		
37	Given the row \mathbf{r} and column		
	c of an element of an $m \times n$	<pre>ind = sub2ind(size(A), r, c)</pre>	ind = (c-1)*m + r
	matrix \mathbf{A} , compute the single	Ind - Subzind(Size(A), 1, C)	Ind = (C 1)*m 1
	index ind which can be used		
	to access that element of ${\bf A}$		
	(also works if \mathbf{r} and \mathbf{c} are vec-		
	tors)		
38	Given equal-sized vectors ${f r}$		
	and \mathbf{c} (each of length k), set	<pre>inds = sub2ind(size(A),r,c);</pre>	inds = cbind(r,c)
	elements in rows (given by \mathbf{r})	A(inds) = 12;	A[inds] = 12
	and columns (given by \mathbf{c}) of		
	matrix A equal to 12. That		
	is, k elements of A will be		
	modified.		
39	Truncate vector \mathbf{v} , keeping	v = v(1:10)	v = v[1:10], or length(v) = 10
10	only the first 10 elements		also works
40	Extract elements of vector v	v(a:end)	v[a:length(v)]
	from position a to the end	45.	
41	All but the k^{th} element of	v([1:(k-1) (k+1):end])	v[-k]
40	vector v	N . 1 . 2 C . 1: . 1	F (
42	All but the j^{th} and k^{th} ele-	No simple way? Generalize the pre-	v[c(-j,-k)]
40	ments of vector v	vious item	1: (1)
43	Reshape matrix A, making it	A = reshape(A,m,n)	dim(A) = c(m,n)
	an $m \times n$ matrix with ele-		
	ments taken columnwise from		
	the original A (which must		
11	have mn elements) Extract the lower-triangular	L = tril(A)	L = A; L[upper.tri(L)]=0
44	portion of matrix A	r - ctit(w)	L - A; L[upper.cri(L)]=0
45	Extract the upper-triangular	U = triu(A)	U = A; U[lower.tri(U)]=0
40	portion of matrix A	O CIIU(A)	A, OLIOWEI.CII(O)]-O
46	Enter $n \times n$ Hilbert matrix H	hilb(n)	Hilbert(n), but this is part of the
10	where $H_{ij} = 1/(i+j-1)$		Matrix package which you'll need to
			install (see item 331 for how to in-
			stall/load packages).
47	Enter an n -dimensional array,	reshape(1:24, 3, 4, 2) or	array(1:24, c(3,4,2)) (Note that
*'	e.g. a $3 \times 4 \times 2$ array with the	reshape(1:24, [3 4 2])	a matrix is 2-D, i.e. rows and
	values 1 through 24		columns, while an array is more gen-
			erally N -D)
			Ciwily 11-D)

2.1 Cell arrays and lists

No.	Description	Matlab	R
48	Build a vector v of length n , capable of containing differ-	v = cell(1,n) In general,	v = vector('list',n) Then you
	ent data types in different elements (called a <i>cell array</i> in MATLAB, and a <i>list</i> in R)	cell(m,n) makes an $m \times n$ cell array. Then you can do e.g.: v{1} = 12 v{2} = 'hi there' v{3} = rand(3)	<pre>can do e.g.: v[[1]] = 12 v[[2]] = 'hi there' v[[3]] = matrix(runif(9),3)</pre>
49	Extract the i^{th} element of a cell/list vector \mathbf{v}	<pre>w = v{i} If you use regular indexing, i.e. w = v(i), then w will be a 1 × 1 cell matrix containing the contents of the ith element of v.</pre>	<pre>w = v[[i]] If you use regular indexing, i.e. w = v[i], then w will be a list of length 1 containing the contents of the ith element of v.</pre>
50	Set the name of the i^{th} element in a list.	(Matlab does not have names associated with elements of cell arrays.)	names(v)[3] = 'myrandmatrix' Use names(v) to see all names, and names(v)=NULL to clear all names.

2.2 Structs and data frames

No.	Description	Matlab	R
51	Create a matrix-like object	avals=2*ones(1,6);	v=c(1,5,3,2,3,7); d=data.frame(
	with different named columns	yvals=6:-1:1; v=[1 5 3 2 3 7];	cbind(a=2, yy=6:1), v)
	(a struct in Matlab, or a	d=struct('a',avals,	
	data frame in R)	'yy', yyvals, 'fac', v);	

Note that I (surprisingly) don't use R for statistics, and therefore have very little experience with data frames (and also very little with MATLAB structs). I will try to add more to this section later on.

3 Computations

3.1 Basic computations

No.	Description	Matlab	R
52	a+b, a-b, ab, a/b	a+b, a-b, a*b, a/b	a+b, a-b, a*b, a/b
53	\sqrt{a}	sqrt(a)	sqrt(a)
54	a^b	a^b	a^b
55	a (note: for complex ar-	abs(a)	abs(a)
	guments, this computes the		
	modulus)		
56	e^a	exp(a)	exp(a)
57	$\ln(a)$	log(a)	log(a)
58	$\log_2(a), \log_{10}(a)$	log2(a), log10(a)	log2(a), log10(a)
59	$\sin(a), \cos(a), \tan(a)$	sin(a), cos(a), tan(a)	sin(a), cos(a), tan(a)
60	$\sin^{-1}(a)$, $\cos^{-1}(a)$, $\tan^{-1}(a)$	asin(a), acos(a), atan(a)	asin(a), acos(a), atan(a)
61	$\sinh(a), \cosh(a), \tanh(a)$	<pre>sinh(a), cosh(a), tanh(a)</pre>	sinh(a), cosh(a), tanh(a)
62	$\sinh^{-1}(a), \qquad \cosh^{-1}(a),$	asinh(a), acosh(a), atanh(a)	asinh(a), acosh(a), atanh(a)
	$\tanh^{-1}(a)$		

No.	Description	Matlab	R
63	$n \mod k$ (modulo arithmetic)	mod(n,k)	n %% k
64	Round to nearest integer	round(x)	round(x) (Note: R uses IEC 60559 standard, rounding 5 to the even digit—so e.g. round(0.5) gives 0, not 1.)
65	Round down to next lowest integer	floor(x)	floor(x)
66	Round up to next largest integer	ceil(x)	ceiling(x)
67	Sign of x (+1, 0, or -1)	<pre>sign(x) (Note: for complex values, this computes x/abs(x).)</pre>	sign(x) (Does not work with complex values)
68	Error function $\operatorname{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-t^2} dt$	erf(x)	2*pnorm(x*sqrt(2))-1
69	Complementary error function $\operatorname{cerf}(x) = (2/\sqrt{\pi}) \int_x^{\infty} e^{-t^2} dt = 1\operatorname{-erf}(x)$	erfc(x)	2*pnorm(x*sqrt(2),lower=FALSE)
70	Inverse error function	erfinv(x)	qnorm((1+x)/2)/sqrt(2)
71	Inverse complementary error function	erfcinv(x)	qnorm(x/2,lower=FALSE)/sqrt(2)
72	Binomial coefficient $\binom{n}{k} = n!/(n!(n-k)!)$	nchoosek(n,k)	choose(n,k)

Note: the various functions above (logarithm, exponential, trig, abs, and rounding functions) all work with vectors and matrices, applying the function to each element, as well as with scalars.

3.2 Complex numbers

No.	Description	Matlab	R
73	Enter a complex number	1+2i	1+2i
74	Modulus (magnitude)	abs(z)	abs(z) or Mod(z)
75	Argument (angle)	angle(z)	Arg(z)
76	Complex conjugate	conj(z)	Conj(z)
77	Real part of z	real(z)	Re(z)
78	Imaginary part of z	imag(z)	Im(z)

3.3 Matrix/vector computations

No.	Description	Matlab	R
79	Vector dot product $\vec{x} \cdot \vec{y} = \vec{x}^T \vec{y}$	dot(x,y)	sum(x*y)
80	Vector cross product $\vec{x} \times \vec{y}$	cross(x,y)	Not in base R, but e.g. the xprod function from the RSEIS package will do it (see item 331 for how to install/load packages)
81	Matrix multiplication AB	A * B	A %*% B
82	Element-by-element multiplication of A and B	A .* B	A * B
83	Transpose of a matrix, A^T	A' (This is actually the complex conjugate (i.e. Hermitian) transpose; use A.' for the non-conjugate transpose if you like; they are equivalent for real matrices.)	t(A) for transpose, or Conj(t(A)) for conjugate (Hermitian) transpose
84	Solve $A\vec{x} = \vec{b}$	A\b Warning: if there is no solution, MATLAB gives you a least-squares "best fit." If there are many solutions, MATLAB just gives you one of them.	solve(A,b) Warning: this only works with square invertible matrices.
85	Reduced echelon form of A	rref(A)	R does not have a function to do this
86	Determinant of A	det(A)	det(A)
87	Inverse of A	inv(A)	solve(A)
88	Trace of A	trace(A)	<pre>sum(diag(A))</pre>
89	Compute AB^{-1}	A/B	A %*% solve(B)
90	Element-by-element division of A and B	A ./ B	A / B
91	Compute $A^{-1}B$	A\B	solve(A,B)
92	Square the matrix A	A^2	A %*% A
93	Raise matrix A to the k^{th} power	A^k	(No easy way to do this in R other than repeated multiplication A %*% A %*% A)
94	Raise each element of A to the k^{th} power	A.^k	A^k
95	Rank of matrix A	rank(A)	qr(A)\$rank
96	Set w to be a vector of eigenvalues of A , and V a matrix containing the corresponding eigenvectors	[V,D]=eig(A) and then w=diag(D) since MATLAB returns the eigenvalues on the diagonal of D	<pre>tmp=eigen(A); w=tmp\$values; V=tmp\$vectors</pre>
97	Permuted LU factorization of a matrix	[L,U,P]=lu(A) then the matrices satisfy $PA = LU$. Note that this works even with non-square matrices	tmp=expand(lu(Matrix(A))); L=tmp\$L; U=tmp\$U; P=tmp\$P then the matrices satisfy $A = PLU$, i.e. $P^{-1}A = LU$. Note that the lu and expand functions are part of the Ma- trix package (see item 331 for how to install/load packages). Also note that this doesn't seem to work correctly with non-square matrices. L , U , and P will be of class Matrix rather than class matrix; to make them the latter, instead do L=as.matrix(tmp\$L), U=as.matrix(tmp\$U), and P=as.matrix(tmp\$P) above.

No.	Description	Matlab	R
98	Singular-value decomposition: given $m \times n$ matrix A with rank r , find $m \times r$ matrix P with orthonormal columns, diagonal $r \times r$ matrix S , and $r \times n$ matrix Q^T with orthonormal rows so that $PSQ^T = A$	[P,S,Q]=svd(A,'econ')	<pre>tmp=svd(A); U=tmp\$u; V=tmp\$v; S=diag(tmp\$d)</pre>
99	Schur decomposition of square matrix, $A = QTQ^H = QTQ^{-1}$ where Q is unitary (i.e. $Q^HQ = I$) and T is upper triangular; $Q^H = \overline{Q^T}$ is the Hermitian (conjugate) transpose	[Q,T]=schur(A)	tmp=Schur(Matrix(A)); T=tmp@T; Q=tmp@Q Note that Schur is part of the Matrix package (see item 331 for how to install/load packages). T and Q will be of class Matrix rather than class matrix; to make them the latter, instead do T=as.matrix(tmp@T) and Q=as.matrix(tmp@Q) above.
100	Cholesky factorization of a square, symmetric, positive definite matrix $A = R^T R$, where R is upper-triangular	R = chol(A)	R = chol(A) Note that chol is part of the Matrix package (see item 331 for how to install/load packages).
101	QR factorization of matrix A , where Q is orthogonal (satisfying $QQ^T = I$) and R is upper-triangular	[Q,R]=qr(A) satisfying $QR = A$, or [Q,R,E]=qr(A) to do permuted QR factorization satisfying $AE = QR$	z=qr(A); $Q=qr.Q(z)$; $R=qr.R(z)$; $E=diag(n)[,z$pivot]$ (where n is the number of columns in A) gives permuted QR factorization satisfying $AE=QR$
102	Vector norms	$\operatorname{norm}(\mathbf{v},1)$ for 1-norm $\ \vec{v}\ _1$, $\operatorname{norm}(\mathbf{v},2)$ for Euclidean norm $\ \vec{v}\ _2$, $\operatorname{norm}(\mathbf{v},\inf)$ for infinity-norm $\ \vec{v}\ _{\infty}$, and $\operatorname{norm}(\mathbf{v},\mathbf{p})$ for p -norm $\ \vec{v}\ _p = (\sum v_i ^p)^{1/p}$	R does not have a norm function for vectors; only one for matrices. But the following will work: $\operatorname{norm}(\operatorname{matrix}(v),'1')$ for 1-norm $\ \vec{v}\ _1$, $\operatorname{norm}(\operatorname{matrix}(v),'i')$ for infinity-norm $\ \vec{v}\ _{\infty}$, and $\operatorname{sum}(\operatorname{abs}(v)\hat{p})\hat{1}/p$ for p -norm $\ \vec{v}\ _p = (\sum v_i ^p)^{1/p}$
103	Matrix norms	$\operatorname{norm}(A,1)$ for 1-norm $\ A\ _1$, $\operatorname{norm}(A)$ for 2-norm $\ A\ _2$, $\operatorname{norm}(A,\inf)$ for infinity-norm $\ A\ _{\infty}$, and $\operatorname{norm}(A,\inf)$ for Frobenius norm $\left(\sum_i (A^T A)_{ii}\right)^{1/2}$	norm(A,'1') for 1-norm $ A _1$, max(svd(A)\$d) for 2-norm $ A _2$, norm(A,'i') for infinity-norm $ A _{\infty}$, and norm(A,'f') for Frobenius norm $\left(\sum_i (A^T A)_{ii}\right)^{1/2}$
104	Condition number cond(A) = $ A _1 A^{-1} _1$ of A, using 1-norm	cond(A,1) (Note: MATLAB also has a function rcond(A) which computes reciprocal condition estimator using the 1-norm)	1/rcond(A,'1')
105	Condition number cond(A) = $ A _2 A^{-1} _2$ of A, using 2-norm	cond(A,2)	kappa(A, exact=TRUE) (leave out the "exact=TRUE" for an estimate)
106	Condition number cond(A) = $ A _{\infty} A^{-1} _{\infty}$ of A, using infinity-norm	<pre>cond(A,inf)</pre>	1/rcond(A,'I')

No.	Description	Matlab	R
107	Compute mean of all ele-	mean(v) for vectors, mean(A(:)) for	mean(v) or mean(A)
	ments in vector or matrix	matrices	
108	Compute means of columns of a matrix	mean(A)	colMeans(A)
109	Compute means of rows of a matrix	mean(A,2)	rowMeans(A)
110	Compute standard deviation	std(v) for vectors, std(A(:)) for	sd(v) for vectors, sd(c(A)) for ma-
	of all elements in vector or	matrices. This normalizes by $n-1$.	trices. This normalizes by $n-1$.
	matrix	Use $std(v,1)$ to normalize by n .	
111	Compute standard deviations	std(A). This normalizes by $n-1$.	sd(A). This normalizes by $n-1$.
	of columns of a matrix	Use $std(A,1)$ to normalize by n	
112	Compute standard deviations	$\operatorname{std}(A,0,2)$ to normalize by $n-1$,	apply(A,1,sd). This normalizes by
110	of rows of a matrix	$\operatorname{std}(A,1,2)$ to normalize by n	n-1.
113	Compute variance of all ele-	var(v) for vectors, var(A(:)) for	var(v) for vectors, var(c(A)) for
	ments in vector or matrix	matrices. This normalizes by $n-1$.	matrices. This normalizes by $n-1$.
114	Compute variance of columns	Use $var(v,1)$ to normalize by n . var(A). This normalizes by $n-1$.	apply(A,2,var). This normalizes by
114	of a matrix	Use $var(A,1)$ to normalize by $n-1$.	apply (A, 2, var). This normalizes by $n-1$.
115	Compute variance of rows of	$\operatorname{var}(A,0,2)$ to normalize by $n-1$,	n-1. apply(A,1,var). This normalizes by
110	a matrix	var(A,1,2) to normalize by n	n-1.
116	Compute covariance for two	$cov(v,w)$ computes the 2×2 co-	cov(v,w)
	vectors of observations	variance matrix; the off-diagonal ele-	
		ments give the desired covariance	
117	Compute covariance matrix,	cov(A)	var(A) or cov(A)
	giving covariances between		
	columns of matrix A		
118	Given matrices A and B ,	I don't know of a direct way to	cov(A,B)
	build covariance matrix C	do this in Matlab. But one way is	
	where c_{ij} is the covariance between column i of A and col-	<pre>[Y,X]=meshgrid(std(B),std(A)); X.*Y.*corr(A,B)</pre>	
	umn j of B	A. +1. +COII (A,B)	
119	Compute Pearson's linear	corr(v,w) Note: v and w must	cor(v,w)
	correlation coefficient be-	be column vectors. To make it	
	tween elements of vectors ${f v}$	work regardless of whether they	
	and \mathbf{w}	are row or column vectors, do	
		corr(v(:),w(:))	
120	Compute Kendall's tau corre-	<pre>corr(v,w,'type','kendall')</pre>	cor(v,w,method='kendall')
	lation statistic for vectors \mathbf{v}		
121	and w Compute Spearman's rho	corr(u u lturo) larearmani)	cor(v,w,method='spearman')
121	Compute Spearman's rho correlation statistic for	<pre>corr(v,w,'type','spearman')</pre>	Cor(v,w,method='spearman')
	vectors v and w		
122	Compute pairwise Pearson's	corr(A) The 'type' argument may	cor(A) The method argument may
-	correlation coefficient be-	also be used as in the previous two	also be used as in the previous two
	tween columns of matrix	items	items
	A		
123	Compute matrix C of pair-	corr(A,B) The 'type' argument	cor(A,B) The method argument
	wise Pearson's correlation co-	may also be used as just above	may also be used as just above
	efficients between each pair of		
	columns of matrices A and B ,		
	i.e. so c_{ij} is the correlation		
	between column i of A and		
	column j of B		

No.	Description	Matlab	R	
124	Compute sum of all elements	<pre>sum(v) for vectors, sum(A(:)) for</pre>	sum(v) or sum(A)	
	in vector or matrix	matrices		
125	Compute sums of columns of matrix	sum(A)	colSums(A)	
126	Compute sums of rows of matrix	sum(A,2)	rowSums(A)	
127	Compute product of all elements in vector or matrix	<pre>prod(v) for vectors, prod(A(:)) for matrices</pre>	prod(v) or prod(A)	
128	Compute products of columns of matrix	prod(A)	apply(A,2,prod)	
129	Compute products of rows of matrix	prod(A,2)	apply(A,1,prod)	
130	Compute matrix exponential $e^A = \sum_{k=0}^{\infty} A^k / k!$	expm(A)	expm(Matrix(A)), but this is part of the Matrix package which you'll need to install (see item 331 for how to in- stall/load packages).	
131	Compute cumulative sum of values in vector	cumsum(v)	cumsum(v)	
132	Compute cumulative sums of columns of matrix	cumsum(A)	apply(A,2,cumsum)	
133	Compute cumulative sums of rows of matrix	cumsum(A,2)	t(apply(A,1,cumsum))	
134	Compute cumulative sum of all elements of matrix (column-by-column)	cumsum(A(:))	cumsum(A)	
135	Cumulative product of elements in vector v	cumprod(v) (Can also be used in the various ways cumsum can)	cumprod(v) (Can also be used in the various ways cumsum can)	
136	Cumulative minimum or maximum of elements in vector v	I don't know of an easy way to do this in Matlab	cummin(v) or cummax(v)	
137	Compute differences between consecutive elements of vector \mathbf{v} . Result is a vector \mathbf{w} 1 element shorter than \mathbf{v} , where element i of \mathbf{w} is element $i+1$ of \mathbf{v} minus element i of \mathbf{v}	diff(v)	diff(v)	
138	Make a vector \mathbf{y} the same size as vector \mathbf{x} , which equals 4 everywhere that \mathbf{x} is greater than 5, and equals 3 everywhere else (done via a vectorized computation).	$z = [3 \ 4]; y = z((x > 5)+1)$	y = ifelse(x > 5, 4, 3)	
139	Compute minimum of values in vector \mathbf{v}	min(v)	min(v)	
140	Compute minimum of all values in matrix \mathbf{A}	min(A(:))	min(A)	
141	Compute minimum value of each column of matrix A	min(A) (returns a row vector)	apply(A,2,min) (returns a vector)	
142	Compute minimum value of each row of matrix \mathbf{A}	min(A, [], 2) (returns a column vector)	apply(A,1,min) (returns a vector)	

No.	Description	Matlab	R
143	Given matrices \mathbf{A} and \mathbf{B} ,	min(A,B)	pmin(A,B)
	compute a matrix where each		
	element is the minimum of		
	the corresponding elements of		
	${f A}$ and ${f B}$		
144	Given matrix \mathbf{A} and scalar	min(A,c)	pmin(A,c)
	c, compute a matrix where		
	each element is the minimum		
	of ${\bf c}$ and the corresponding el-		
	ement of $\bf A$		
145	Find minimum among all val-	min([A(:) ; B(:)])	min(A,B)
	ues in matrices ${\bf A}$ and ${\bf B}$		
146	Find index of the first time	[y,ind] = min(v)	<pre>ind = which.min(v)</pre>
	min(v) appears in v , and		
	store that index in ind		

Notes:

- Matlab and R both have a max function (and R has pmax and which.max as well) which behaves in the same ways as min but to compute maxima rather than minima.
- Functions like exp, sin, sqrt etc. will operate on arrays in both Matlab and R, doing the computations for each element of the matrix.

Mo	Description	MATLAB	R
No.	•		
147	Number of rows in A	size(A,1)	nrow(A)
148	Number of columns in A	size(A,2)	ncol(A)
149	Dimensions of A , listed in a vector	size(A)	dim(A)
150	Number of elements in vector \mathbf{v}	length(v)	length(v)
151	Total number of elements in matrix A	numel(A)	length(A)
152	Max. dimension of A	length(A)	max(dim(A))
153	Sort values in vector v	sort(v)	sort(v)
154	Sort values in \mathbf{v} , putting sorted values in \mathbf{s} , and indices in \mathbf{idx} , in the sense that $\mathbf{s}[\mathbf{k}]$ = $\mathbf{x}[\mathbf{idx}[\mathbf{k}]]$	[s,idx]=sort(v)	<pre>tmp=sort(v,index.return=TRUE); s=tmp\$x; idx=tmp\$ix</pre>
155	Sort the order of the rows of matrix m	sortrows(m) This sorts according to the first column, then uses column 2 to break ties, then column 3 for remaining ties, etc. Complex numbers are sorted by abs(x), and ties are then broken by angle(x).	m[order(m[,1]),] This only sorts according to the first column. To use column 2 to break ties, and then column 3 to break further ties, do m[order(m[,1], m[,2], m[,3]),] Complex numbers are sorted first by real part, then by imaginary part.
156	Sort order of rows of matrix m, specifying to use columns c1, c2, c3 as the sorting "keys"	sortrows(m, [c1 c2 c2])	m[order(m[,c1], m[,c2], m[,c3]),]

No.	Description	Matlab	R
157	Same as previous item, but sort in decreasing order for columns c1 and c2	sortrows(m, [-c1 -c2 c2])	m[order(-m[,c1], -m[,c2], m[,c3]),]
158	Sort order of rows of matrix m , and keep indices used for sorting	[y,i] = sortrows(m)	i=order(m[1,]); y=m[i,]
159	To count how many values in the vector v are between 4 and 7 (inclusive on the upper end)	sum((v > 4) & (v <= 7))	sum((v > 4) & (v <= 7))
160	Given vector v , return list of indices of elements of v which are greater than 5	find(v > 5)	which(v > 5)
161	Given matrix A , return list of indices of elements of A which are greater than 5, us- ing single-indexing	find(A > 5)	which(A > 5)
162	Given matrix A , generate vectors r and c giving rows and columns of elements of A which are greater than 5	[r,c] = find(A > 5)	<pre>w = which(A > 5, arr.ind=TRUE); r=w[,1]; c=w[,2]</pre>
163	Given vector x (of presumably discrete values), build a vector v listing unique values in x , and corresponding vector c indicating how many times those values appear in x	<pre>v = unique(x); c = hist(x,v);</pre>	<pre>w=table(x); c=as.numeric(w); v=as.numeric(names(w))</pre>
164	Given vector \mathbf{x} (of presumably continuous values), divide the range of values into k equally-sized bins, and build a vector \mathbf{m} containing the midpoints of the bins and a corresponding vector \mathbf{c} containing the counts of values in the bins	[c,m] = hist(x,k)	<pre>w=hist(x,seq(min(x),max(x), length.out=k+1), plot=FALSE); m=w\$mids; c=w\$counts</pre>
165	Convolution / polynomial multiplication (given vectors x and y containing polynomial coefficients, their convolution is a vector containing coefficients of the product of the two polynomials)	conv(x,y)	convolve(x,rev(y),type='open') Note: the accuracy of this is not as good as MATLAB; e.g. doing v=c(1,-1); for (i in 2:20) v=convolve(v,c(-i,1), type='open') to generate the 20^{th} -degree Wilkinson polynomial $W(x) = \prod_{i=1}^{20} (x-i)$ gives a coefficient of ≈ -780.19 for x^{19} , rather than the correct value -210.

3.4 Root-finding

No.	Description	Matlab	R
166	Find roots of polynomial	roots(v)	polyroot(rev(v)) (This function
	whose coefficients are stored		really wants the vector to have the
	in vector \mathbf{v} (coefficients in \mathbf{v}		constant coefficient first in v ; rev re-
	are highest-order first)		verses their order to achieve this.)
167	Find zero (root) of a function	Define function $f(x)$, then do	Define function $f(x)$, then do
	f(x) of one variable	fzero(f,x0) to search for a root	uniroot(f, c(a,b)) to find a root
		near $x0$, or fzero(f,[a b]) to find	between a and b , assuming the sign
		a root between a and b , assuming	of $f(x)$ differs at $x = a$ and $x = b$.
		the sign of $f(x)$ differs at $x = a$	Default forward error tolerance (i.e.
		and $x = b$. Default forward error	error in x) is fourth root of machine
		tolerance (i.e. error in x) is machine	epsilon, $(\epsilon_{\rm mach})^{0.25}$. To specify e.g.
		epsilon ϵ_{mach} .	a tolerance of 2^{-52} , do uniroot(f,
			c(a,b), tol=2^-52).

3.5 Function optimization/minimization

No.	Description	Matlab	R
168	Find value m which mini-	Define function $f(x)$, then do	Define function $f(x)$, then do
	mizes a function $f(x)$ of one variable within the interval from a to b	m = fminbnd(f, a, b)	<pre>m = optimize(f,c(a,b))\$minimum</pre>
169	Find value m which minimizes a function $f(x, p_1, p_2)$ with given extra parameters (but minimization is only occurring over the first argument), in the interval from a to b .	Define function f(x,p1,p2), then use an "anonymous function": % first define values for p1 % and p2, and then do: m=fminbnd(@(x) f(x,p1,p2),a,b)	Define function f(x,p1,p2), then: # first define values for p1 # and p2, and then do: m = optimize(f, c(a,b), p1=p1, p2=p2)\$minimum
170	Find values of x, y, z which minimize function $f(x, y, z)$, using a starting guess of $x = 1$, $y = 2.2$, and $z = 3.4$.	First write function $f(\mathbf{v})$ which accepts a vector argument \mathbf{v} containing values of $x, y,$ and $z,$ and returns the scalar value $f(x, y, z),$ then do: fminsearch(@f,[1 2.2 3.4])	First write function $f(\mathbf{v})$ which accepts a vector argument \mathbf{v} containing values of x , y , and z , and returns the scalar value $f(x, y, z)$, then do: optim(c(1,2.2,3.4),f)\$par
171	Find values of x, y, z which minimize function $f(x, y, z, p_1, p_2)$, using a starting guess of $x = 1$, $y = 2.2$, and $z = 3.4$, where the function takes some extra parameters (useful e.g. for doing things like nonlinear least-squares optimization where you pass in some data vectors as extra parameters).	First write function $f(v,p1,p2)$ which accepts a vector argument v containing values of x , y , and z , along with the extra parameters, and returns the scalar value $f(x,y,z,p_1,p_2)$, then do: fminsearch(@f,[1 2.2 3.4], [], p1, p2) Or use an anonymous function: fminsearch(@(x) f(x,p1,p2), [1 2.2 3.4])	First write function $f(\mathbf{v},\mathbf{p1},\mathbf{p2})$ which accepts a vector argument \mathbf{v} containing values of $x, y,$ and $z,$ along with the extra parameters, and returns the scalar value $f(x,y,z,p_1,p_2),$ then do: optim(c(1,2.2,3.4), f, p1=p1, p2=p2)\$par

${\bf 3.6}\quad {\bf Numerical\ integration\ /\ quadrature}$

No.	Description	Matlab	R
172	Numerically integrate func-	quad(f,a,b) uses adaptive Simp-	integrate(f,a,b) uses adaptive
	tion $f(x)$ over interval from	son's quadrature, with a default	quadrature with default absolute
	a to b	absolute tolerance of 10^{-6} . To	and relative error tolerances being
		specify absolute tolerance, use	
		quad(f,a,b,tol)	$(\epsilon_{\rm mach})^{0.25} \approx 1.22 \times 10^{-4}$. Tol-
			erances can be specified by using
			<pre>integrate(f,a,b, rel.tol=tol1,</pre>
			abs.tol=tol2). Note that the func-
			tion f must be written to work even
			when given a vector of x values as its
			argument.
173	Simple trapezoidal numerical	trapz(x,y)	<pre>sum(diff(x)*(y[-length(y)]+</pre>
	integration using (x, y) values		y[-1])/2)
	in vectors \mathbf{x} and \mathbf{y}		

3.7 Curve fitting

No.	Description	Matlab	R
174	Fit the line $y = c_1 x + c_0$ to data in vectors \mathbf{x} and \mathbf{y} .	p = polyfit(x,y,1)	p = coef(lm(y ~ x))
		The return vector \mathbf{p} has the coefficients in descending order, i.e. $\mathbf{p}(1)$ is c_1 , and $\mathbf{p}(2)$ is c_0 .	The return vector \mathbf{p} has the coefficients in ascending order, i.e. $\mathbf{p}[1]$ is c_0 , and $\mathbf{p}[2]$ is c_1 .
175	Fit the quadratic polynomial $y = c_2 x^2 + c_1 x + c_0$ to data in vectors \mathbf{x} and \mathbf{y} .	p = polyfit(x,y,2)	$p = coef(lm(y ~x + I(x^2)))$
		The return vector \mathbf{p} has the coefficients in descending order, i.e. $\mathbf{p(1)}$ is c_2 , $\mathbf{p(2)}$ is c_1 , and $\mathbf{p(3)}$ is c_0 .	The return vector \mathbf{p} has the coefficients in ascending order, i.e. $\mathbf{p[1]}$ is c_0 , $\mathbf{p[2]}$ is c_1 , and $\mathbf{p[3]}$ is c_2 .
176	Fit n^{th} degree polynomial $y = c_n x^n + c_{n-1} x^{n-1} + \ldots + c_1 x + c_0$ to data in vectors \mathbf{x}	<pre>p = polyfit(x,y,n)</pre>	No simple built-in way. But this will work: coef(lm(as.formula(paste('y~',paste('I(x~',1:n,')',
	and \mathbf{y} .	The return vector \mathbf{p} has the coefficients in descending order, $\mathbf{p}(1)$ is c^n , $\mathbf{p}(2)$ is c^{n-1} , etc.	sep='',collapse='+')))) This more concise "lower-level" method will also work: coef(lm.fit(outer(x,0:n,'^'),y)) Note that both of the above return the coefficients in ascending order. Also see the polyreg function in the mda package (see item 331 for how to install/load packages).
177	Fit the quadratic polynomial with zero intercept, $y = c_2x^2 + c_1x$ to data in vectors \mathbf{x} and \mathbf{y} .	(I don't know a simple way do this in MATLAB, other than to write a function which computes the sum of squared residuals and use fmin- search on that function. There is likely an easy way to do it in the Statistics Toolbox.)	p=coef(lm(y ~ -1 + x + I(x^2))) The return vector \mathbf{p} has the coefficients in ascending order, i.e. $\mathbf{p}[1]$ is c_1 , and $\mathbf{p}[2]$ is c_2 .
178	Fit natural cubic spline $(S''(x)) = 0$ at both endpoints) to points (x_i, y_i) whose coordinates are in vectors \mathbf{x} and \mathbf{y} ; evaluate at points whose x coordinates are in vector $\mathbf{x}\mathbf{x}$, storing corresponding y 's in $\mathbf{y}\mathbf{y}$	<pre>pp=csape(x,y,'variational'); yy=ppval(pp,xx) but note that csape is in MATLAB's Spline Toolbox</pre>	<pre>tmp=spline(x,y,method='natural', xout=xx); yy=tmp\$y</pre>
179	Fit cubic spline using Forsythe, Malcolm and Moler method (third derivatives at endpoints match third derivatives of exact cubics through the four points at each end) to points (x_i, y_i) whose coordinates are in vectors \mathbf{x} and \mathbf{y} ; evaluate at points whose x coordinates are in vector $\mathbf{x}\mathbf{x}$, storing corresponding y 's in $\mathbf{y}\mathbf{y}$	I'm not aware of a function to do this in Matlab	<pre>tmp=spline(x,y,xout=xx); yy=tmp\$y</pre>

No.	Description	Matlab	R
180	Fit cubic spline such that	<pre>pp=csape(x,y); yy=ppval(pp,xx)</pre>	I'm not aware of a function to do this
	first derivatives at endpoints	but csape is in Matlab's Spline	in R
	match first derivatives of ex-	Toolbox	
	act cubics through the four		
	points at each end) to points		
	(x_i, y_i) whose coordinates are		
	in vectors \mathbf{x} and \mathbf{y} ; evaluate		
	at points whose x coordinates		
	are in vector $\mathbf{x}\mathbf{x}$, storing cor-		
	responding y 's in yy		
181	Fit cubic spline with periodic	<pre>pp=csape(x,y,'periodic');</pre>	tmp=spline(x,y,method=
	boundaries, i.e. so that first	yy=ppval(pp,xx) but csape is in	'periodic', xout=xx); yy=tmp\$y
	and second derivatives match	Matlab's Spline Toolbox	
	at the left and right ends		
	(the first and last y values		
	of the provided data should		
	also agree), to points (x_i, y_i)		
	whose coordinates are in vec-		
	tors \mathbf{x} and \mathbf{y} ; evaluate at		
	points whose x coordinates		
	are in vector $\mathbf{x}\mathbf{x}$, storing cor-		
	responding y 's in yy		
182	Fit cubic spline with "not-	<pre>yy=spline(x,y,xx)</pre>	I'm not aware of a function to do this
	a-knot" conditions (the first		in R
	two piecewise cubics coincide,		
	as do the last two), to points		
	(x_i, y_i) whose coordinates are		
	in vectors \mathbf{x} and \mathbf{y} ; evaluate		
	at points whose x coordinates		
	are in vector $\mathbf{x}\mathbf{x}$, storing cor-		
	responding y 's in yy		

4 Conditionals, control structure, loops

No.	Description	Matlab	R
183	"for" loops over values in a vector v (the vector v is of-	for i=v	If only one command inside the loop: for (i in v)
	ten constructed via a:b)	command1 command2	command
		end	or
			for (i in v) command
			If multiple commands inside the loop:
			<pre>for (i in v) { command1 command2 }</pre>

No. Description	MATLAB	R
184 "if" statemen clause	if cond command1 command2 end	If only one command inside the clause: if (cond) command or if (cond) command If multiple commands: if (cond) { command1 command2 }
185 "if/else" state	if cond command1 command2 else command3 command4	If one command in clauses: if (cond) command1 else command2 or if (cond) cmd1 else cmd2 If multiple commands: if (cond) { command1 command2 } else { command3 command4 } Warning: the "else" must be on the same line as command1 or the "}" (when typed interactively at the command prompt), otherwise R thinks the "if" statement was finished and gives an error. R does not have an "elseif" statement.

 $\label{logical comparisons} \mbox{ Logical comparisons which can be used on scalars in "if" statements, or which operate element-by-element on vectors/matrices:$

MATLAB	R	Description
x < a	x < a	True if x is less than a
x > a	x > a	True if x is greater than a
x <= a	x <= a	True if x is less than or equal to a
x >= a	x >= a	True if x is greater than or equal to a
x == a	x == a	True if x is equal to a
x ~= a	x != a	True if x is not equal to a

Scalar logical operators:

Description	Matlab	R
a AND b	a && b	a && b
a OR b	a b	a b
a XOR b	xor(a,b)	xor(a,b)
NOT a	~a	!a

The && and | | operators are short-circuiting, i.e. && stops as soon as any of its terms are FALSE, and | | stops as soon as any of its terms are TRUE.

Matrix logical operators (they operate element-by-element):

Description	Matlab	R
a AND b	a & b	a & b
a OR b	a b	a b
a XOR b	xor(a,b)	xor(a,b)
NOT a	~a	!a

No.	Description	Matlab	R
186	To test whether a scalar value	if $((x > 4) \&\& (x <= 7))$	if ((x > 4) && (x <= 7))
	\mathbf{x} is between 4 and 7 (inclu-		
	sive on the upper end)		
187	To count how many values in	sum((x > 4) & (x <= 7))	sum((x > 4) & (x <= 7))
	the vector \mathbf{x} are between 4		
	and 7 (inclusive on the upper		
	end)		
188	Test whether all values in	all(v)	all(v)
	a logical/boolean vector are		
	TRUE		
189	Test whether any values in	any(v)	any(v)
	a logical/boolean vector are		
	TRUE		

No.	Description	MATLAB	R
190	"while" statements to do iteration (useful when you don't know ahead of time how many iterations you'll need). E.g. to add uniform random numbers between 0 and 1 (and their squares) until their sum is greater than 20:	<pre>mysum = 0; mysumsqr = 0; while (mysum < 20) r = rand; mysum = mysum + r; mysumsqr = mysumsqr + r^2; end</pre>	<pre>mysum = 0 mysumsqr = 0 while (mysum < 20) { r = runif(1) mysum = mysum + r mysumsqr = mysumsqr + r^2 } (As with "if" statements and "for" loops, the curly brackets are not necessary if there's only one statement inside the "while" loop.)</pre>
191	More flow control: these commands exit or move on to the next iteration of the innermost while or for loop, respectively.	break and continue	break and next
192	"Switch" statements for integers	<pre>switch (x) case 10 disp('ten') case {12,13} disp('dozen (bakers?)') otherwise disp('unrecognized') end</pre>	R doesn't have a switch statement capable of doing this. It has a function which is fairly limited for integers, but can which do string matching. See ?switch for more. But a basic example of what it can do for integers is below, showing that you can use it to return different expressions based on whether a value is 1, 2, mystr = switch(x, 'one', 'two', 'three'); print(mystr) Note that switch returns NULL if x is larger than 3 in the above case. Also, continuous values of x will be truncated to integers.

5 Functions, ODEs

No.	Description	Matlab	R
193	$\begin{array}{ccc} \text{Implement} & \text{a} & \text{function} \\ \textbf{add}(\mathbf{x}, \mathbf{y}) & & & & & \\ \end{array}$	Put the following in add.m: function retval=add(x,y) retval = x+y; Then you can do e.g. add(2,3)	Enter the following, or put it in a file and source that file: add = function(x,y) { return(x+y) } Then you can do e.g. add(2,3). Note, the curly brackets aren't needed if your function only has one line. Also, the return keyword is optional in the above example, as the value of the last expression in a function gets returned, so just x+y would work too.
194	Implement a function $f(x,y,z)$ which returns multiple values, and store those return values in variables u and v	Write function as follows: function [a,b] = f(x,y,z) a = x*y+z; b=2*sin(x-z); Then call the function by doing: [u,v] = f(2,8,12)	Write function as follows: f = function(x,y,z) { a = x*y+z; b=2*sin(x-z) return(list(a,b)) } Then call the function by doing: tmp=f(2,8,12); u=tmp[[1]]; v=tmp[[2]]. The above is most general, and will work even when u and v are different types of data. If they are both scalars, the function could simply return them packed in a vector, i.e. return(c(a,b)). If they are vectors of the same size, the function could return them packed together into the columns of a matrix, i.e. return(cbind(a,b)).

No.	Description	Matlab	R
195	Numerically solve ODE	First implement function	First implement function
130	dx/dt = 5x from $t = 3$ to $t = 12$ with initial condition $x(3) = 7$	function retval=f(t,x) retval = 5*x; Then do ode45(@f,[3,12],7) to plot solution, or [t,x]=ode45(@f,[3,12],7) to get back vector t containing time values and vector x containing correspond- ing function values. If you want	<pre>f = function(t,x,parms) { return(list(5*x)) } Then do y=lsoda(7, seq(3,12, 0.1), f,NA) to obtain solution values at times 3,3.1,3.2,,11.9,12. The first column of y, namely y[,1]</pre>
100		function values at specific times, e.g. 3,3.1,3.2,,11.9,12, you can do [t,x]=ode45(@f,3:0.1:12,7). Note: in older versions of MATLAB, use 'f' instead of @f.	contains the time values; the second column y[,2] contains the corresponding function values. Note: lsoda is part of the deSolve package (see item 331 for how to install/load packages).
196	Numerically solve system of ODEs $dw/dt = 5w$, $dz/dt = 3w + 7z$ from $t = 3$ to $t = 12$ with initial conditions $w(3) = 7$, $z(3) = 8.2$	First implement function function retval=myfunc(t,x) $w = x(1); z = x(2);$ $retval = zeros(2,1);$ $retval(1) = 5*w;$ $retval(2) = 3*w + 7*z;$ Then do ode45(@myfunc,[3,12],[7; 8.2]) to plot solution, or $[t,x]=ode45(@myfunc,[3,12],[7;$ 8.2]) to get back vector t containing time values and matrix x, whose first column containing corresponding $w(t)$ values and second column contains $z(t)$ values. If you want function values at specific times, e.g. 3,3.1,3.2,,11.9,12, you can do $[t,x]=ode45(@myfunc,3:0.1:12,[7;8.2])$. Note: in older versions of	First implement function myfunc = function(t,x,parms) { $w = x[1]; z = x[2];$ return(list(c(5*w, 3*w+7*z))) } Then do y=lsoda(c(7,8.2), seq(3,12, 0.1), myfunc,NA) to obtain solution values at times 3,3.1,3.2,,11.9,12. The first column of y, namely y[,1] contains the time values; the second column y[,2] contains the corresponding values of $w(t)$; and the third column contains $z(t)$. Note: lsoda is part of the deSolve package (see item 331 for how to install/load packages).
197	Pass parameters such as $r = 1.3$ and $K = 50$ to an ODE function from the command line, solving $dx/dt = rx(1 - x/K)$ from $t = 0$ to $t = 20$ with initial condition $x(0) = 2.5$.	MATLAB, use 'f' instead of @f. First implement function function retval=func2(t,x,r,K) retval = r*x*(1-x/K) Then do ode45(@func2,[0 20], 2.5, [], 1.3, 50). The empty matrix is necessary between the initial condition and the beginning of your extra parameters.	First implement function func2=function(t,x,parms) { r=parms[1]; K=parms[2] return(list(r*x*(1-x/K))) } Then do y=lsoda(2.5,seq(0,20,0.1), func2,c(1.3,50)) Note: lsoda is part of the deSolve package (see item 331 for how to in- stall/load packages).

6 Probability and random values

No.	Description	Matlab	R
198	Generate a continuous uniform random value between 0 and 1	rand	runif(1)
199	Generate vector of n uniform random vals between 0 and 1	rand(n,1) or rand(1,n)	runif(n)
200	Generate $m \times n$ matrix of uniform random values between 0 and 1	rand(m,n)	<pre>matrix(runif(m*n),m,n) or just matrix(runif(m*n),m)</pre>
201	Generate $m \times n$ matrix of continuous uniform random values between a and b	a+rand(m,n)*(b-a) or if you have the Statistics toolbox then unifrnd(a,b,m,n)	<pre>matrix(runif(m*n,a,b),m)</pre>
202	Generate a random integer between 1 and k	floor(k*rand) + 1	floor(k*runif(1)) + 1 Note: sample(k)[1] would also work, but I believe in general will be less efficient, because that actually generates many random numbers and then just uses one of them.
203	Generate $m \times n$ matrix of discrete uniform random integers between 1 and k	floor(k*rand(m,n))+1 or if you have the Statistics toolbox then unidrnd(k,m,n)	floor(k*matrix(runif(m*n),m))+1
204	Generate $m \times n$ matrix where each entry is 1 with probability p , otherwise is 0	<pre>(rand(m,n)<p)*1 (true="" 1="" also="" back="" by="" could="" do="" double(rand(m,n)<p)<="" false)="" into="" logical="" multiplying="" note:="" numeric="" pre="" re-="" sult="" the="" turns="" values.="" you=""></p)*1></pre>	(matrix(runif(m,n),m) <p)*1 (Note: multiplying by 1 turns the logical (true/false) result back into numeric values; using as.numeric() to do it would lose the shape of the matrix.)</p)*1
205	Generate $m \times n$ matrix where each entry is a with probability p , otherwise is b	b + (a-b)*(rand(m,n) <p)< td=""><td>b + (a-b)*(matrix(runif(m,n),m)<p)< td=""></p)<></td></p)<>	b + (a-b)*(matrix(runif(m,n),m) <p)< td=""></p)<>
206	Generate a random integer between a and b inclusive	floor((b-a+1)*rand)+a or if you have the Statistics toolbox then unidrnd(b-a+1)+a-1	floor((b-a+1)*runif(1))+a
207	Flip a coin which comes up heads with probability p , and perform some action if it does come up heads	<pre>if (rand < p) some commands end</pre>	<pre>if (runif(1) < p) { some commands }</pre>
208	Generate a random permutation of the integers $1, 2, \ldots, n$	randperm(n)	sample(n)
209	Generate a random selection of k unique integers between 1 and n (i.e. sampling without replacement)	<pre>[s,idx]=sort(rand(n,1)); ri=idx(1:k) or another way is ri=randperm(n); ri=ri(1:k). Or if you have the Statistics Toolbox, then randsample(n,k)</pre>	ri=sample(n,k)
210	Choose k values (with replacement) from the vector \mathbf{v} , storing result in \mathbf{w}	L=length(v); w=v(floor(L*rand(k,1))+1) Or, if you have the Statistics Toolbox, w=randsample(v,k)	w=sample(v,k,replace=TRUE)

No.	Description	Matlab	R
211	Choose k values (without re-	<pre>L=length(v); ri=randperm(L);</pre>	w=sample(v,k,replace=FALSE)
	placement) from the vector \mathbf{v} ,	ri=ri(1:k); w=v(ri) Or, if	
	storing result in w	you have the Statistics Toolbox,	
		w=randsample(v,k)	
212	Set the random-number gen-	rand('state', 12) Note: begin-	set.seed(12)
	erator back to a known state	ning in Matlab 7.7, use this in-	
	(useful to do at the beginning	stead: RandStream('mt19937ar',	
	of a stochastic simulation	'Seed', 12) though the previous	
	when debugging, so you'll get	method is still supported for now.	
	the same sequence of random		
	numbers each time)		

Note that the "*rnd," "*pdf," and "*cdf" functions described below are all part of the MATLAB Statistics Toolbox, and not part of the core MATLAB distribution.

	Statistics Toolbox, and not part of the core MATLAB distribution.			
No.	Description	Matlab	R	
213	Generate a random value	binornd(n,p)	rbinom(1,n,p)	
	from the binomial (n, p) dis-			
	tribution			
214	Generate a random value	poissrnd(lambda)	rpois(1,lambda)	
	from the Poisson distribution			
	with parameter λ			
215	Generate a random value	exprnd(mu) or -mu*log(rand) will	rexp(1, 1/mu)	
	from the exponential distri-	work even without the Statistics		
	bution with mean μ	Toolbox.		
216	Generate a random value	unidrnd(k) or floor(rand*k)+1	sample(k,1)	
	from the discrete uniform dis-	will work even without the Statistics		
	tribution on integers $1 \dots k$	Toolbox.		
217	Generate n iid random values	unidrnd(k,n,1) or	sample(k,n,replace=TRUE)	
	from the discrete uniform dis-	floor(rand(n,1)*k)+1 will work		
	tribution on integers $1 \dots k$	even without the Statistics Toolbox.		
218	Generate a random value	unifrnd(a,b) or (b-a)*rand + a	runif(1,a,b)	
	from the continuous uniform	will work even without the Statistics		
	distribution on the interval	Toolbox.		
	(a,b)			
219	Generate a random value	normrnd(mu,sigma) or	rnorm(1,mu,sigma)	
	from the normal distribution	mu + sigma*randn will work		
	with mean μ and standard	even without the Statistics Toolbox.		
	deviation σ			
220	Generate a random vector	mnrnd(n,p)	rmultinom(1,n,p)	
	from the multinomial distri-			
	bution, with n trials and			
	probability vector p			
221	Generate j random vectors	mnrnd(n,p,j)	rmultinom(j,n,p)	
	from the multinomial distri-	The vectors are returned as rows of	The vectors are returned as columns	
	bution, with n trials and	a matrix	of a matrix	
	probability vector p			

Notes:

• The Matlab "*rnd" functions above can all take additional \mathbf{r} , \mathbf{c} arguments to build an $r \times c$ matrix of iid random values. E.g. $\mathtt{poissrnd(3.5,4,7)}$ for a 4×7 matrix of iid values from the Poisson distribution with mean $\lambda = 3.5$. The $\mathtt{unidrnd(k,n,1)}$ command above is an example of this, to generate a $k \times 1$ column vector.

• The first parameter of the R "r*" functions above specifies how many values are desired. E.g. to generate 28 iid random values from a Poisson distribution with mean 3.5, use rpois(28,3.5). To get a 4 × 7 matrix of such values, use matrix(rpois(28,3.5),4).

No.	Description	Matlab	R
222	Compute probability that	binopdf(x,n,p) or	dbinom(x,n,p)
	a random variable from the	$nchoosek(n,x)*p^x*(1-p)^(n-x)$	
	Binomial (n, p) distribution	will work even without the Statistics	
	has value \mathbf{x} (i.e. the density,	Toolbox, as long as \mathbf{n} and \mathbf{x} are	
	or pdf).	non-negative integers and $0 \leq \mathbf{p}$	
		≤ 1 .	
223	Compute probability that a	poisspdf(x,lambda) or	dpois(x,lambda)
	random variable from the	exp(-lambda)*lambda^x /	
	Poisson(λ) distribution has	factorial(x) will work even	
	value \mathbf{x} .	without the Statistics Toolbox, as	
		long as \mathbf{x} is a non-negative integer	
		and $lambda \ge 0$.	
224	Compute probability density	exppdf(x,mu) or	dexp(x,1/mu)
	function at \mathbf{x} for a random	(x>=0)*exp(-x/mu)/mu will work	
	variable from the exponential	even without the Statistics Toolbox,	
	distribution with mean μ .	as long as mu is positive.	
225	Compute probability density	normpdf(x,mu,sigma) or	dnorm(x,mu,sigma)
	function at \mathbf{x} for a random	$\exp(-(x-mu)^2/(2*sigma^2))/$	
	variable from the Normal dis-	(sqrt(2*pi)*sigma) will work even	
	tribution with mean μ and	without the Statistics Toolbox.	
	standard deviation σ .		
226	Compute probability density	unifpdf(x,a,b) or	<pre>dunif(x,a,b)</pre>
	function at \mathbf{x} for a random	((x>=a)&&(x<=b))/(b-a) will	
	variable from the continuous	work even without the Statistics	
	uniform distribution on inter-	Toolbox.	
	$\operatorname{val}(a,b).$		
227	Compute probability that a	unidpdf(x,n) or ((x==floor(x))	((x==round(x)) && (x >= 1) &&
	random variable from the dis-	&& (x>=1)&&(x<=n))/n will work	(x <= n))/n
	crete uniform distribution on	even without the Statistics Toolbox,	
	integers $1 \dots n$ has value \mathbf{x} .	as long as n is a positive integer.	
228	Compute probability that	mnpdf(x,p)	dmultinom(x,prob=p)
	a random vector from the	Note: vector p must sum to one.	
	multinomial distribution	Also, \mathbf{x} and \mathbf{p} can be vectors of	
	with probability vector \vec{p} has	length k , or if one or both are $m \times k$	
	the value \vec{x}	matrices then the computations are	
	N	performed for each row.	

Note: one or more of the parameters in the above "*pdf" (MATLAB) or "d*" (R) functions can be vectors, but they must be the same size. Scalars are promoted to arrays of the appropriate size.

The corresponding CDF functions are below:

No.	Description Description	Matlab	R
229	Compute probability that a	binocdf(x,n,p). Without the	pbinom(x,n,p)
	random variable from the	Statistics Toolbox, as long	
	Binomial (n, p) distribution is	as n is a non-negative in-	
	less than or equal to \mathbf{x} (i.e.	teger, this will work: $r =$	
	the cumulative distribution	<pre>0:floor(x); sum(factorial(n)./</pre>	
	function, or cdf).	<pre>(factorial(r).*factorial(n-r))</pre>	
		.*p.^r.*(1-p).^(n-r)). (Un-	
		fortunately, Matlab's nchoosek	
		function won't take a vector argu-	
		ment for \mathbf{k} .)	
230	Compute probability that a	poisscdf(x,lambda). With-	<pre>ppois(x,lambda)</pre>
	random variable from the	out the Statistics Toolbox, as	
	Poisson(λ) distribution is less	$long$ as $lambda \ge 0$, this	
	than or equal to \mathbf{x} .	<pre>will work: r = 0:floor(x);</pre>	
		<pre>sum(exp(-lambda)*lambda.^r</pre>	
		./factorial(r))	
231	Compute cumulative distri-	expcdf(x,mu) or	pexp(x,1/mu)
	bution function at \mathbf{x} for a	(x>=0)*(1-exp(-x/mu)) will	
	random variable from the ex-	work even without the Statistics	
	ponential distribution with	Toolbox, as long as mu is positive.	
	mean μ .		
232	Compute cumulative distri-	normcdf(x,mu,sigma) or 1/2 -	pnorm(x,mu,sigma)
	bution function at \mathbf{x} for a ran-	erf(-(x-mu)/(sigma*sqrt(2)))/2	
	dom variable from the Nor-	will work even without the Statis-	
	mal distribution with mean μ	tics Toolbox, as long as sigma is	
	and standard deviation σ .	positive.	
233	Compute cumulative distri-	unifcdf(x,a,b) or	<pre>punif(x,a,b)</pre>
	bution function at \mathbf{x} for a ran-	(x>a)*(min(x,b)-a)/(b-a) will	
	dom variable from the contin-	work even without the Statistics	
	uous uniform distribution on	Toolbox, as long as $\mathbf{b} > \mathbf{a}$.	
	interval (a, b) .		
234	Compute probability that a	unidcdf(x,n) or	(x>=1)*min(floor(x),n)/n
	random variable from the dis-	(x>=1)*min(floor(x),n)/n will	
	crete uniform distribution on	work even without the Statistics	
	integers $1 \dots n$ is less than or	Toolbox, as long as \mathbf{n} is a positive	
	equal to \mathbf{x} .	integer.	

7 Graphics

7.1 Various types of plotting

No.	Description	Matlab	R	
235	Create a new figure window	figure	dev.new() Notes: internally, on Windows this calls windows(), on MacOS it calls quartz(), and on Linux it calls X11(). X11() is also available on MacOS; you can tell R to use it by default by doing options(device='X11'). In R sometime after 2.7.0, X11 graphics started doing antialising by default, which makes plots look smoother but takes longer to draw. If you are using X11 graphics in R and notice that figure plotting is extremely slow (especially if making many plots), do this before calling dev.new(): X11.options(type='Xlib') or X11.options(antialias='none'). Or just use e.g. X11(type='Xlib') to make new figure windows. They are uglier (lines are more jagged), but render much more quickly.	
236	Select figure number n	figure(n) (will create the figure if it doesn't exist)	dev.set(n) (returns the actual device selected; will be different from n if there is no figure device with number n)	
237	Determine which figure window is currently active	gcf	dev.cur()	
238	List open figure windows	get(0,'children') (The 0 handle refers to the root graphics object.)	<pre>dev.list()</pre>	
239	Close figure window(s)	close to close the current figure window, close(n) to close a specified figure, and close all to close all figures	<pre>dev.off() to close the currently ac- tive figure device, dev.off(n) to close a specified one, and graphics.off() to close all figure devices.</pre>	
240	Plot points using open circles	plot(x,y,'o')	plot(x,y)	
241	Plot points using solid lines	plot(x,y)	plot(x,y,type='1') (Note: that's a lower-case 'L', not the number 1)	
242	Plotting: color, point markers, linestyle	<pre>plot(x,y,str) where str is a string specifying color, point marker, and/or linestyle (see table below) (e.g. 'gs' for green squares with dashed line)</pre>	<pre>plot(x,y,type=str1, pch=arg2,col=str3, lty=arg4)</pre>	
243	Diatting with logowithmic	gomilogy gomilogy and letter	See tables below for possible values of the 4 parameters	
243	Plotting with logarithmic axes	semilogx, semilogy, and loglog functions take arguments like plot , and plot with logarithmic scales for x, y , and both axes, respectively	plot(, log='x'), plot(, log='y'), and plot(, log='xy') plot with logarithmic scales for x, y, and both axes, respectively	

No.	Description	Matlab	R
244	Make bar graph where the x coordinates of the bars are in \mathbf{x} , and their heights are in \mathbf{y}	bar(x,y) Or just bar(y) if you only want to specify heights. Note: if A is a matrix, bar(A) interprets each column as a separate set of observations, and each row as a different observation within a set. So a 20×2 matrix is plotted as 2 sets of 20 observations, while a 2×20 matrix is plotted as 20 sets of 2 observations.	Can't do this in R; but barplot(y) makes a bar graph where you specify the heights, barplot(y,w) also specifies the widths of the bars, and hist can make plots like this too.
245	Make histogram of values in \mathbf{x}	hist(x)	hist(x)
246	Given vector \mathbf{x} containing discrete values, make a bar graph where the x coordinates of bars are the values, and heights are the counts of how many times the values appear in \mathbf{x}	<pre>v=unique(x); c=hist(x,v); bar(v,c)</pre>	<pre>barplot(table(x))</pre>
247	Given vector \mathbf{x} containing continuous values, lump the data into k bins and make a histogram / bar graph of the binned data	<pre>[c,m] = hist(x,k); bar(m,c) or for slightly different plot style use hist(x,k)</pre>	<pre>hist(x,seq(min(x), max(x), length.out=k+1))</pre>
248	Make a plot containing errorbars of height s above and below (x, y) points	errorbar(x,y,s)	errbar(x,y,y+s,y-s) Note: errbar is part of the Hmisc package (see item 331 for how to install/load packages).
249	Make a plot containing errorbars of height a above and b below (x, y) points	errorbar(x,y,b,a)	errbar(x,y,y+a,y-b) Note: errbar is part of the Hmisc package (see item 331 for how to install/load packages).
250	Other types of 2-D plots	stem(x,y) and stairs(x,y) for other types of 2-D plots. polar(theta,r) to use polar coordinates for plotting.	pie(v)

No.	Description	Matlab	R
251	Make a 3-D plot of some data	plot3(x,y,z) This works much like	cloud(z~x*y) You can also use
	points with given x, y, z co-	plot, as far as plotting symbols, line-	arguments pch and col as with
	ordinates in the vectors \mathbf{x} , \mathbf{y} ,	types, and colors.	plot. To make a 3-D plot with
	and \mathbf{z} .	types, and colors.	lines, do cloud(z~x*y,type='1',
	and Z.		panel.cloud=panel.3dwire)
252	Surface plot of data in matrix		paner.croud-paner.odwire)
202	A		
	A	surf(A)	persp(A)
		You can then click on the small	You can include shading in the im-
		curved arrow in the figure window	age via e.g. persp(A,shade=0.5).
		(or choose "Rotate 3D" from the	There are two viewing angles you
		"Tools" menu), and then click and	can also specify, among other pa-
		drag the mouse in the figure to ro-	rameters, e.g. persp(A, shade=0.5,
		tate it in three dimensions.	theta=50, phi=35).
253	Surface plot of $f(x,y) =$		
	$sin(x+y)\sqrt{y}$ for 100 values		(2.42.7
	of x between 0 and 10, and	x = linspace(0,10,100);	x = seq(0,10,len=100)
	90 values of y between 2 and	y = linspace(2,8,90);	y = seq(2,8,len=90)
	8	[X,Y] = meshgrid(x,y);	f = function(x,y)
	Č	Z = sin(X+Y).*sqrt(Y);	return(sin(x+y)*sqrt(y))
		<pre>surf(X,Y,Z)</pre>	z = outer(x,y,f)
		shading flat	persp(x,y,z)
254	Other ways of plotting the	mesh(X,Y,Z), surfc(X,Y,Z),	contour(x,y,z) Or do
204	data from the previous com-	surfl(X,Y,Z), contour(X,Y,Z),	s=expand.grid(x=x,y=y), and
	mand	pcolor(X,Y,Z),	then wireframe(z~x*y,s) or
	iiidiid	waterfall(X,Y,Z). Also see the	wireframe(z~x*y,s,shade=TRUE)
		slice command.	(Note: wireframe is part of the
		biios communa.	lattice package; see item 331 for how
			to load packages). If you have vectors
			x , y , and z all the same length, you
			can also do symbols(x,y,z).
255	Set axis ranges in a figure	axis([x1 x2 y1 y2])	You have to do this when
	window		you make the plot, e.g.
			plot(x,y,xlim=c(x1,x2),
			ylim=c(y1,y2))
256	Add title to plot	<pre>title('somestring')</pre>	title(main='somestring')
	-		adds a main title,
			title(sub='somestring') adds
			a subtitle. You can also include
			main= and sub= arguments in a
			plot command.
257	Add axis labels to plot	xlabel('somestring') and	title(xlab='somestring',
		ylabel('somestring')	ylab='anotherstr'). You can
		J	also include xlab = and ylab =
			arguments in a plot command.
			argamento in a piot command.

No.	Description	MATLAB	R
258	Include Greek letters or symbols in plot axis labels	You can use basic TeX commands, e.g. plot(x,y); xlabel('\phi^2 + \mu_{i,j}') or xlabel('fecundity \phi') See also help tex and parts of doc text_props for more about building labels using general LaTeX commands	<pre>plot(x,y,xlab= expression(phi^2 + mu['i,j'])) or plot(x,y,xlab=expression(paste('fecundity ', phi))) See also help(plotmath) and p. 98 of the R Graphics book by Paul Murrell for more.</pre>
259	Change font size to 16 in plot labels	For the legends and numerical axis labels, use set(gca, 'FontSize', 16), and for text labels on axes do e.g. xlabel('my x var', 'FontSize', 16)	For on-screen graphics, do par(ps=16) followed by e.g. a plot command. For PostScript or PDF plots, add a pointsize=16 argument, e.g. pdf('myfile.pdf', width=8, height=8, pointsize=16) (see items 275 and 276)
260	Add grid lines to plot	grid on (and grid off to turn off)	grid() Note that if you'll be printing the plot, the default style for grid-lines is to use gray dotted lines, which are almost invisible on some printers. You may want to do e.g. grid(lty='dashed', col='black') to use black dashed lines which are easier to see.
261	Add a text label to a plot	<pre>text(x,y,'hello')</pre>	text(x,y,'hello')
262	Add set of text labels to a plot. xv and yv are vectors.	<pre>s={'hi', 'there'}; text(xv,yv,s)</pre>	s=c('hi', 'there'); text(xv,yv,s)
263	Add an arrow to current plot, with tail at (xt, yt) and head at (xh, yh)	annotation('arrow', [xt xh], [yt yh]) Note: coordinates should be normalized figure coordinates, not coordinates within your displayed axes. Find and download from The Mathworks the file dsxy2figxy.m which converts for you, then do this: [fx,fy]=dsxy2figxy([xt xh], [yt yh]); annotation('arrow', fx, fy)	arrows(xt, yt, xh, yh)
264	Add a double-headed arrow to current plot, with coordinates $(x0, y0)$ and $(x1, y1)$	annotation('doublearrow', [x0 x1], [y0 y1]) See note in previous item about normalized figure coordinates.	arrows(x0, y0, x1, y1, code=3)
265	Add figure legend to top-left corner of plot	legend('first', 'second', 'Location', 'NorthWest')	<pre>legend('topleft', legend=c('first', 'second'), col=c('red', 'blue'), pch=c('*','o'))</pre>

MATLAB note: sometimes you build a graph piece-by-piece, and then want to manually add a legend which doesn't correspond with the order you put things in the plot. You can manually construct a legend by plotting "invisible" things, then building the legend using them. E.g. to make a legend with black stars and solid lines, and red circles and dashed lines: h1=plot(0,0,'k*-'); set(h1,'Visible', 'off'); h2=plot(0,0,'k*-'); set(h2,'Visible', 'off'); legend([h1 h2], 'blah, 'whoa'). Just be sure to choose coordinates for your "invisible" points within the current figure's axis ranges.

No.	Description	Matlab	R
266	Adding more things to a figure	hold on means everything plotted from now on in that figure window is added to what's already there. hold off turns it off. clf clears the figure and turns off hold.	points() and lines() work like plot, but add to what's already in the figure rather than clearing the figure first. points and lines are basically identical, just with different default plotting styles. Note: axes are not recalculated/redrawn when adding more things to a figure.
267	Plot multiple data sets at once	plot(x,y) where x and y are 2-D matrices. Each column of x is plotted against the corresponding column of y. If x has only one column, it will be re-used.	matplot(x,y) where x and y are 2-D matrices. Each column of x is plotted against the corresponding column of y. If x has only one column, it will be re-used.
268	Plot $\sin(2x)$ for x between 7 and 18	fplot('sin(2*x)', [7 18])	curve(sin(2*x), 7, 18, 200) makes the plot, by sampling the value of the function at 200 values between 7 and 18 (if you don't specify the number of points, 101 is the default). You could do this manually yourself via commands like tmpx=seq(7,18,len=200); plot(tmpx, sin(2*tmpx)).
269	Plot color image of integer values in matrix A	image(A) to use array values as raw indices into colormap, or imagesc(A) to automatically scale values first (these both draw row 1 of the matrix at the top of the image); or pcolor(A) (draws row 1 of the matrix at the bottom of the image). After using pcolor, try the commands shading flat or shading interp.	image(A) (it rotates the matrix 90 degrees counterclockwise: it draws row 1 of A as the left column of the image, and column 1 of A as the bottom row of the image, so the row number is the x coord and column number is the y coord). It also rescales colors. If you are using a colormap with k entries, but the value k does not appear in A , use image(A,zlim=c(1,k)) to avoid rescaling of colors. Or e.g. image(A,zlim=c(0,k-1)) if you want values 0 through $k-1$ to be plotted using the k colors.
270	Add colorbar legend to image plot	colorbar, after using image or pcolor.	Use filled.contour(A) rather than image(A), although it "blurs" the data via interpolation, or use levelplot(A) from the lattice package (see item 331 for how to load packages). To use a colormap with the latter, do e.g. levelplot(A,col.regions=terrain.colors(100)).
271	Set colormap in image	colormap(hot). Instead of hot, you can also use gray, flag, jet (the default), cool, bone, copper, pink, hsv, prism. By default, the length of the new colormap is the same as the currently-installed one; use e.g. colormap(hot(256)) to specify the number of entries.	image(A,col=terrain.colors(100)). The parameter 100 specifies the length of the colormap. Other colormaps are heat.colors(), topo.colors(), and cm.colors().

No.	Description	Matlab	R
272	Build your own colormap us-	Use an $n \times 3$ matrix; each row	Use a vector of hexadecimal strings,
	ing Red/Green/Blue triplets	gives R,G,B intensities between 0	each beginning with '#' and giving
		and 1. Can use as argument with	R,G,B intensities between 00 and FF.
		colormap. E.g. for 2 colors: mycmap	E.g. c('#80CC33','#3333B3'); can
		= [0.5 0.8 0.2 ; 0.2 0.2 0.7]	use as argument to col = parameter
			to image. You can build such a
			vector of strings from vectors of Red,
			Green, and Blue intensities (each
			between 0 and 1) as follows (for a
			2-color example): r=c(0.5,0.2);
			g=c(0.8,0.2); b=c(0.2,0.7);
			mycolors=rgb(r,g,b).

MATLAB plotting specifications, for use with plot, fplot, semilogx, semilogy, loglog, etc:

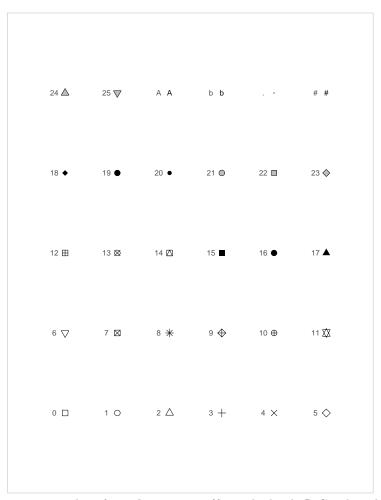
Symbol	Color	Symbol	Marker	Symbol	Linestyle
b	blue		point (.)	-	solid line
g	green	0	circle (o)	:	dotted line
r	red	х	cross(x)		dash-dot line
С	cyan	+	plus sign (+)		dashed line
m	magenta	*	asterisk (*)		
У	yellow	s	square (\Box)		
k	black	d	diamond (\lozenge)		
W	white	v	triangle (down) (∇)		
		^	triangle (up) (Δ)		
		<	triangle (left) (\triangleleft)		
		>	triangle (right) (\triangleright)		
		р	pentragram star		
		h	hexagram star		

R plotting specifications for \mathbf{col} (color), \mathbf{pch} (plotting character), and \mathbf{type} arguments, for use with \mathbf{plot} , $\mathbf{matplot}$, \mathbf{points} , and \mathbf{lines} :

col	Description	pch	Description	type	Description
'blue'	Blue	'a'	a (similarly for other	р	points
			characters, but see '.'		
			below for an exception		
'green'	Green	0	open square	1	lines
'red'	Red	1	open circle	Ъ	both
'cyan'	Cyan	2	triangle point-up	С	lines part only of "b"
'magenta'	Magenta	3	+ (plus)	0	lines, points overplotted
'yellow'	Yellow	4	\times (cross)	h	histogram-like lines
'black'	Black	5	diamond	s	steps
'#RRGGBB'	hexadecimal specifica-	6	triangle point-down	S	another kind of steps
	tion of Red, Green,				
	Blue				
(Other names)	See colors() for list of	, ,	rectangle of size 0.01	n	no plotting (can be use-
	available color names.		inch, 1 pixel, or 1 point		ful for setting up axis
			(1/72 inch) depending		ranges, etc.)
			on device		
			(See table on next page		
			for more)		

R plotting specifications for lty (line-type) argument, for use with plot, matplot, points, and lines:

lty	Description
0	blank
1	solid
2	dashed
3	dotted
4	dotdash
5	longdash
6	twodash



R plotting characters, i.e. values for ${f pch}$ argument (from the book R Graphics, by Paul Murrell, Chapman & Hall / CRC, 2006)

No.	Description	Matlab	R
No. 273	Divide up a figure window into smaller sub-figures	subplot(m,n,k) divides the current figure window into an $m \times n$ array of subplots, and draws in subplot number k as numbered in "reading order," i.e. left-to-right, top-to-bottom. E.g. subplot(2,3,4) selects the first sub-figure in the second row of a 2×3 array of sub-figures. You can do more complex things, e.g. subplot(5,5,[1 2 6 7]) selects the first two subplots in the first row, and first two subplots in the second row, i.e. gives you a bigger subplot within a 5×5 array of subplots. (If you that command followed by e.g. subplot(5,5,3) you'll see what's meant by that.)	There are several ways to do this, e.g. using layout or split.screen, although they aren't quite as friendly as MATLAB 's. E.g. if you let A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 3 \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will \\ 4 & 5 & 6 \end{bmatrix}, then layout(A) will take up the upper-left \\ 2 \times 2 \times 2 \times 4 \ti
274	Force graphics windows to update	drawnow (Matlab normally only updates figure windows when a script/function finishes and returns control to the Matlab prompt, or under a couple of other circumstances. This forces it to update figure windows to reflect any recent plotting commands.)	R automatically updates graphics windows even before functions/scripts finish executing, so it's not necessary to explictly request it. But note that some graphics functions (particularly those in the lattice package) don't display their results when called from scripts or functions; e.g. rather than levelplot() you need to doprint(levelplot()). Such functions will automatically display their plots when called interactively from the command prompt.

7.2 Printing/saving graphics

No.	Description	Matlab	R
275	To print/save to a PDF file named fname.pdf	print -dpdf fname saves the contents of currently active figure window	First do pdf('fname.pdf'). Then, do various plotting commands to make your image, as if you were plotting in a window. Finally, do dev.off() to close/save the PDF file. To print the contents of the active figure window, do dev.copy(device=pdf, file='fname.pdf'); dev.off(). (But this will not work if you've turned off the display list via dev.control(displaylist='inhibit').) You can also simply use dev.copy2pdf(file='fname.pdf').
276	To print/save to a PostScript file fname.ps or fname.eps	print -dps fname for black & white PostScript; print -dpsc fname for color PostScript; print -deps fname for black & white Encapsulated PostScript; print -depsc fname for color Encapsulated PostScript. The first two save to fname.ps, while the latter two save to fname.eps.	postscript('fname.eps'), followed by your plotting commands, followed by dev.off() to close/save the file. Note: you may want to use postscript('fname.eps', horizontal=FALSE) to save your figure in portrait mode rather than the default landscape mode. To print the contents of the active figure window, do dev.copy(device=postscript, file='fname.eps'); dev.off(). (But this will not work if you've turned off the display list via dev.control(displaylist='inhibit').) You can also include the horizontal=FALSE argument with dev.copy(). The command dev.copy2eps(file='fname.eps') also saves in portrait mode.
277	To print/save to a JPEG file fname.jpg with jpeg quality = 90 (higher quality looks better but makes the file larger)	print -djpeg90 fname	<pre>jpeg('fname.jpg',quality=90), followed by your plotting commands, followed by dev.off() to close/save the file.</pre>

7.3 Animating cellular automata / lattice simulations

No.	Description	Matlab	R
278	To display images of cellu-	Repeatedly use either pcolor or	If you simply call image repeatedly,
	lar automata or other lattice	image to display the data. Don't	there is a great deal of flicker-
	simulations while running in	forget to call drawnow as well, oth-	ing/flashing. To avoid this, after
	real time	erwise the figure window will not be	drawing the image for the first time
		updated with each image.	using e.g. image(A), from then
			on only use image(A,add=TRUE),
			which avoids redrawing the entire
			image (and the associated flicker).
			However, this will soon consume a
			great deal of memory, as all drawn
			images are saved in the image buffer.
			There are two solutions to that
			problem: (1) every k time steps,
			leave off the "add=TRUE" argument
			to flush the image buffer (and get
			occasional flickering), where you choose k to balance the flickering
			vs. memory-usage tradeoff; or
			(2) after drawing the first image,
			do dev.control(displaylist=
			'inhibit') to prohibit retaining the
			data. However, the latter solution
			means that after the simulation is
			done, the figure window will not be
			redrawn if it is resized, or temporarily
			obscured by another window. (A
			call to dev.control(displaylist=
			'enable') and then one final
			image(A) at the end of the sim-
			ulation will re-enable re-drawing
			after resizing or obscuring, without
			consuming extra memory.)

8 Working with files

No.	Description	Matlab	R
279	Create a folder (also known	mkdir dirname	dir.create('dirname')
	as a "directory")		
280	Set/change working directory	cd dirname	setwd('dirname')
281	See list of files in current	dir	dir()
	working directory		
282	Run commands in file 'foo.m'	foo	source('foo.R')
	or 'foo.R' respectively		
283	Read data from text file	A=load('data.txt') or	A=as.matrix(read.table(
	"data.txt" into matrix A	A=importdata('data.txt') Note	'data.txt')) This will ignore
		that both routines will ignore com-	comments (anything on a line
		ments (anything on a line following	following a "#" character). To ig-
		a "%" character)	nore comments indicated by "%",
			do A=as.matrix(read.table(
			'data.txt', comment.char='%'))
284	Read data from text file		A=as.matrix(read.table(
	"data.txt" into matrix A ,	<pre>tmp=importdata('data.txt',</pre>	'data.txt', skip=s))
	skipping the first \mathbf{s} lines of the	', ',s);	
	file	a=tmp.data	
		a omp. aaoa	
285	Write data from matrix A	save data.txt A -ascii	<pre>write(t(A), file='data.txt',</pre>
	into text file "data.txt"		ncolumn=dim(A)[2])
L			

9 Miscellaneous

9.1 Variables

No.	Description	Matlab	R
286	Assigning to variables	x = 5	$x \leftarrow 5 \text{ or } x = 5 \text{ Note: for compati-}$
			bility with S-plus, many people prefer
			the first form.
287	From within a function, as-	assignin('base', 'y', 7)	у <<- 7
	sign a value to variable \mathbf{y}		
	in the base environment (i.e.		
	the command prompt envi-		
	ronment)		
288	From within a function, ac-	evalin('base', 'y')	<pre>get('y', envir=globalenv())</pre>
	cess the value of variable \mathbf{y}		Though note that inside a function,
	in the base environment (i.e.		if there isn't a local variable \mathbf{y} , then
	the command prompt envi-		just the expression y will look for one
	ronment)		in the base environment, but if there
			is a local \mathbf{y} then that one will be used
			instead.
289	Short list of defined variables	who	ls()
290	Long list of defined variables	whos	ls.str()
291	See detailed info about the	whos ab	str(ab)
	variable ab		
292	See detailed info about all	whos *ab*	ls.str(pattern='ab')
	variables with "ab" in their		
	name		
293	Open graphical data editor,	openvar(A), or double-click on the	fix(A)
	to edit the value of variable	variable in the Workspace pane (if	
	A (useful for editing values in	it's being displayed) of your MAT-	
	a matrix, though it works for	LABdesktop	
20.4	non-matrix variables as well)	_	
294	Clear one variable	clear x	rm(x)
295	Clear two variables	clear x y	rm(x,y)
296	Clear all variables	clear all	rm(list=ls())
297	See what type of object \mathbf{x} is	class(x)	class(x) and typeof(x) give differ-
200	(77 • 11	77 • 11	ent aspects of the "type" of x
298	(Variable names)	Variable names must begin with a	Variable names may contain letters,
		letter, but after that they may con-	digits, the period, and the underscore
		tain any combination of letters, dig-	character. They cannot begin with a
		its, and the underscore character.	digit or underscore, or with a period
		Names are case-sensitive.	followed by a digit. Names are case-
200	D 1, C1 ,		sensitive.
299	Result of last command	ans contains the result of the last	.Last.value contains the result of
		command which did not assign its	the last command, whether or not its
		value to a variable. E.g. after 2+5;	value was assigned to a variable. E.g.
		x=3, then ans will contain 7.	after 2+5; x=3, then .Last.value will
			contain 3.

9.2 Strings and Misc.

No.	Description	Matlab	R
300	Line continuation	If you want to break up a MATLAB	In R, you can spread commands out
		command over more than one line,	over multiple lines, and nothing ex-
		end all but the last line with three	tra is necessary. R will continue read-
		periods: "". E.g.:	ing input until the command is com-
		$x = 3 + \dots$	plete. However, this only works when
		4	the syntax makes it clear that the first
		or	line was not complete. E.g.:
		x = 3	x = 3 +
		+ 4	4
		• •	works, but
			x = 3
			+ 4
			does not treat the second line as a con-
			tinuation of the first.
301	Controlling formatting of	format short g and	options(digits=6) tells R you'd like
001	output	format long g are handy; see	to use 6 digits of precision in values it
	o dop do	help format	displays (it is only a suggestion, not
		р 101	strictly followed)
302	Exit the program	quit or exit	q() or quit()
303	Comments	% this is a comment	# this is a comment
304	Display a string	disp('hi there') or to	print('hi there') Note: to
		omit trailing newline use	avoid having double-quotes
		<pre>fprintf('hi there')</pre>	around the displayed string, do
		•	print('hi there', quote=FALSE)
			or print(noquote('hi there')).
305	Display a string containing	disp('It''s nice') or	print('It\'s nice') or
	single quotes	to omit trailing newline	print("It's nice")
		<pre>fprintf('It''s nice')</pre>	
306	Give prompt and read numer-	<pre>x = input('Enter data:')</pre>	<pre>print('Enter data:'); x=scan()</pre>
	ical input from user		But note: if in a script and you use
			the Edit \rightarrow Execute menu item to
			run it, the selected text after the
			scan statement will be used as source
			for the input, rather than keyboard.
307	Give prompt and read char-	x = input('Enter string:','s')	x = readline('Enter string:')
	acter (string) input from user		
308	Concatenate strings	['two hal' 'ves']	<pre>paste('two hal', 'ves', sep='')</pre>
309	Concatenate strings stored in	<pre>v={'two ', 'halves'};</pre>	v=c('two ', 'halves');
	a vector	strcat(v{:}) But note that	paste(v, collapse='')
		this drops trailing spaces on	
		strings. To avoid that, instead do	
010		strcat([v{:}])	
310	Extract substring of a string	text1='hi there';	text1='hi there';
911	Determine whether -1	text2=text(2:6)	text2=substr(text1,2,6)
311	Determine whether elements	$x = \{'a', 'aa', 'bc', 'c'\}; y$	x = c('a', 'aa', 'bc', 'c'); y
	of a vector are in a set, and	= {'da', 'a', 'bc', 'a', 'bc',	= c('da', 'a', 'bc', 'a', 'bc',
	give positions of correspond-	'aa'}; [tf, loc]=ismember(x,y)	'aa'); loc=match(x,y) Then loc
	ing elements in the set.	Then loc contains the locations of	contains the locations of first oc-
		last occurrences of elements of x	curences of elements of x in the set
1			
		in the set \mathbf{y} , and 0 for unmatched elements.	y, and NA for unmatched elements.

No.	Description	Matlab	R
312	Find indices of regular expression pattern \mathbf{p} in string \mathbf{s}	v=regexp(s,p)	v=gregexpr(p,s)[[1]] (The returned vector also has a "match.length" attribute giving lengths of the matches; this attribute can be removed via attributes(v)=NULL.)
313	Perform some commands only if the regular expression p is contained in the string s	<pre>if (regexp(s,p) commands end</pre>	<pre>if (grepl(p,s)) { commands }</pre>
314	Convert number to string	num2str(x)	as.character(x)
315	Use sprintf to create a formatted string. Use %d for integers ("d" stands for "decimal", i.e. base 10), %f for floating-point numbers, %e for scientific-notation floating point, %g to automatically choose %e or %f based on the value. You can specify field-widths/precisions, e.g. %5d for integers with padding to 5 spaces, or %.7f for floating-point with 7 digits of precision. There are many other options too; see the docs.	<pre>x=2; y=3.5; s=sprintf('x is %d, y=%g', x, y)</pre>	<pre>x=2; y=3.5 s=sprintf('x is %d, y is %g', x, y)</pre>
316	Machine epsilon ϵ_{mach} , i.e. difference between 1 and the next largest double-precision floating-point number	eps (See help eps for various other things eps can give.)	.Machine\$double.eps
317	Pause for x seconds	pause(x)	Sys.sleep(x)
318	Wait for user to press any key	pause	Don't know of a way to do this in R, but scan(quiet=TRUE) will wait until the user presses the Enter key
319	Produce a beep (or possibly a visual signal, depending on preferences set)	beep	alarm()
320	Measure CPU time used to do some commands	<pre>t1=cputime;commands ; cputime-t1</pre>	<pre>t1=proc.time();commands ; (proc.time()-t1)[1]</pre>
321	Measure elapsed ("wall-clock") time used to do some commands	<pre>tic;commands; toc or t1=clock;commands; etime(clock,t1)</pre>	<pre>t1=proc.time();commands ; (proc.time()-t1)[3]</pre>
322	Print an error message an interrupt execution	error('Problem!')	stop('Problem!')
323	Print a warning message	warning('Smaller problem!')	<pre>warning('Smaller problem!')</pre>
324	Putting multiple statements on one line	Separate statements by commas or semicolons. A semicolon at the end of a statement suppresses display of the results (also useful even with just a single statement on a line), while a comma does not.	Separate statements by semicolons.

No.	Description	Matlab	R
325	Evaluate contents of a string s as command(s).	eval(s)	eval(parse(text=s))
326	Get a command prompt for debugging, while executing a script or function. While at that prompt, you can type expressions to see the values of variables, etc.	Insert the command keyboard in your file. Note that your prompt will change to K>>. When you are done debugging and want to continue executing the file, type return.	Insert the command browser() in your file. Note that your prompt will change to Browse [1]>. When you are done debugging and want to continue executing the file, either type c or just press return (i.e. enter a blank line). Note, if you type n, you enter the step debugger.
327	Show where a command is	which sqrt shows you where the file defining the sqrt function is (but note that many basic functions are "built in," so the MATLAB function file is really just a stub containing documentation). This is useful if a command is doing something strange, e.g. sqrt isn't working. If you've accidentally defined a variable called sqrt, then which sqrt will tell you, so you can clear sqrt to erase it so that you can go back to using the function sqrt.	R does not execute commands directly from files, so there is no equivalent command.
328	Query/set the search path.	path displays the current search path (the list of places MATLAB searches for commands you enter). To add a directory ~/foo to the beginning of the search path, do addpath ~/foo -begin or to add it to the end of the path, do addpath ~/foo -end (Note: you should generally add the full path of a directory, i.e. in Linux or Mac OS-X something like ~/foo as above or of the form /usr/local/lib/foo, while under Windows it would be something like C:/foo)	R does not use a search path to look for files.
329	Startup sequence	If a file startup.m exists in the startup directory for MATLAB, its contents are executed. (See the MATLAB docs for how to change the startup directory.)	If a file .Rprofile exists in the current directory or the user's home directory (in that order), its contents are sourced; saved data from the file .RData (if it exists) are then loaded. If a function .First() has been defined, it is then called (so the obvious place to define this function is in your .Rprofile file).
330	Shutdown sequence	Upon typing quit or exit, MATLAB will run the script finish.m if present somewhere in the search path.	Upon typing q() or quit(), R will call the function .Last() if it has been de- fined (one obvious place to define it would be in the .Rprofile file)

No.	Description	Matlab	R
331	Install and load a package.	Matlab does not have packages. It	To install e.g. the deSolve pack-
		has toolboxes, which you can pur-	age, you can use the command
		chase and install. "Contributed"	install.packages('deSolve').
		code (written by end users) can sim-	You then need to load the package
		ply be downloaded and put in a di-	in order to use it, via the command
		rectory which you then add to MAT-	library('deSolve'). When running
		LAB's path (see item 328 for how to	R again later you'll need to load the
		add things to Matlab's path).	package again to use it, but you
			should not need to re-install it. Note
			that the lattice package is typically
			included with binary distributions of
			R, so it only needs to be loaded, not
			installed.

10 Spatial Modeling

No.	Description	Matlab	R
332	Take an $L \times L$ matrix A of	A = (A (rand(L) < p))*1;	A = (A (matrix(runif(L^2),L)
	0s and 1s, and "seed" frac-)*1
	tion p of the 0s (turn them		
	into 1s), not changing entries		
	which are already 1.		
333	Take an $L \times L$ matrix A of 0s	A = (A & (rand(L) < 1-p))*1;	$A = (A \& (matrix(runif(L^2),L))$
	and 1s, and "kill" fraction p		< 1-p))*1
	of the 1s (turn them into 0s),		
	not changing the rest of the		
	entries		
334	Do "wraparound" on a coor-	mod(newx-1,L)+1 Note: for porta-	((newx-1) %% L) + 1 Note: for
	dinate newx that you've al-	bility with other languages such as	portability with other languages such
	ready calculated. You can	C which handle MOD of negative	as C which handle MOD of nega-
	replace \mathbf{newx} with $\mathbf{x} + \mathbf{dx}$ if	values differently, you may want to	tive values differently, you may want
	you want to do wraparound	get in the habit of instead doing	to get in the habit of instead doing
	on an offset x coordinate.	mod(newx-1+L,L)+1	((newx-1+L)%%L) + 1
335	Randomly initialize a portion	dx=ix2-ix1+1; dy=iy2-iy1+1;	dx=ix2-ix1+1; dy=iy2-iy1+1;
	of an array: set fraction p of	$A(iy1:iy2,ix1:ix2) = \dots$	A[iy1:iy2,ix1:ix2] =
	sites in rows iy1 through iy2	(rand(dy,dx) < p0)*1;	(matrix(runif(dy*dx),dy) <
	and columns ix1 through ix2		p0)*1
	equal to 1 (and set the rest of		
	the sites in that block equal		
	to zero). Note: this assume		
	iy1 < iy2 and $ix1 < ix2$.		

Index of MATLAB commands and concepts

, 83 , , 324 , , 82 , , 300 , , 90 , , 94 , 89 , , 12-14 , 89 , 12-14 , 132 , 14-14 , 15-14 , 15-14 , 15-14 , 15-14 , 15-14 , 15-14 , 15-15 ,
.*, 82, 300, 300, 300, 94, 266, 94, 89, 12-14, 200, 214, 224, 228, 230, 248, 2
, 300 ./, 90 ./, 90 ./, 90 ./, 94 ./, 89 .:, 12-14 .:, 324 .:, 324 .:, 324 .:, 286 .:, 6-8 ./, 303 ./, 84, 91 ./, 91 ./, 84, 91 ./, 89 .:, 54, 92, 93 ./, 84, 91 ./, 84, 91 ./, 85, 74 ./, 80 ./, 8
./, 90 ./, 94 ./, 89 .; 12-14 .; 324 .; 324 .; 324 .; 324 .; 325 .; 324 .; 324 .; 324 .; 326 .; 327 .; 328 .; 328 .; 329 .; 329 .; 329 .; 321 .; 324 .; 324 .; 321 .; 324 .; 321 .; 322 .; 324 .; 322 .; 322 .; 323 .; 324 .; 324 .; 322 .; 322 .; 323 .; 324 .; 322 .; 323 .; 324 .; 324 .; 324 .; 326 .; 327 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 328 .; 324 .; 327 .; 328
.↑, 94 /, 89
1, 89
$\begin{array}{c} :, 12-14 \\ :, 324 \\ :, 324 \\ :, 286 \\ \vdots \\ \text{Colormap} \\ \text{L}, 6-8 \\ \text{Many Surface} \\ \text{Many Surface} \\ \text{L}, 186, 187 \\ \text{Colormap, 271, 272} \\ \text{Colomap, 271, 272} \\ \text{Colormap, 271, 272} \\ \text{Colormap, 271, 272} \\$
; 324 colorbar, 270 colormap $= 2.86$ colormap $= 2.86$ colormap $= 2.86$ building your own, 272 $= 2.86$ building your own, 272 $= 2.86$ colormap, 271, 272 column vector, 7 $= 2.86$ complex numbers, 73–78 $= 2.86$ complex numbers, 73–78 $= 2.86$ cond, 104–106 $= 2.86$ cond, 104–106 $= 2.86$ cond, 104–106 $= 2.86$ cond, 104–106 $= 2.86$ continue, 191 $= 2.86$ continue, 191 $= 2.86$ continue, 191 $= 2.86$ continue, 254 $= 2.86$ conv. 165 $= 2.86$ conv. 165 $= 2.86$ conv. 165 $= 2.86$ conv. 165 $= 2.86$ conv. 116, 117 $= 2.86$ cosh, 61 $= 2.86$ cosh, 62 cosh, 61 $= 2.86$ cosh, 61 $=$
E, 286
[, 6-8
%, 303 colormap, 271, 272 &, 186, 187 column vector, 7 7, 54, 92, 93 comments, 303 84, 91 complex numbers, 73–78 { 49 cond, 104–106 abs, 55, 74 continue, 191 acosh, 62 continue, 191 adopath, 328 corr, 118–123 al1, 188 cos, 59 angle, 75 cosh, 61 annotation, 263, 264 cov, 116, 117 ans, 299 cputime, 320 arrows in plots, 263, 264 cross, 80 arrows in plots, 263, 264 csape, 178, 180, 181 asin, 60 csape, 178, 180, 181 asin, 62 natural, 178 assin, 62 natural, 178 assignin, 287 natural, 182 atan, 60 cumulative distribution functions bar, 244, 246, 247 binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 binopdf, 222 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponentia
&, 186, 187 column vector, 7 $^\circ$, 54, 92, 93 comments, 303 $^\circ$, 84, 91 complex numbers, 73–78 $^\circ$, 49 cond, 104–106 conj, 76 continue, 191 acos, 60 continue, 254 acosh, 62 conv, 165 addpath, 328 corr, 118–123 all, 188 cos, 59 angle, 75 cosh, 61 annotation, 263, 264 cov, 116, 117 ans, 299 cputime, 320 arrows in plots, 263, 264 csape, 178, 180, 181 asin, 60 cape, 178, 180, 181 asinh, 62 natural, 178 assignin, 287 not-a-knot, 182 atan, 60 periodic, 181 atan, 60 periodic, 181 atan, 62 cumprod, 135 average, see mean cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 binordf, 229 poisson, 230 binordal, 232 Poisson, 230 break, 191 debugging, 326 det, 86 det, 86
&, 186, 187 column vector, 7 $^\circ$, 54, 92, 93 comments, 303 $^\circ$, 84, 91 complex numbers, 73–78 $^\circ$, 49 cond, 104–106 conj, 76 continue, 191 acos, 60 continue, 254 acosh, 62 conv, 165 addpath, 328 corr, 118–123 all, 188 cos, 59 angle, 75 cosh, 61 annotation, 263, 264 cov, 116, 117 ans, 299 cputime, 320 arrows in plots, 263, 264 csape, 178, 180, 181 asin, 60 cape, 178, 180, 181 asinh, 62 natural, 178 assignin, 287 not-a-knot, 182 atan, 60 periodic, 181 atan, 60 periodic, 181 atan, 62 cumprod, 135 average, see mean cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 binordf, 229 poisson, 230 binordal, 232 Poisson, 230 break, 191 debugging, 326 det, 86 det, 86
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$
$ \begin{array}{llllllllllllllllllllllllllllllllllll$
$ \begin{tabular}{lllllllllllllllllllllllllllllllllll$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{llllllllllllllllllllllllllllllllllll$
$\begin{array}{llllllllllllllllllllllllllllllllllll$
$\begin{array}{llllllllllllllllllllllllllllllllllll$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{llllllllllllllllllllllllllllllllllll$
annotation, 263, 264 ans, 299 any, 189 arrows in plots, 263, 264 asin, 60 asinh, 62 assignin, 287 atan, 60 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cov, 116, 117 cputime, 320 cov, 18, 180, 181 cubic splines, 179, 180 natural, 178 not-a-knot, 182 periodic, 181 cumprod, 135 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
ans, 299 any, 189 arrows in plots, 263, 264 asin, 60 asinh, 62 asinjn, 287 atan, 60 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 casin, 60 caspe, 178, 180, 181 cubic splines, 179, 180 natural, 178 not-a-knot, 182 periodic, 181 cumprod, 135 cumprod, 135 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
any, 189 arrows in plots, 263, 264 asin, 60 asinh, 62 assignin, 287 atan, 60 atanh, 62 average, see mean axis, 255 axis, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 coross, 80 cross, 80 cross, 80 csape, 178, 180, 181 cubic splines, 179, 180 natural, 178 not-a-knot, 182 periodic, 181 cumprod, 135 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 poisson, 230 det, 86 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
arrows in plots, 263, 264 asin, 60 asinh, 62 assignin, 287 atan, 60 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 casinh, 62 caspe, 178, 180, 181 cubic splines, 179, 180 natural, 178 not-a-knot, 182 periodic, 181 cumprod, 135 cumsum, 131–134 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 poisson, 230 debugging, 326 det, 86 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
asin, 60 asinh, 62 assignin, 287 atan, 60 atanh, 62 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cubic splines, 179, 180 antural, 178 not-a-knot, 182 periodic, 181 cumprod, 135 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
asinh, 62 assignin, 287 atan, 60 atanh, 62 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cassignin, 287 anot-a-knot, 182 periodic, 181 cumprod, 135 cumsum, 131–134 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 break, 191 diag, 22, 23 diff, 137 differential equations, see ode45
assignin, 287 atan, 60 atanh, 62 average, see mean axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 catanh, 62 atanh, 62 cumprod, 135 cumsum, 131–134 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
atan, 60 atanh, 62 atanh, 62 average, see mean axis, 255 cumsum, 131–134 cumulative distribution functions bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 ceil, 66 periodic, 181 cumprod, 135 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a, b), 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
atanh, 62 cumprod, 135 cumsum, $131-134$ axis, 255 cumulative distribution functions binomial, 229 continuous uniform on interval (a,b) , 233 discrete uniform from $1n$, 234 exponential, 231 normal, 232 binornd, 213 boolean tests scalar, 186 vector, $187-189$ det, 86 break, 191 differential equations, see ode45
average, see mean axis, 255 cumsum, 131–134 cumsum, 131–134 cumulative distribution functions binomial, 229 continuous uniform on interval (a,b) , 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 binornd, 213 boolean tests scalar, 186 vector, 187–189 det, 86 test, 191 diag, 22, 23 cd, 280 ceil 66 diag, 22, 23 diff, 137 differential equations, see ode45
axis, 255 bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 cumulative distribution functions binomial, 229 continuous uniform on interval (a,b) , 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
bar, 244, 246, 247 beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 ceil, 66 binomial, 229 continuous uniform on interval (a, b) , 233 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
beep, 319 binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, $187-189$ break, 191 continuous uniform on interval (a,b) , 233 discrete uniform from $1n$, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 det, 86 break, 191 diag, 22, 23 diff, 137 differential equations, see ode45
binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 ceil 66 discrete uniform from 1n, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
binocdf, 229 binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 ceil 66 disserte timform from 1t, 234 exponential, 231 normal, 232 Poisson, 230 debugging, 326 det, 86 det, 86 diff, 137 differential equations, see ode45
binopdf, 222 binornd, 213 boolean tests scalar, 186 vector, 187–189 break, 191 cd, 280 ceil 66 binopdf, 222 normal, 232 Poisson, 230 debugging, 326 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
binornd, 213 boolean tests scalar, 186 vector, 187–189 det, 86 break, 191 diag, 22, 23 cd, 280 ceil 66 diag, 22, 23 diff, 137 differential equations, see ode45
boolean tests scalar, 186 vector, 187-189 break, 191 diag, 22, 23 cd, 280 ceil 66 debugging, 326 det, 86 diff, 137 differential equations, see ode45
vector, 187-189 break, 191 cd, 280 ceil 66 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
vector, 187-189 break, 191 cd, 280 ceil 66 det, 86 diag, 22, 23 diff, 137 differential equations, see ode45
break, 191 diag, 22, 23 cd, 280 diff, 137 ceil 66 differential equations, see ode45
cd, 280 diff, 137 differential equations, see ode45
ceil 66 differential equations, see ode45
A = 101
dir, 281 cell, 48
cell arrays, 48 disp, 304, 305
extracting elements of, 49 doc, 4 dot, 79
cellular automata animation, 278

$\mathtt{drawnow},274,278$	user-written, 193 returning multiple values, 194
echelon form, see matrix	fzero, 167
eig, 96	,
element-by-element matrix operations, see ma-	gca, 259
trix	gcf, 237
else, 185	get, 238
elseif, 185	Greek letters
end, 40	in plot labels, 258
eps, 316	grid,260
erf, 68	
erfc, 69	$\mathtt{help},\ 1 ext{}3$
erfcinv, 71	${\tt helpbrowser},4$
erfinv, 70	${\tt helpdesk},4$
error, 322	hilb, 46
errorbar, 248, 249	hist, 163, 164, 245, 246
etime, 321	hold, 266
eval, 325	
evalin, 288	identity, see matrix
exit, 302, 330	if, 184-186
exp, 56	imag, 78
expcdf, 231	image, 269, 278
expm, 130	${\tt imagesc},269$
exppdf, 224	importdata, 283, 284
exprnd, 215	$\mathtt{ind2sub},36$
eye, 21	indexing
	matrix, 10
figure, 235, 236	with a single index, 11
file	vector, 9
running commands in, 282	input, 306, 307
text	inv, 87
reading data from, 283, 284	inverse, see matrix
saving data to, 285	ismember, 311
find, $160-162$	
finish.m, 330	keyboard, 326
fliplr, 34	1 207
flipud, 35	legend, 265
floor, 65	length, 150, 152
fminbnd, 168, 169	linspace, 15
fminsearch, 170, 171	load, 283
font size in plots, 259	log, 57
for, 183	log10, 58
format, 301	log2, 58
fplot, 268	loglog, 243
fprintf, 304, 305	logspace, 16
function	lookfor, 5
multi-variable	lu, 97
minimization, 170	
minimization over first parameter only, 169	matrix, 8
minimization over only some parameters,	boolean operations on, 161, 162
171	changing shape of, 43
single-variable	Cholesky factorization, 100
minimization, 168	circular shift, 33
111111111111111111111111111111111111111	condition number, 104–106

containing all zeros, 19 converting row, column to single index, 37 converting single-index to row, column, 36 cumulative sums of columns, 132 min, 139–142, 144–144.6 mind, 143 midd; 137 middir, 179 mpdf, 228 mprnd, 220, 221 mod, 63, 334 modulo arithmetic, 63, 334 multiplic statements on one line, 324 columns, 102, 102 normodf, 232 normodf, 23	containing all indentical entries, 20	$\mathtt{mean},\ 107109$
converting single-index to row, column, 36 cumulative sums of all elements of, 134 cumulative sums of columns, 132 cumulative sums of rows, 133 determinant, 86 diagonal, 22 cehelon form, 85 cigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a rectangular piece of, 31 extracting a pecified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147-149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose, 83 transpose, 88 transpose, 88 transpose, 88 transpose, 83 transpose, 85 transp		mesh, 254
converting single-index to row, column, 36 cumulative sums of all elements of, 134 cumulative sums of columns, 132 cumulative sums of rows, 133 determinant, 86 diagonal, 22 echelon form, 85 eigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a rotumn of, 28 extracting a rectangular piece of, 31 extracting a preclamgular post of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 reverse elements in ros, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147-149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 89 tra	converting row, column to single index, 37	meshgrid, 26, 118, 253
cumulative sums of all elements of, 134 cumulative sums of columns, 132 cumulative sums of columns, 132 cumulative sums of rows, 133 determinant, 86 diagonal, 22 echelon form, 85 eigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a row of, 29 extracting a row of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 size of, 147-149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose,		min, 139-142, 144-146
cumulative sums of rows, 133 determinant, 86 diagonal, 22 echelon form, 85 eigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a row of, 29 extracting a row of, 29 extracting a row of, 29 extracting a prow of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose, 84 transpose, 84 transpose, 84 transpose, 84 transpose, 8		mind, 143
cumulative sums of rows, 133 determinant, 86 diagonal, 22 echelon form, 85 cigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a row of, 29 extracting a row of, 29 extracting a row of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose, 84 transpose, 84 transpose, 84 transpose, 84 transpose, 84 transpose, 84 transpose, 85 transp	cumulative sums of columns, 132	mkdir, 279
diagonal, 22 echelon form, 85 cigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a rectangular piece of, 31 entracting a row of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 endifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 element-by-element, 82 element-by-element, 82 element-by-element, 82 of rows of, 129 elements, 127 of columns of, 128 of rows of, 129 elements in columns, 35 reverse elements in columns, 35 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147-149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose, 84 transpose, 84 transpose, 85 transpose, 85 transpose, 85 transpose, 85 transpose, 85 transpose, 85		mnpdf, 228
diagonal, 22 echelon form, 85 eigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a roctangular piece of, 31 extracting a row of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 transpose,		mnrnd, 220, 221
echelon form, 85 eigenvalues and eigenvectors of, 96 equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a rectangular piece of, 31 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 trans		mod, 63, 334
equation solving, 84 exponential of, 130 extracting a column of, 28 extracting a row of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 t	echelon form, 85	modulo arithmetic, 63, 334
solving, 84 exponential of, 130 extracting a column of, 28 extracting a rectangular piece of, 31 extracting a row of, 29 extracting a rectangular piece of, 31 extracting a row of, 29 extracting a rectangular piece of, 31 extracting a row of, 29 extracting a rectangular piece of, 31 extracting a row of, 29 normed, 225 normrid, 229 ode45, 195–197 ones, 18, 20 openvar, 293 optimization, 168–171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 permutation of integers 1n, 208 plot, 240–242, 267 Greek letters in axis labels, 258 plot3, 251 poisscuff, 230 poissrd, 231 poissrd, 232 poissrd, 24 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 pval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomal, 228 normal, 225 poissrd, 230 poissrd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 pval, 178, 180, 181 print, 275–2		multiple statements on one line, 324
exponential of, 130 extracting a column of, 28 extracting a rectangular piece of, 31 extracting a rectangular piece of, 31 extracting a pown of, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83	equation	
extracting a column of, 28 extracting a rectangular piece of, 31 extracting a reword, 29 extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 tr	solving, 84	
extracting a rectangular piece of, 31 extracting a rectangular piece of, 31 extracting a rectangular piece of, 31 extracting a row of, 29 normrid, 219 num2str, 314 nume1, 151 ode45, 195-197 ones, 18, 20 openvar, 293 optimization, 168-171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 permutation of integers 1n, 208 plots, 240-242, 267 Greek letters in axis labels, 258 plot3, 251 poissend, 223 poissend, 224 polar, 250 polyfit, 174-176 polynomial least-squares fitted, 175-177 multiplication, 165 roots of, 166 pval, 178, 180, 181 print, 275-277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	exponential of, 130	
extracting a row of, 29 extracting a pecified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 reverse elements in columns, 35 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 of rows of, 126 trace, 88 transpose, 83 tr	extracting a column of, 28	
extracting specified rows and columns of, 32 "gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147-149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper triangular portion of, 45 numel, 151 ode45, 195-197 ones, 18, 20 openvar, 293 optimization, 168-171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 eremutation of integers 1n, 208 plot, 240-242, 267 Greek letters in axis labels, 258 plota, 251 poissord, 230 poisspdf, 233 poissrnd, 214 polar, 250 polynimization, 168-171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 Greek letters in axis labels, 258 plota, 251 poissord, 230 poisspdf, 233 poissrnd, 214 polar, 250 polynomial least-squares fitted, 175-177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275-277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	extracting a rectangular piece of, 31	
"gluing" together, 24, 25 identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each cloumn of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper triangular portion of, 44 ode 45, 195–197 ones, 18, 20 openvar, 293 optimization, 168–171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 permutation of integers 1n, 208 plot, 240–242, 267 Greek letters in axis labels, 258 plot3, 251 poisscaft, 230 poisspdf, 223 poisspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	extracting a row of, 29	
identity, 21 inverse, 87 lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements in to a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper triangular portion of 45.	extracting specified rows and columns of, 32	num2str, 314
identity, 21 inverse, 87 inverse, 87 lower-triangular portion of, 44 UU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 VV		numel, 151
lower-triangular portion of, 44 LU factorization, 97 minimum of values of, 140 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 44 uninimum of values of, 140 penvar, 293 optimization, 168–171 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 permutation of integers 1n, 208 plot, 240–242, 267 Greek letters in axis labels, 258 plot3, 251 poisscuff, 230 poisspdf, 223 poisspnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 portor, 254, 269, 278 perform some commands with probability p, 207 Greek letters in axis labels, 258 plot3, 251 poisscuff, 230 poisspnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 225 poisson, 223		
minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements in to a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45 volume of each column of, 141 minimum value of each column of, 142 path, 328 pause, 317, 318 pcolor, 254, 269, 278 Greek letters in axis labels, 258 plott3, 251 poisscuft, 230 poisspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	inverse, 87	
minimum of values of, 140 minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns of, 128 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 permutation of integers 1n, 208 plott, 240–242, 267 Greek letters in axis labels, 258 plott3, 251 poisscrdf, 230 poisspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	lower-triangular portion of, 44	
minimum value of each column of, 141 minimum value of each column of, 141 minimum value of each column of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper triangular portion of, 45 minimum value of each column of, 141 minimum value of each row of, 142 path, 328 pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p, 207 permutation of integers 1n, 208 plot, 240–242, 267 Greek letters in axis labels, 258 plot3, 251 poisscrdf, 230 poisspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	LU factorization, 97	- ·
minimum value of each column of, 141 minimum value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 N and N a	minimum of values of, 140	optimization, 168–171
minimul value of each row of, 142 modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 N modifying elements given lists of rows and columns of, 125 of rows of, 126 trace, 88 transpose, 83 N multiplication, 142 N pause, 317, 318 pcolor, 254, 269, 278 perform some commands with probability p , 207 permutation of integers $1n$, 208 plot, 240–242, 267 Greek letters in axis labels, 258 plot3, 251 poisscoff, 230 poisspaff, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 pval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b) , 226 discrete uniform from $1n$, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		
modifying elements given lists of rows and columns, 38 multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 $polor, 254, 269, 278$ perform some commands with probability p , 207 permutation of integers $1n$, 208 plot, 240 – 242 , 267 Greek letters in axis labels, 258 plot3, 251 poisscuff, 230 poisspuff, 223 poissrnd, 214 polar, 250 polyfit, 174 – 176 polynomial least-squares fitted, 175 – 177 multiplication, 165 roots of, 166 ppval, 178 , 180 , 181 print, 275 – 277 probability density functions binomial, 222 continuous uniform on interval (a, b) , 226 discrete uniform from $1n$, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		- -
columns, 38 multiplication, 81 element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 trace, 88 transpose, 83 upper-triangular portion of 45		-
multiplication, 81 element-by-element, 82 N -dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 126 trace, 88 transpose, 83 upperstriagenglar portion of 45		
element-by-element, 82 N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		- *
N-dimensional, 47 norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper triangular portion of 45 Greek letters in axis labels, 258 plot3, 251 poissodf, 230 poisspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		-
norm, 103 powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45	· · · · · · · · · · · · · · · · · · ·	-
powers of, 93 product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 99 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 poissond, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a,b) , 226 discrete uniform from $1n$, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		
product of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45		
of all elements, 127 of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45 polsspdf, 223 poissrnd, 214 polar, 250 polyfit, 174–176 polynomial least-squares fitted, 175–177 multiplication, 165 roots of, 166 ppval, 178, 180, 181 print, 275–277 probability density functions binomial, 222 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	- · ·	
of columns of, 128 of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
of rows of, 129 QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		-
QR factorization, 101 rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
rank, 95 re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
re-shaping its elements into a vector, 30 reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
reverse elements in columns, 35 reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		- · · · · · · · · · · · · · · · · · · ·
reverse elements in rows, 34 Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
Schur decomposition, 99 singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
singular value decomposition, 98 size of, 147–149, 151, 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of, 45		
size of, 147 – 149 , 151 , 152 sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 probability density functions binomial, 222 continuous uniform on interval (a,b) , 226 discrete uniform from $1n$, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		- ·
sum of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45		
of all elements, 124 of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 continuous uniform on interval (a, b), 226 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		
of columns of, 125 of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 discrete uniform from 1n, 227 exponential, 224 multinomial, 228 normal, 225 Poisson, 223		
of rows of, 126 trace, 88 transpose, 83 upper-triangular portion of 45 exponential, 224 multinomial, 228 normal, 225 Poisson, 223	•	
trace, 88 transpose, 83 normal, 225 poisson, 223		
transpose, 83 upper-triangular portion of 45 Poisson, 223		
upper-triangular portion of 45		
nrod 127/ 120		
max, see min		prod, 127-129

qr, 101	splines, see cubic splines
quad, 172	sprintf, 315
quit, 302, 330	sqrt, 53
1,,	stairs, 250
rand, 198-206, 212	standard deviation, see std
random values	startup.m, 329
Bernoulli, 204	std, 110-112
binomial, 213	stem, 250
continuous uniform distribution on interval	stop, 322
(a,b), 201, 218	strcat, 309
continuous uniform distribution on interval	string
(0,1), 198-200	concatenation, 308
discrete uniform distribution from ab , 206	converting number to, 314
discrete uniform distribution from $1k$, 203 ,	pattern matching, 312, 313
216, 217	substrings, 310
discrete uniform distribution, 202	struct, 51
exponential, 215	sub2ind, 37, 38
k unique values sampled from integers 1 n ,	subplot, 273
209	sum, 124–126, 187
multinomial, 220, 221	surf, 252, 253
normal, 219	surfc, 254
Poisson, 214	surfl, 254
setting the seed, 212	svd, 98
randperm, 208, 209	switch, 192
randsample, 209-211	SWITCH, 192
rank, 95	tan, 59
rcond, 104	tanh, 61
real, 77	text, 261, 262
regexp, 312, 313	tic, 321
reshape, 43, 47	title, 256
roots	toc, 321
of general single-variable function, 167	trace, 88
polynomial, 166	transpose, see matrix
roots, 166	trapz, 173
round, 64	tril, 44
row vector, 6	triu, 45
rref, 85	
	unidcdf, 234
sampling values from a vector, 210, 211	$\mathtt{unidpdf},227$
save, 285	unidrnd, 216, 217
schur, 99	$\mathtt{unifcdf}, 233$
semilogx, 243	$\mathtt{unifpdf},226$
semilogy, 243	unifrnd, 218
set, 259	unique, 163 , 246
shading, 269	119 115
sign, 67	var, 113–115
\sin , 59	variables
sinh, 61	assigning, 286
${\tt size},147149$	assigning in base environment from func-
slice, 254	tion, 287
sort, 153, 154, 209	evaluating from base environment within func
sortrows, 155-158	tion, 288
spline, 182	names, 298
	variance, see var

vector

boolean operations on, 159, 160 containing all indentical entries, 18 containing all zeros, 17 counts of binned values in, 164 counts of discrete values in, 163 cross product, 80 cumulative sum of elements of, 131 differences between consecutive elements of, 137 dot product, 79 minimum of values of, 139 norm, 102 position of first occurance of minimum value in, 146 product of all elements, 127 reversing order of elements in, 27 size of, 150 sum of all elements, 124 truncating, 39

 $\begin{array}{l} \mathtt{warning},\,323\\ \mathtt{waterfall},\,254\\ \mathtt{which},\,327\\ \mathtt{while},\,190\\ \mathtt{who},\,289\\ \mathtt{whos},\,290-292\\ \end{array}$

 $\mathtt{xlabel},\ 257\text{--}259$

ylabel, 257, 258

zeros, 17, 19

Index of ${\sf R}$ commands and concepts

*, 92	colormap
/, 90	building your own, 272
;, 12, 13	
;, 324	for image, 271
	colSums, 125
<-, 286	column vector, 7
<<-, 287	comments, 303
=, 286	complex numbers, 73–78
$\frac{1}{2}$	Conj, 76
[[, 49	contour, 254
#, 303	convolve, 165
%%, 63, 334	cor, 119-123
&, 186, 187	$\cos, 59$
^, 54, 94	cosh, 61
aba 55 74	cov, 116–118
abs, 55, 74	cubic splines, 179, 180, 182
acos, 60	natural, 178
acosh, 62	periodic, 181
alarm, 319	$\mathtt{cummax}, 136$
all, 188	cummin, 136
any, 189	cumprod, 135
apply, 34, 35, 112, 114, 115, 128, 141, 142	$\mathtt{cumsum},\ 131-134$
Arg, 75	cumulative distribution functions
array, 47	binomial, 229
arrows, 263, 264	continuous uniform on interval (a, b) , 233
as.character, 314	discrete uniform from $1n$, 234
as.formula, 176	exponential, 231
as.numeric, 163	normal, 232
asin, 60	Poisson, 230
$\operatorname{asinh}, 62$	curve, 268
atan, 60	
$\mathtt{atanh},62$	$\mathtt{data.frame},51$
average, see mean	${\tt dbinom},222$
harmlat 244 246	debugging, 326
barplot, 244, 246	$\mathtt{det},86$
boolean tests	dev.control, 275, 276, 278
scalar, 186	${\tt dev.copy},275,276$
vector, 187–189	dev.copy2eps, 276
break, 191	dev.copy2pdf, 275
browser, 326	$\operatorname{dev.cur}(), 237$
c, 6, 7	dev.list, 238
cbind, 24, 38	$\mathtt{dev.new},235$
ceiling, 66	${\tt dev.off},239,275 – 277$
cellular automata animation, 278	$\mathtt{dev.set}, 236$
chol, 100	dexp, 224
choose, 72	$\mathtt{diag},2123$
class, 297	diff, 137
cloud, 251	differential equations, see lsoda
	dim, 43, 149, 152
coef, 174-177	dir, 281
colMeans, 108	dir.create, 279
colon, see:	,

dmultinom, 228	$\mathtt{hist}, 164, 244, 245, 247$
dnorm, 225	• 1 • • • • • • • • • • • • • • • • • •
dpois, 223	identity, see matrix
dunif, 226	if, 184-186
	ifelse, 138
echelon form, see matrix	Im, 78
eig, 96	$\mathtt{image},269,278$
element-by-element matrix operations, see ma-	indexing
trix	matrix, 10
else, 185	with a single index, 11
errbar, 248, 249	vector, 9
eval, 325	${\tt install.packages},331$
exp, 56	integrate, 172
expand, 97	inverse, see matrix
expand.grid, 254	
expm, 130	jpeg, 277
file	kappa, 105
running commands in, 282	•• /
text	$.\mathtt{Last},330$
reading data from, 283, 284	.Last.value, 299
saving data to, 285	lattice package, 254, 270, 274, 331
	layout, 273
filled.contour, 270	legend, 265
.First, 329	length, 39, 40, 150, 151
fix, 293	levelplot, 270, 274
floor, 65	library, 3, 331
font size in plots, 259	lines, 266
for, 183	lists, 48
function	extracting elements of, 49
multi-variable	1m, 174–177
minimization, 170	
minimization over first parameter only, 169	lm.fit, 176
minimization over only some parameters,	log, 57
171	log10, 58
single-variable	log2, 58
minimization, 168	lower.tri, 45
user-written, 193	ls, 289
returning multiple values, 194	ls.str, 290, 292 lsoda, 195-197
get, 288	,
globalenv, 288	.Machine $\$$ double.eps, 316
graphics	$\mathtt{match},311$
not being displayed from scripts/functions,	$\mathtt{matplot},267$
274	matrix, 8
Greek letters	boolean operations on, 161, 162
in plot labels, 258	changing shape of, 43
gregexpr, 312	Cholesky factorization, 100
grepl, 313	circular shift, 33
grid, 260	condition number, 104–106
5-14, 200	containing all indentical entries, 20
help, 1, 2	containing all zeros, 19
help.search, 5	converting row, column to single index, 3'
help.start, 4	converting single-index to row, column, 30
Hilbert. 46	cumulative sums of all elements of, 134

cumulative sums of columns, 132	multiple statements on one line, 324
cumulative sums of rows, 133	names, $50, 163$
determinant, 86	ncol, 148
diagonal, 22	next, 191
echelon form, 85	
eigenvalues and eigenvectors of, 96	norm, 102, 103
equation	nrow, 147
solving, 84	optim, 170, 171
exponential of, 130	optimization, 168–171
extracting a column of, 28	optimize, 168, 169
extracting a rectangular piece of, 31	- · · · · · · · · · · · · · · · · · · ·
extracting a row of, 29	options
extracting specified rows and columns of, 32	digits=, 301
"gluing" together, 24, 25	order, 155-158
identity, 21	outer, $176, 253$
inverse, 87	packages
lower-triangular portion of, 44	installing, 331
LU factorization, 97	loading, 331
minimum of values of, 140	9.
minimum value of each column of, 141	par, 259
minimum value of each row of, 142	par
modifying elements given lists of rows and	mfcol=, 273
columns, 38	mfrow=, 273
multiplication, 81	parse, 325
element-by-element, 82	paste, 176, 308, 309
N-dimensional, 47	pbinom, 229
norm, 103	pdf, 259, 275
powers of, 93	perform some commands with probability $p, 207$
product	permutation of integers $1n$, 208
of all elements, 127	persp, 252, 253
of columns of, 128	pexp, 231
of rows of, 129	$\mathtt{pie},250$
QR factorization, 101	$\mathtt{plot},240243$
rank, 95	Greek letters in axis labels, 258
re-shaping its elements into a vector, 30	$\mathtt{main=},256$
	$\mathtt{sub}\mathtt{=},256$
reverse elements in columns, 35	xlab=, 257, 258
reverse elements in rows, 34	$\mathtt{xlim}=,255$
Schur decomposition, 99	ylab=, 257, 258
singular value decomposition, 98	ylim=, 255
size of, 147–149, 151, 152	pmin, 143, 144
sum	pnorm, 68, 69, 232
of all elements, 124	points, 266
of columns of, 125	polynomial
of rows of, 126	least-squares fitted, 175–177
trace, 88	multiplication, 165
transpose, 83	roots of, 166
upper-triangular portion of, 45	polyreg, 176
matrix, 8, 19, 20	polyroot, 166
$\max, see \min$	postscript, 276
$\mathtt{mean},107$	ppois, 230
$\min, 139-142, 145$	print, 274, 304, 305
$\operatorname{Mod} olimits, 74$	probability density functions
modulo arithmetic, 63, 334	binomial, 222
	· · · · · · · · · · · · · · · · · · ·

continuous uniform on interval (a, b) , 226	polynomial, 166
discrete uniform from $1n$, 227	round, 64
exponential, 224	row vector, 6
multinomial, 228	rowMeans, 109
normal, 225	rpois, 214
Poisson, 223	.Rprofile, 329
proc.time, 320, 321	$\mathtt{runif}, 198204, 206, 218$
prod, 127-129	sample, 208-211, 216, 217
punif, 233	
- 202 220	sampling values from a vector, 210, 211
q, 302, 330	scan, 306, 318
qnorm, 70, 71	Schur, 99
qr, 95, 101	sd, 110–112
quartz, 235	seq, 14-16
quit, 302, 330	set.seed, 212
rand 205	setwd, 280
rand, 205 random values	sign, 67
Bernoulli, 204	sin, 59
	sinh, 61
binomial, 213	solve, 84, 87, 89, 91
continuous uniform distribution on interval	sort, 153, 154
(a,b), 201, 218	source, 282
continuous uniform distribution on interval	spline, 178, 179, 181
(0,1), 198, 200	splines, see cubic splines
continuous uniform distribution on inteval	split.screen, 273
(0,1), 199	sprintf, 315
discrete uniform distribution from a.b, 206	sqrt, 53
discrete uniform distribution from $1k$, 203 ,	standard deviation, see sd
216, 217	str, 291
discrete uniform distribution, 202	string
exponential, 215	concatenation, 308
k unique values sampled from integers $1n$,	converting number to, 314
209	pattern matching, 312, 313
multinomial, 220, 221	substrings, 310
normal, 219	$\mathtt{substr},310$
Poisson, 214	sum, 124, 126, 187
setting the seed, 212	svd, 98
rbind, 25	$\mathtt{switch},192$
rbinom, 213	${\tt symbols},254$
rcond, 104, 106	Sys.sleep, 317
.RData, 329	
Re, 77	t, 83
$\mathtt{read.table},283,284$	table, $163, 246$
${\tt readline},307$	$\mathtt{tan},59$
rep, 17, 18	tanh, 61
rev, 27	text, 261, 262
rexp, 215	$\mathtt{title},256,257$
rgb, 272	transpose, see matrix
rm, 294-296	${ t typeof}, 297$
rmultinom, 220, 221	
rnorm, 219	uniroot, 167
roots	upper.tri, 44
of general single-variable function, 167	var 113-115 117

```
variables
    assigning, 286
    assigning in base environment from func-
         tion, 287
    evaluating from base environment within func-
         tion, 288
    names, 298
variance, see var
vector
    boolean operations on, 159, 160
    containing all indentical entries, 18
    containing all zeros, 17
    counts of binned values in, 164
    counts of discrete values in, 163
    cross product, 80
    cumulative sum of elements of, 131
    differences between consecutive elements of,
         137
    dot product, 79
    minimum of values of, 139
    norm, 102
    position of first occurance of minimum value
         in, 146
    product of all elements, 127
    reversing order of elements in, 27
    size of, 150
    sum of all elements, 124
    truncating, 39
vector, 48
warning, 323
which, 160-162
which.max, see which.min
which.min, 146
while, 190
windows, 235
wireframe, 254
write, 285
x11, 235
```