

NumPy for R (and S-Plus) users

Help

R/S-Plus	Python	Description
<code>help.start()</code>	<code>help()</code>	Browse help interactively
<code>help()</code>	<code>help</code>	Help on using help
<code>help(plot)</code> <i>or</i> <code>?plot</code>	<code>help(plot)</code> <i>or</i> <code>?plot</code>	Help for a function
<code>help(package='splines')</code>	<code>help(pylab)</code>	Help for a toolbox/library package
<code>demo()</code>		Demonstration examples
<code>example(plot)</code>		Example using a function

Searching available documentation

R/S-Plus	Python	Description
<code>help.search('plot')</code>		Search help files
<code>apropos('plot')</code>		Find objects by partial name
<code>library()</code>	<code>help(); modules [Numeric]</code>	List available packages
<code>find(plot)</code>	<code>help(plot)</code>	Locate functions
<code>methods(plot)</code>		List available methods for a function

Using interactively

R/S-Plus	Python	Description
<code>Rgui</code>	<code>ipython -pylab</code>	Start session
	TAB	Auto completion
<code>source('foo.R')</code>	<code>execfile('foo.py')</code> <i>or</i> <code>run foo.py</code>	Run code from file
<code>history()</code>	<code>hist -n</code>	Command history
<code>savehistory(file=".Rhistory")</code>		Save command history
<code>q(save='no')</code>	CTRL-D CTRL-Z # windows <code>sys.exit()</code>	End session

Operators

R/S-Plus	Python	Description
<code>help(Syntax)</code>		Help on operator syntax

Arithmetic operators

R/S-Plus	Python	Description
<code>a<-1; b<-2</code>	<code>a=1; b=1</code>	Assignment; defining a number
<code>a + b</code>	<code>a + b</code> <i>or</i> <code>add(a,b)</code>	Addition
<code>a - b</code>	<code>a - b</code> <i>or</i> <code>subtract(a,b)</code>	Subtraction
<code>a * b</code>	<code>a * b</code> <i>or</i> <code>multiply(a,b)</code>	Multiplication

<code>a / b</code>	<code>a / b</code> <i>or</i> <code>divide(a,b)</code>	Division
<code>a ^ b</code>	<code>a ** b</code> <code>power(a,b)</code> <code>pow(a,b)</code>	Power, a^b
<code>a %% b</code>	<code>a % b</code> <code>remainder(a,b)</code> <code>fmod(a,b)</code>	Remainder
<code>a %/% b</code>		Integer division
	<code>a+=b</code> <i>or</i> <code>add(a,b,a)</code>	In place operation to save array creation overhead
<code>factorial(a)</code>		Factorial, $n!$

Relational operators

R/S-Plus	Python	Description
<code>a == b</code>	<code>a == b</code> <i>or</i> <code>equal(a,b)</code>	Equal
<code>a < b</code>	<code>a < b</code> <i>or</i> <code>less(a,b)</code>	Less than
<code>a > b</code>	<code>a > b</code> <i>or</i> <code>greater(a,b)</code>	Greater than
<code>a <= b</code>	<code>a <= b</code> <i>or</i> <code>less_equal(a,b)</code>	Less than or equal
<code>a >= b</code>	<code>a >= b</code> <i>or</i> <code>greater_equal(a,b)</code>	Greater than or equal
<code>a != b</code>	<code>a != b</code> <i>or</i> <code>not_equal(a,b)</code>	Not Equal

Logical operators

R/S-Plus	Python	Description
<code>a && b</code>	<code>a and b</code>	Short-circuit logical AND
<code>a b</code>	<code>a or b</code>	Short-circuit logical OR
<code>a & b</code>	<code>logical_and(a,b)</code> <i>or</i> <code>a and b</code>	Element-wise logical AND
<code>a b</code>	<code>logical_or(a,b)</code> <i>or</i> <code>a or b</code>	Element-wise logical OR
<code>xor(a, b)</code>	<code>logical_xor(a,b)</code>	Logical EXCLUSIVE OR
<code>!a</code>	<code>logical_not(a)</code> <i>or</i> <code>not a</code>	Logical NOT

root and logarithm

R/S-Plus	Python	Description
<code>sqrt(a)</code>	<code>math.sqrt(a)</code>	Square root
<code>log(a)</code>	<code>math.log(a)</code>	Logarithm, base e (natural)
<code>log10(a)</code>	<code>math.log10(a)</code>	Logarithm, base 10
<code>log2(a)</code>	<code>math.log(a, 2)</code>	Logarithm, base 2 (binary)
<code>exp(a)</code>	<code>math.exp(a)</code>	Exponential function

Round off

R/S-Plus	Python	Description
<code>round(a)</code>	<code>round(a)</code> <i>or</i> <code>math.round(a)</code>	Round
<code>ceil(a)</code>	<code>ceil(a)</code>	Round up
<code>floor(a)</code>	<code>floor(a)</code>	Round down
	<code>fix(a)</code>	Round towards zero

Mathematical constants

R/S-Plus	Python	Description
<code>pi</code>	<code>math.pi</code>	$\pi=3.141592$
<code>exp(1)</code>	<code>math.e</code> <i>or</i> <code>math.exp(1)</code>	$e=2.718281$

Missing values; IEEE-754 floating point status flags

R/S-Plus	Python	Description
	<code>nan</code>	Not a Number
	<code>inf</code>	Infinity, $+\infty$
	<code>plus_inf</code>	Infinity, $+\infty$
	<code>minus_inf</code>	Infinity, $-\infty$
	<code>plus_zero</code>	Plus zero, $+0$
	<code>minus_zero</code>	Minus zero, -0

Complex numbers

R/S-Plus	Python	Description
<code>1i</code>	<code>z = 1j</code>	Imaginary unit
<code>z <- 3+4i</code>	<code>z = 3+4j</code> <i>or</i> <code>z = complex(3,4)</code>	A complex number, $3+4i$
<code>abs(3+4i)</code> <i>or</i> <code>Mod(3+4i)</code>	<code>abs(3+4j)</code>	Absolute value (modulus)
<code>Re(3+4i)</code>	<code>z.real</code>	Real part
<code>Im(3+4i)</code>	<code>z.imag</code>	Imaginary part
<code>Arg(3+4i)</code>		Argument
<code>Conj(3+4i)</code>	<code>z.conj()</code> ; <code>z.conjugate()</code>	Complex conjugate

Trigonometry

R/S-Plus	Python	Description
<code>atan2(b,a)</code>	<code>atan2(b,a)</code>	Arctangent, $\arctan(b/a)$
	<code>hypot(x,y)</code>	Hypotenuse; Euclidean distance

Generate random numbers

R/S-Plus	Python	Description
<code>runif(10)</code>	<code>random.random((10,))</code> <code>random.uniform((10,))</code>	Uniform distribution
<code>runif(10, min=2, max=7)</code>	<code>random.uniform(2,7,(10,))</code>	Uniform: Numbers between 2 and 7
<code>matrix(runif(36),6)</code>	<code>random.uniform(0,1,(6,6))</code>	Uniform: 6,6 array
<code>rnorm(10)</code>	<code>random.standard_normal((10,))</code>	Normal distribution

Vectors

R/S-Plus	Python	Description
<code>a <- c(2,3,4,5)</code>	<code>a=array([2,3,4,5])</code>	Row vector, $1 \times n$ -matrix
<code>adash <- t(c(2,3,4,5))</code>	<code>array([2,3,4,5])[:,NewAxis]</code> <code>array([2,3,4,5]).reshape(-1,1)</code> <code>r_[1:10,'c']</code>	Column vector, $m \times 1$ -matrix

Sequences

R/S-Plus	Python	Description
<code>seq(10)</code> <i>or</i> <code>1:10</code>	<code>arange(1,11, dtype=Float)</code> <code>range(1,11)</code>	1,2,3, ... ,10
<code>seq(0, length=10)</code>	<code>arange(10.)</code>	0.0,1.0,2.0, ... ,9.0
<code>seq(1,10,by=3)</code>	<code>arange(1,11,3)</code>	1,4,7,10
<code>seq(10,1)</code> <i>or</i> <code>10:1</code>	<code>arange(10,0,-1)</code>	10,9,8, ... ,1
<code>seq(from=10,to=1,by=-3)</code>	<code>arange(10,0,-3)</code>	10,7,4,1
<code>seq(1,10,length=7)</code>	<code>linspace(1,10,7)</code>	Linearly spaced vector of n=7 points
<code>rev(a)</code>	<code>a[::-1]</code> <i>or</i> <code>a.fill(3), a[:] = 3</code>	Reverse Set all values to same scalar value

Concatenation (vectors)

R/S-Plus	Python	Description
<code>c(a,a)</code>	<code>concatenate((a,a))</code>	Concatenate two vectors
<code>c(1:4,a)</code>	<code>concatenate((range(1,5),a), axis=1)</code>	

Repeating

R/S-Plus	Python	Description
<code>rep(a,times=2)</code>	<code>concatenate((a,a))</code>	1 2 3, 1 2 3
<code>rep(a,each=3)</code>	<code>a.repeat(3)</code> <i>or</i>	1 1 1, 2 2 2, 3 3 3
<code>rep(a,a)</code>	<code>a.repeat(a)</code> <i>or</i>	1, 2 2, 3 3 3

Miss those elements out

R/S-Plus	Python	Description
<code>a[-1]</code>	<code>a[1:]</code>	miss the first element
<code>a[-10]</code>		miss the tenth element
<code>a[-seq(1,50,3)]</code>		miss 1,4,7, ...
	<code>a[-1]</code>	last element
	<code>a[-2:]</code>	last two elements

Maximum and minimum

R/S-Plus	Python	Description
<code>pmax(a,b)</code>	<code>maximum(a,b)</code>	pairwise max
<code>max(a,b)</code>	<code>concatenate((a,b)).max()</code>	max of all values in two vectors
<code>v <- max(a) ; i <- which.max(a)</code>	<code>v,i = a.max(0),a.argmax(0)</code>	

Vector multiplication

R/S-Plus	Python	Description
<code>a*a</code>	<code>a*a</code>	Multiply two vectors
	<code>dot(u,v)</code>	Vector dot product, $u \cdot v$

Matrices

R/S-Plus	Python	Description
<code>rbind(c(2,3),c(4,5))</code> <code>array(c(2,3,4,5), dim=c(2,2))</code>	<code>a = array([[2,3],[4,5]])</code>	Define a matrix

Concatenation (matrices); `rbind` and `cbind`

R/S-Plus	Python	Description
<code>rbind(a,b)</code>	<code>concatenate((a,b), axis=0)</code> <code>vstack((a,b))</code>	Bind rows
<code>cbind(a,b)</code>	<code>concatenate((a,b), axis=1)</code> <code>hstack((a,b))</code>	Bind columns
	<code>concatenate((a,b), axis=2)</code> <code>dstack((a,b))</code>	Bind slices (three-way arrays)
	<code>concatenate((a,b), axis=None)</code>	Concatenate matrices into one vector
<code>rbind(1:4,1:4)</code>	<code>concatenate((r_[1:5],r_[1:5])).reshape(2,-1)</code> <code>vstack((r_[1:5],r_[1:5]))</code>	Bind rows (from vectors)
<code>cbind(1:4,1:4)</code>		Bind columns (from vectors)

Array creation

R/S-Plus	Python	Description
<code>matrix(0,3,5) or array(0,c(3,5))</code>	<code>zeros((3,5),Float)</code>	0 filled array
	<code>zeros((3,5))</code>	0 filled array of integers
<code>matrix(1,3,5) or array(1,c(3,5))</code>	<code>ones((3,5),Float)</code>	1 filled array
<code>matrix(9,3,5) or array(9,c(3,5))</code>		Any number filled array
<code>diag(1,3)</code>	<code>identity(3)</code>	Identity matrix
<code>diag(c(4,5,6))</code>	<code>diag((4,5,6))</code>	Diagonal
	<code>a = empty((3,3))</code>	Empty array

Reshape and flatten matrices

R/S-Plus	Python	Description
<code>matrix(1:6,nrow=3,byrow=T)</code>	<code>arange(1,7).reshape(2,-1)</code> <code>a.setshape(2,3)</code>	Reshaping (rows first)
<code>matrix(1:6,nrow=2)</code> <code>array(1:6,c(2,3))</code>	<code>arange(1,7).reshape(-1,2).transpose()</code>	Reshaping (columns first)
<code>as.vector(t(a))</code>	<code>a.flatten() or</code>	Flatten to vector (by rows, like comics)
<code>as.vector(a)</code>	<code>a.flatten(1)</code>	Flatten to vector (by columns)
<code>a[row(a) <= col(a)]</code>		Flatten upper triangle (by columns)

Shared data (slicing)

R/S-Plus	Python	Description
<code>b = a</code>	<code>b = a.copy()</code>	Copy of a

Indexing and accessing elements (Python: slicing)

R/S-Plus	Python	Description
<code>a <- rbind(c(11, 12, 13, 14), c(21, 22, 23, 24), c(31, 32, 33, 34))</code>	<code>a = array([[11, 12, 13, 14], [21, 22, 23, 24], [31, 32, 33, 34]])</code>	Input is a 3,4 array
<code>a[2,3]</code>	<code>a[1,2]</code>	Element 2,3 (row,col)
<code>a[1,]</code>	<code>a[0,]</code>	First row
<code>a[,1]</code>	<code>a[:,0]</code>	First column
	<code>a.take([0,2]).take([0,3], axis=1)</code>	Array as indices
<code>a[-1,]</code>	<code>a[1:,:] </code>	All, except first row
	<code>a[-2:,:] </code>	Last two rows
	<code>a[:,2::] </code>	Strides: Every other row
	<code>a[... ,2]</code>	Third in last dimension (axis)
<code>a[-2,-3]</code>		All, except row,column (2,3)
<code>a[, -2]</code>	<code>a.take([0,2,3],axis=1)</code>	Remove one column
	<code>a.diagonal(offset=0)</code>	Diagonal

Assignment

R/S-Plus	Python	Description
<code>a[,1] <- 99</code>	<code>a[:,0] = 99</code>	
<code>a[,1] <- c(99,98,97)</code>	<code>a[:,0] = array([99,98,97])</code>	
<code>a[a>90] <- 90</code>	<code>(a>90).choose(a,90)</code> <code>a.clip(min=None, max=90)</code>	Clipping: Replace all elements over 90
	<code>a.clip(min=2, max=5)</code>	Clip upper and lower values

Transpose and inverse

R/S-Plus	Python	Description
<code>t(a)</code>	<code>a.conj().transpose()</code>	Transpose
	<code>a.transpose()</code>	Non-conjugate transpose
<code>det(a)</code>	<code>linalg.det(a)</code> <i>or</i>	Determinant
<code>solve(a)</code>	<code>linalg.inv(a)</code> <i>or</i>	Inverse
<code>ginv(a)</code>	<code>linalg.pinv(a)</code>	Pseudo-inverse
	<code>norm(a)</code>	Norms
<code>eigen(a)\$values</code>	<code>linalg.eig(a)[0]</code>	Eigenvalues
<code>svd(a)\$d</code>	<code>linalg.svd(a)</code>	Singular values
	<code>linalg.cholesky(a)</code>	Cholesky factorization
<code>eigen(a)\$vectors</code>	<code>linalg.eig(a)[1]</code>	Eigenvectors
<code>rank(a)</code>	<code>rank(a)</code>	Rank

Sum

R/S-Plus	Python	Description
<code>apply(a,2,sum)</code>	<code>a.sum(axis=0)</code>	Sum of each column
<code>apply(a,1,sum)</code>	<code>a.sum(axis=1)</code>	Sum of each row
<code>sum(a)</code>	<code>a.sum()</code>	Sum of all elements
	<code>a.trace(offset=0)</code>	Sum along diagonal
<code>apply(a,2,cumsum)</code>	<code>a.cumsum(axis=0)</code>	Cumulative sum (columns)

Sorting

R/S-Plus	Python	Description
	<code>a = array([[4,3,2],[2,8,6],[1,4,7]])</code>	Example data
<code>t(sort(a))</code>	<code>a.ravel().sort()</code> <i>or</i>	Flat and sorted
<code>apply(a,2,sort)</code>	<code>a.sort(axis=0)</code> <i>or</i> <code>msort(a)</code>	Sort each column
<code>t(apply(a,1,sort))</code>	<code>a.sort(axis=1)</code>	Sort each row
	<code>a[a[:,0].argsort(),]</code>	Sort rows (by first row)
<code>order(a)</code>	<code>a.ravel().argsort()</code>	Sort, return indices
	<code>a.argsort(axis=0)</code>	Sort each column, return indices
	<code>a.argsort(axis=1)</code>	Sort each row, return indices

Maximum and minimum

R/S-Plus	Python	Description
<code>apply(a,2,max)</code>	<code>a.max(0)</code> <i>or</i> <code>amax(a[,axis=0])</code>	max in each column
<code>apply(a,1,max)</code>	<code>a.max(1)</code> <i>or</i> <code>amax(a,axis=1)</code>	max in each row
<code>max(a)</code>	<code>a.max()</code> <i>or</i>	max in array
<code>i <- apply(a,1,which.max)</code>		return indices, i
<code>pmax(b,c)</code>	<code>maximum(b,c)</code>	pairwise max
<code>apply(a,2,cummax)</code>		
	<code>a.ptp(); a.ptp(0)</code>	max-to-min range

Matrix manipulation

R/S-Plus	Python	Description
<code>a[,4:1]</code>	<code>flip1r(a)</code> <i>or</i> <code>a[:,::-1]</code>	Flip left-right
<code>a[3:1,]</code>	<code>flipud(a)</code> <i>or</i> <code>a[::-1,]</code>	Flip up-down
	<code>rot90(a)</code>	Rotate 90 degrees
<code>kronecker(matrix(1,2,3),a)</code>	<code>kron(ones((2,3)),a)</code>	Repeat matrix: [a a a ; a a a]
<code>a[lower.tri(a)] <- 0</code>	<code>triu(a)</code>	Triangular, upper
<code>a[upper.tri(a)] <- 0</code>	<code>tril(a)</code>	Triangular, lower

Equivalents to "size"

R/S-Plus	Python	Description
<code>dim(a)</code>	<code>a.shape</code> <i>or</i> <code>a.getshape()</code>	Matrix dimensions
<code>ncol(a)</code>	<code>a.shape[1]</code> <i>or</i> <code>size(a,axis=1)</code>	Number of columns
<code>prod(dim(a))</code>	<code>a.size</code> <i>or</i> <code>size(a[,axis=None])</code>	Number of elements
	<code>a.ndim</code>	Number of dimensions
<code>object.size(a)</code>	<code>a.nbytes</code>	Number of bytes used in memory

Matrix- and elementwise- multiplication

R/S-Plus	Python	Description
<code>a * b</code>	<code>a * b</code> <i>or</i> <code>multiply(a,b)</code>	Elementwise operations
<code>a %*% b</code>	<code>matrixmultiply(a,b)</code>	Matrix product (dot product)
	<code>inner(a,b)</code> <i>or</i>	Inner matrix vector multiplication

		$a \cdot b$
<code>outer(a,b)</code> <i>or</i> <code>a %o% b</code>	<code>outer(a,b)</code> <i>or</i>	Outer product
<code>crossprod(a,b)</code> <i>or</i> <code>t(a) %*% b</code>		Cross product
<code>kronecker(a,b)</code>	<code>kron(a,b)</code>	Kronecker product
<code>solve(a,b)</code>	<code>linalg.solve(a,b)</code>	Left matrix division, $b^{-1} \cdot a$ \newline (solve linear equations)
	<code>vdot(a,b)</code>	Vector dot product
	<code>cross(a,b)</code>	Cross product

Find; conditional indexing

R/S-Plus	Python	Description
<code>which(a != 0)</code>	<code>a.ravel().nonzero()</code>	Non-zero elements, indices
<code>which(a != 0, arr.ind=T)</code>	<code>(i,j) = a.nonzero()</code> <code>(i,j) = where(a!=0)</code>	Non-zero elements, array indices
<code>ij <- which(a != 0, arr.ind=T);</code> <code>v <- a[ij]</code>	<code>v = a.compress((a!=0).flat)</code> <code>v = extract(a!=0,a)</code>	Vector of non-zero values
<code>which(a>5.5)</code>	<code>(a>5.5).nonzero()</code>	Condition, indices
<code>ij <- which(a>5.5, arr.ind=T);</code> <code>v <- a[ij]</code>	<code>a.compress((a>5.5).flat)</code>	Return values
	<code>where(a>5.5,0,a)</code> <i>or</i> <code>a * (a>5.5)</code>	Zero out elements above 5.5
	<code>a.put(2, indices)</code>	Replace values

Multi-way arrays

R/S-Plus	Python	Description
	<code>a = array([[[[1,2],[1,2]], [[3,4],[3,4]]]])</code> <code>a[0,...]</code>	Define a 3-way array

File input and output

R/S-Plus	Python	Description
<code>f <- read.table("data.txt")</code>	<code>f = fromfile("data.txt")</code> <code>f = load("data.txt")</code>	Reading from a file (2d)
<code>f <- read.table("data.txt")</code>	<code>f = load("data.txt")</code>	Reading from a file (2d)
<code>f <- read.table(file="data.csv", sep=";")</code>	<code>f = load('data.csv', delimiter=';')</code>	Reading from a CSV file (2d)
<code>write(f,file="data.txt")</code>	<code>save('data.csv', f, fmt='%.6f', delimiter=';')</code>	Writing to a file (2d)
	<code>f.tofile(file='data.csv', format='%.6f', sep=';')</code>	Writing to a file (1d)
	<code>f = fromfile(file='data.csv', sep=';')</code>	Reading from a file (1d)

Plotting

Basic x-y plots

R/S-Plus	Python	Description
----------	--------	-------------

<code>plot(a, type="l")</code>	<code>plot(a)</code>	1d line plot
<code>plot(x[,1],x[,2])</code>	<code>plot(x[:,0],x[:,1], 'o')</code>	2d scatter plot
	<code>plot(x1,y1, 'bo', x2,y2, 'go')</code>	Two graphs in one plot
<code>plot(x1,y1)</code> <code>matplot(x2,y2,add=T)</code>	<code>plot(x1,y1, 'o')</code> <code>plot(x2,y2, 'o')</code> <code>show() # as normal</code>	Overplotting: Add new plots to current
	<code>subplot(211)</code>	subplots
<code>plot(x,y,type="b",col="red")</code>	<code>plot(x,y, 'ro-')</code>	Plotting symbols and color

Axes and titles

R/S-Plus	Python	Description
<code>grid()</code>	<code>grid()</code>	Turn on grid lines
<code>plot(c(1:10,10:1), asp=1)</code>	<code>figure(figsize=(6,6))</code>	1:1 aspect ratio
<code>plot(x,y, xlim=c(0,10), ylim=c(0,5))</code>	<code>axis([0, 10, 0, 5])</code>	Set axes manually
<code>plot(1:10, main="title", xlab="x-axis", ylab="y-axis")</code>		Axis labels and titles
	<code>text(2,25, 'hello')</code>	Insert text

Log plots

R/S-Plus	Python	Description
<code>plot(x,y, log="y")</code>	<code>semilogy(a)</code>	logarithmic y-axis
<code>plot(x,y, log="x")</code>	<code>semilogx(a)</code>	logarithmic x-axis
<code>plot(x,y, log="xy")</code>	<code>loglog(a)</code>	logarithmic x and y axes

Filled plots and bar plots

R/S-Plus	Python	Description
<code>plot(t,s, type="n", xlab="", ylab="")</code> <code>polygon(t,s, col="lightblue")</code> <code>polygon(t,c, col="lightgreen")</code>	<code>fill(t,s, 'b', t,c, 'g', alpha=0.2)</code>	Filled plot
<code>stem(x[,3])</code>		Stem-and-Leaf plot

Functions

R/S-Plus	Python	Description
<code>f <- function(x) sin(x/3) - cos(x/5)</code>		Defining functions
<code>plot(f, xlim=c(0,40), type='p')</code>	<code>x = arange(0,40,.5)</code> <code>y = sin(x/3) - cos(x/5)</code> <code>plot(x,y, 'o')</code>	Plot a function for given range

Polar plots

R/S-Plus	Python	Description
	<code>theta = arange(0,2*pi,0.001)</code> <code>r = sin(2*theta)</code> <code>polar(theta, rho)</code>	

Histogram plots

R/S-Plus	Python	Description
<code>hist(rnorm(1000))</code>		
<code>hist(rnorm(1000), breaks= -4:4)</code>		
<code>hist(rnorm(1000), breaks=c(seq(-5,0,0.25), seq(0.5,5,0.5)), freq=F)</code>		
<code>plot(apply(a,1,sort),type="l")</code>		

3d data

Contour and image plots

R/S-Plus	Python	Description
<code>contour(z)</code>	<code>levels, colls = contour(Z, V, origin='lower', extent= (-3,3,-3,3)) clabel(colls, levels, inline=1, fmt='%1.1f', fontsize=10)</code>	Contour plot
<code>filled.contour(x,y,z, nlevels=7, color=gray.colors)</code>	<code>contourf(Z, V, cmap=cm.gray, origin='lower', extent=(-3,3,-3,3))</code>	Filled contour plot
<code>image(z, col=gray.colors(256))</code>	<code>im = imshow(Z, interpolation='bilinear', origin='lower', extent=(-3,3,-3,3))</code>	Plot image data
	<code># imshow() and contour() as above</code>	Image with contours
	<code>quiver()</code>	Direction field vectors

Perspective plots of surfaces over the x-y plane

R/S-Plus	Python	Description
<code>f <- function(x,y) x*exp(-x^2- y^2)</code> <code>n <- seq(-2,2, length=40)</code> <code>z <- outer(n,n,f)</code>	<code>n=arrayrange(-2,2,.1)</code> <code>[x,y] = meshgrid(n,n)</code> <code>z = x*power(math.e,-x**2-y**2)</code>	
<code>persp(x,y,z, theta=30, phi=30, expand=0.6, ticktype='detailed')</code>		Mesh plot
<code>persp(x,y,z, theta=30, phi=30, expand=0.6, col='lightblue', shade=0.75, ltheta=120, ticktype='detailed')</code>		Surface plot

Scatter (cloud) plots

R/S-Plus	Python	Description
<code>cloud(z~x*y)</code>		3d scatter plot

Save plot to a graphics file

R/S-Plus	Python	Description
<code>postscript(file="foo.eps")</code> <code>plot(1:10)</code> <code>dev.off()</code>	<code>savefig('foo.eps')</code>	PostScript
<code>pdf(file='foo.pdf')</code>	<code>savefig('foo.pdf')</code>	PDF
<code>devSVG(file='foo.svg')</code>	<code>savefig('foo.svg')</code>	SVG (vector graphics for www)
<code>png(filename = "Rplot%03d.png"</code>	<code>savefig('foo.png')</code>	PNG (raster graphics)

Data analysis

Set membership operators

R/S-Plus	Python	Description
<code>a <- c(1,2,2,5,2)</code> <code>b <- c(2,3,4)</code>	<code>a = array([1,2,2,5,2])</code> <code>b = array([2,3,4])</code> <code>a = set([1,2,2,5,2])</code> <code>b = set([2,3,4])</code>	Create sets
<code>unique(a)</code>	<code>unique1d(a)</code> <code>unique(a)</code> <code>set(a)</code>	Set unique
<code>union(a,b)</code>	<code>union1d(a,b)</code> <code>a.union(b)</code>	Set union
<code>intersect(a,b)</code>	<code>intersect1d(a)</code> <code>a.intersection(b)</code>	Set intersection
<code>setdiff(a,b)</code>	<code>setdiff1d(a,b)</code> <code>a.difference(b)</code>	Set difference
<code>setdiff(union(a,b), intersect(a,b))</code>	<code>setxor1d(a,b)</code> <code>a.symmetric_difference(b)</code>	Set exclusion
<code>is.element(2,a) <i>or</i> 2 %in% a</code>	<code>2 in a</code> <code>setmember1d(2,a)</code> <code>contains(a,2)</code>	True for set member

Statistics

R/S-Plus	Python	Description
<code>apply(a,2,mean)</code>	<code>a.mean(axis=0)</code> <code>mean(a [,axis=0])</code>	Average
<code>apply(a,2,median)</code>	<code>median(a) <i>or</i> median(a [,axis=0])</code>	Median
<code>apply(a,2,sd)</code>	<code>a.std(axis=0) <i>or</i> std(a [,axis=0])</code>	Standard deviation
<code>apply(a,2,var)</code>	<code>a.var(axis=0) <i>or</i> var(a)</code>	Variance
<code>cor(x,y)</code>	<code>correlate(x,y) <i>or</i> corrcoef(x,y)</code>	Correlation coefficient
<code>cov(x,y)</code>	<code>cov(x,y)</code>	Covariance

Interpolation and regression

R/S-Plus	Python	Description
<code>z <- lm(y~x)</code> <code>plot(x,y)</code> <code>abline(z)</code>	<code>(a,b) = polyfit(x,y,1)</code> <code>plot(x,y,'o', x,a*x+b,'-')</code>	Straight line fit

<code>solve(a,b)</code>	<code>linalg.lstsq(x,y)</code>	Linear least squares $y = ax + b$
	<code>polyfit(x,y,3)</code>	Polynomial fit

Non-linear methods

Polynomials, root finding

R/S-Plus	Python	Description
	<code>poly()</code>	Polynomial
<code>polyroot(c(1,-1,-1))</code>	<code>roots()</code>	Find zeros of polynomial
	<code>polyval(array([1,2,1,2]),arange(1,11))</code>	Evaluate polynomial

Differential equations

R/S-Plus	Python	Description
	<code>diff(x, n=1, axis=0)</code>	Discrete difference function and approximate derivative

Fourier analysis

R/S-Plus	Python	Description
<code>fft(a)</code>	<code>fft(a)</code> <i>or</i>	Fast fourier transform
<code>fft(a, inverse=TRUE)</code>	<code>ifft(a)</code> <i>or</i>	Inverse fourier transform
	<code>convolve(x,y)</code>	Linear convolution

Symbolic algebra; calculus

R/S-Plus Python Description

Programming

R/S-Plus	Python	Description
<code>.R</code>	<code>.py</code>	Script file extension
<code>#</code>	<code>#</code>	Comment symbol (rest of line)
<code>library(RSvgDevice)</code>	<code>from pylab import *</code>	Import library functions
<code>string <- "a <- 234"</code> <code>eval(parse(text=string))</code>	<code>string="a=234"</code> <code>eval(string)</code>	Eval

Loops

R/S-Plus	Python	Description
<code>for(i in 1:5) print(i)</code>	<code>for i in range(1,6): print(i)</code>	for-statement
<code>for(i in 1:5) {</code> <code>print(i)</code> <code>print(i*2)</code> <code>}</code>	<code>for i in range(1,6):</code> <code>print(i)</code> <code>print(i*2)</code>	Multiline for statements

Conditionals

R/S-Plus	Python	Description
<code>if (1>0) a <- 100</code>	<code>if 1>0: a=100</code>	if-statement

```
ifelse(a>0,a,0)
```

Ternary operator (if?true:false)

Debugging

R/S-Plus	Python	Description
<code>.Last.value</code>		Most recent evaluated expression
<code>objects()</code>		List variables loaded into memory
<code>rm(x)</code>		Clear variable \$x\$ from memory
<code>print(a)</code>	<code>print a</code>	Print

Working directory and OS

R/S-Plus	Python	Description
<code>list.files()</code> <i>or</i> <code>dir()</code>	<code>os.listdir(".")</code>	List files in directory
<code>list.files(pattern="\.r\$")</code>	<code>grep.grep("*.py")</code>	List script files in directory
<code>getwd()</code>	<code>os.getcwd()</code>	Displays the current working directory
<code>setwd('foo')</code>	<code>os.chdir('foo')</code>	Change working directory
<code>system("notepad")</code>	<code>os.system('notepad')</code> <code>os.popen('notepad')</code>	Invoke a System Command

Time-stamp: "2007-11-09T16:46:36 vidar"

©2006 Vidar Bronken Gundersen, /mathesaurus.sf.net

Permission is granted to copy, distribute and/or modify this document as long as the above attribution is retained.