The QuantEcon MATLAB-Python-Julia Cheat Sheet

QuantEcon

September 19, 2016

CONTENTS

This document summarizes commonly-used, equivalent commands across MATLAB, Python, and Julia.

CHAPTER

ONE

CREATING VECTORS

Operation	MATLAB	Python	Julia
Create a row vector	`A = [1 2 3]`	`A = np.array([1, 2, 3])`	`A = [1 2 3]`
Create a column vector	`A = [1; 2; 3]`	`A = np.array([1, 2, 3]).reshape(3, 1)`	`A = [1; 2; 3]`
Sequence starting at j ending at n, with difference k between points	`A = j:k:n`	`A = np.arange(j, n+1, k)`	`A = j:k:n`
Linearly spaced vector of k points	`A = linspace(1, 5, k)`	`A = np.linspace(1, 5, k)`	`A = linspace(1, 5, k)`

CHAPTER

TWO

CREATING MATRICES

Operation	MATLAB	Python	Julia
Create a matrix	`A = [1 2; 3 4]`	`A = np.array([[1, 2], [3, 4]])`	`A = [1 2; 3 4]`
Create a 2 by 2 matrix of zeros	`A = zeros(2, 2)`	`A = np.zeros((2, 2))`	`A = zeros(2, 2)`
Create a 2 by 2 matrix of ones	`A = ones(2, 2)`	`A = np.ones((2, 2))`	`A = ones(2, 2)`
Create a 2 by 2 identity matrix	`A = eye(2, 2)`	`A = np.eye(2)`	`A = eye(2, 2)`
Create a diagonal matrix	`A = diag([1 2 3])`	`A = np.diag([1, 2, 3])`	`A = diagm([1; 2; 3])`
Matrix of uniformly distributed random numbers	`A = rand(2, 2)`	`A = np.random.rand(2, 2)`	`A = rand(2, 2)`
Matrix of random numbers drawn a standard normal	`A = randn(2, 2)`	`A = np.random.randn(2, 2)`	`A = randn(2, 2)`

CHAPTER

THREE

MANIPULATING VECTORS AND MATRICES

Operation	MATLAB	Python	Julia
Transpose	`A'`	`A.T`	`A'`

Concatenate horizontally	`A = [[1 2] [1 2]]`
	or
	`A = horzcat([1 2], [1 2])`
		`B = np.array([1, 2])`
		`A = np.hstack((B, B))`
			`A = [[1 2] [1 2]]`
			or
			`A = hcat([1 2], [1 2])`

Concatenate vertically	`A = [[1 2]; [1 2]]`
	or
	`A = vertcat([1 2], [1 2])`
		`B = np.array([1, 2])`
		`A = np.vstack((B, B))`
			`A = [[1 2]; [1 2]]`
			or
			`A = vcat([1 2], [1 2])`

Reshape (to 5 rows, 2 columns)	`A = reshape(1:10, 5, 2)`	`A = A.reshape(5,2)`	`A = reshape(1:10, 5, 2)`
Convert matrix to vector	`A(:)`	`A = A.flatten()`	`A[:]`
Flip left/right	`fliplr(A)`	`np.fliplr(A)`	`flipdim(A, 2)`
Flip up/down	`flipud(A)`	`np.flipud(A)`	`flipdim(A, 1)`
Repeat matrix (3 times in the row dimension, 4 times in the column dimension)	`repmat(A, 3, 4)`	`np.tile(A, (4, 3))`	`repmat(A, 3, 4)`

CHAPTER

FOUR

ACCESSING VECTOR/MATRIX ELEMENTS

Operation	MATLAB	Python	Julia
Access one element	`A(2, 2)`	`A[2, 2]`	`A[2, 2]`
Access specific rows	`A(1:4, :)`	`A[0:4, :]`	`A[1:4, :]`
Access specific columns	`A(:, 1:4)`	`A[:, 0:4]`	`A[:, 1:4]`
Remove a row	`A([1 2 4], :)`	`A[[0, 1, 3], :]`	`A[[1, 2, 4], :]`
Diagonals of matrix	`diag(A)`	`np.diag(A)`	`diag(A)`
Get dimensions of matrix	`[nrow ncol] = size(A)`	`nrow, ncol = np.shape(A)`	`nrow, ncol = size(A)`

CHAPTER

FIVE

MATHEMATICAL OPERATIONS

Operation	MATLAB	Python	Julia
Vector dot product	`dot(A, B)`	`np.dot(A, B)` **or** `A@B`	`dot(A, B)`
Matrix multiplication	`A*B`	`np.dot(A, B)` **or** `A@B`	`A*B`
Element-wise matrix multiplication	`A.*B`	`A*B`	`A.*B`
Matrix to a power	`A^2`	`np.linalg.matrix_power(A, 2)`	`A^2`
Matrix to a power, element-wise	`A.^2`	`A**2`	`A.^2`
Inverse of a matrix	`inv(A)` or `A^(-1)`	`np.linalg.inv(A)`	`inv(A)` or `A^(-1)`
Determinant of a matrix	`det(A)`	`np.linalg.det(A)`	`det(A)`
Eigenvalues and eigenvectors	`[vec, val] = eig(A)`	`val, vec = np.linalg.eig(A)`	`val, vec = eig(A)`
Euclidean norm	`norm(A)`	`np.linalg.norm(A)`	`norm(A)`
Solve linear system $Ax = b$	`A\b`	`np.linalg.solve(A, b)`	`A\b`

CHAPTER

SIX

SUM/MAXIMUM/MINIMUM

Operation	MATLAB	Python	Julia

Sum/maximum/minimum of each column	`sum(A, 1)`
	`max(A, [], 1)`
	`min(A, [], 1)`
		`sum(A, 0)`
		`np.amax(A, 0)`
		`np.amin(A, 0)`
			`sum(A, 1)`
			`maximum(A, 1)`
			`minimum(A, 1)`

Sum/maximum/minimum of each row	`sum(A, 2)`
	`max(A, [], 2)`
	`min(A, [], 2)`
		`sum(A, 1)`
		`np.amax(A, 1)`
		`np.amin(A, 1)`
			`sum(A, 2)`
			`maximum(A, 2)`
			`minimum(A, 2)`

Sum/maximum/minimum of entire matrix	`sum(A(:))`
	`max(A(:))`
	`min(A(:))`
		`np.sum(A)`
		`np.amax(A)`
		`np.amin(A)`
			`sum(A)`
			`maximum(A)`
			`minimum(A)`

Cumulative sum/maximum/minimum by row	`cumsum(A, 1)`
	`cummax(A, 1)`
	`cummin(A, 1)`
		`np.cumsum(A, 0)`
		`np.maximum.accumulate(A, 0)`
		`np.minimum.accumulate(A, 0)`
			`cumsum(A, 1)`
			`cummax(A, 1)`
			`cummin(A, 1)`

Cumulative sum/maximum/minimum by column	`cumsum(A, 2)`
	`cummax(A, 2)`
	`cummin(A, 2)`
		`np.cumsum(A, 1)`
		`np.maximum.accumulate(A, 1)`
		`np.minimum.accumulate(A, 1)`
			`cumsum(A, 2)`
			`cummax(A, 2)`
			`cummin(A, 2)`

CHAPTER

SEVEN

PROGRAMMING

Operation	MATLAB	Python	Julia
Comment one line	`% This is a comment`	`# This is a comment`	`# This is a comment`

Comment block	`%{`
	`Comment block`
	`%}`
		`# Block`
		`# comment`
		`# following PEP8`
			`#=`
			`Comment block`
			`=#`

For loop	`for i = 1:N`
	`  % do something`
	`end`
		`for i in range(n):`
		`  # do something`
			`for i = 1:N`
			`  # do something`
			`end`

While loop	`while i <= N`
	`  % do something`
	`end`
		`while i <= N:`
		`  # do something`
			`while i <= N`
			`  # do something`
			`end`

If statement	`if i <= N`
	`  % do something`
	`end`
		`if i <= N:`
		`  # do something`
			`if i <= N`
			`  # do something`
			`end`

If/else statement	`if i <= N`
	`  % do something`
	`else`
	`  % do something else`
	`end`
		`if i <= N:`
		`  # do something`
		`else:`
		`  # do something else`
			`if i <= N`
			`  # do something`
			`else`
			`  # do something else`
			`end`

Print text and variable to screen	`x = 10` `fprintf('The value of x is %d. \n', x)`	`x = 10` `print('The value of x is {}.'.format(x))`	`x = 10` `println("The value of x is $(x).")`
Function: one line/ anonymous	`fun = @(x) x^2`	`fun = lambda x: x**2`	`fun(x) = x^2`

Function: multiple lines	`function out = fun(x)`
	`  out = x^2`
	`end`
		`def fun(x):`
		`  return x**2`
			`function fun(x)`
			`  return x^2`
			`end`