Bio 723: Class Session 3
Matrix Operations

September 9, 2014

```
In []: %matplotlib inline
```

```
In []: import numpy as np
       from matplotlib import pyplot as plt
       import pandas as pd
```

1 Creating Matrices and Accessing Elements using Numpy

Two-dimensional NumPy arrays are used to represent matrices in Python.

```
In []: # create the matrix by hand
       X = np.array([[1,2,3],
                     [3,4,5],
                     [7,8,9]])
```

```
In []: X
```

```
In []: # initiate the matrix with a list
       X = np.array(range(1,10))
```

```
In []: X
```

```
In []: # examine the shape
       X.shape
```

```
In []: # reshape the matrix
       X.shape = 3,3
```

```
In []: X
```

```
In []: # Access elements
       X[1,1]
```

```
In []: X[2,1]
```

```
In []: # Access rows
       X[1,:]
```

```
In []: # Access columns
       X[:,1]
```

```
In []: # note that the column was returned as a row-vector
       # To get it as a column vector
       X[:,[1]]
```

```
In []: # get 0th and 2nd rows
       X[[0,2],:]

In []: # get first columns and everything after
       X[:,1:]

In []: # get the diagonal elements of the matrix
       X.diagonal()
```

2 Matrix Arithmetic

```
In []: A = np.array(range(1,13))
       A.shape = 4,3

In []: A

In []: # np.random.seed seeds the random number generator
       # giving a specific seeds allows us to generate random numbers
       # deterministically so that you're results will match mine

       np.random.seed(20140909)
       B = np.random.normal(1, 1, size=(4,3))

In []: B

In []: # matrix addition and subtraction
       A + B

In []: A - B

In []: # scalar multiplication and division
       A * 3.

In []: A / 3.0

In []: # comapre to above
       A / 3

In []: np.dot(A,B)  # doesn't work because not conformable!

In []: np.dot(A, B.transpose())

In []: # why is this false?
       np.dot(A, B.transpose()) == np.dot(B.transpose(), A)

In []: # matrix inverse is in the linalg submodule of numpy
       from numpy import linalg as la

In []: la.inv(A)  # did this work? why or why not?

In []: C = np.random.normal(size=(4,4))

In []: C

In []: invC = la.inv(C)

In []: invC
```

```
In []: print invC

In []: # "pretty print" the matrix
       print np.array2string(invC, suppress_small=True)

In []: # mathematically these should be equal
       np.dot(C, la.inv(C)) == np.identity(4)

In []: # use allclose to test approx. equality for floating point values
       # see allclose() help for more info on arguments

       identity4 = np.identity(4)
       C_invC = np.dot(C, la.inv(C))
       np.allclose(identity4, C_invC, rtol=1e-9)
```