Exercises02

February 21, 2015

0.1 Computer lab 02

These exercises provide more practice in data manipulation and working with numpy arrays.

```
In [1]: import os
        import sys
        import glob
        import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        %matplotlib inline
        %precision 4
        plt.style.use('ggplot')
```

Exercise 1 [**10 pts**]. Write a 12 by 12 times table chart without explicit looping (i.e. no for, while or comprehensions). Your code should generate this output:

```
[[  1   2   3   4   5   6   7   8   9  10  11  12]
 [  2   4   6   8  10  12  14  16  18  20  22  24]
 [  3   6   9  12  15  18  21  24  27  30  33  36]
 [  4   8  12  16  20  24  28  32  36  40  44  48]
 [  5  10  15  20  25  30  35  40  45  50  55  60]
 [  6  12  18  24  30  36  42  48  54  60  66  72]
 [  7  14  21  28  35  42  49  56  63  70  77  84]
 [  8  16  24  32  40  48  56  64  72  80  88  96]
 [  9  18  27  36  45  54  63  72  81  90  99 108]
 [ 10  20  30  40  50  60  70  80  90 100 110 120]
 [ 11  22  33  44  55  66  77  88  99 110 121 132]
 [ 12  24  36  48  60  72  84  96 108 120 132 144]]
```

```
In [2]: # Your code here
```

Exercise 2 [**10 pts**]. Create a new matrix that normalizes the given matrix so that all *columns* sum to 1.0 without using any loops. Create another matrix so that all *rows* sum to 1.0. In other words, if the 3 matrices were `xs` (given), `ys` (column normalized) and `zs` (row normalized), we would have

`ys.sum(axis=0)` = `[ 1., 1., 1., 1., 1., 1.]`

and

`zs.sum(axis=1)` = `[ 1., 1., 1., 1.]`

Start by creating the following matrix `xs`

```
[[  1.   2.   3.   4.   5.   6.]
 [  7.   8.   9.  10.  11.  12.]
 [ 13.  14.  15.  16.  17.  18.]
 [ 19.  20.  21.  22.  23.  24.]]
```

```
In [3]: # Your code here
```

Exercise 3 [10 pts]. In this exercise, we will practice using Pandas dataframes to explore and summarize a data set `heart`.

This data contains the survival time after receiving a heart transplant, the age of the patient and whether or not the survival time was censored.

- Number of Observations - 69
- Number of Variables - 3

Variable name definitions::

- survival - Days after surgery until death
- censors - indicates if an observation is censored. 1 is uncensored
- age - age at the time of surgery

Answer the following questions with respect to the `heart` data set:

- How many patients were censored?
- What is the correlation coefficient between age and survival for uncensored patients?
- What is the average age for censored and uncensored patients?
- What is the average survival time for censored and uncensored patients under the age of 45?
- What is the survival time of the youngest and oldest uncensored patient?

```
In [4]: import statsmodels.api as sm
        heart = sm.datasets.heart.load_pandas().data

        heart.head(n=6)
```

```
Out[4]:    survival  censors   age
        0        15        1  54.3
        1         3        1  40.4
        2       624        1  51.0
        3        46        1  42.5
        4       127        1  48.0
        5        64        1  54.6
```

```
In [5]: import statsmodels.api as sm
        heart = sm.datasets.heart.load_pandas().data

        # Your code here
```

Exercise 4 [20 pts]. Normalize the given matrix M so that all rows sum to 1.0 (as in Exercise 2). This can then be considered as a transition matrix P for a Markov chain. Find the stationary distribution of this matrix in the following ways using `numpy` and `numpy.linalg` (or `scipy.linalg`):

- By repeated matrix multiplication of a random probabilty vector v (a row vector normalized to sum to 1.0) with P using matrix multiplication with `np.dot`.
- By raising the matrix P to some large power unitl it doesn't change with higher powers (see `np.linalg.matrix_power`) and then calculating vP
- From the equation for stationarity $wP = w$, we can see that w must be a left eigenvector of P with eigenvalue 1 (Note: np.linalg.eig returns the right eigenvectors, but the left eighenvector of a matrix is the right eigenvector of the transposed matrix). Use this to find w using `np.linalg.eig`.

- Suppose $w = (w_1, w_2, w_3)$. Then from $wP = w$, we have:

$$w_1 P_{11} + w_2 P_{21} + w_3 P_{31} = w_1 \tag{1}$$
$$w_1 P_{12} + w_2 P_{22} + w_3 P_{32} = w_2 \tag{2}$$
$$w_1 P_{13} + w_2 P_{23} + w_3 P_{331} = w_3 \tag{3}$$
$$\tag{4}$$

  This is a singular system, but we also know that $w_1 + w_2 + w_3 = 1$. Use these facts to set up a linear system of equations that can be solved with `np.linalg.solve` to find w.

Given matrix M

```
[[7, 8, 8],
 [1, 3, 8],
 [9, 2, 1]]
```

In [6]:
```
# Your code here
```