HW2 (Score: 14.5 / 20.0)

DATA 601: HW2

Fall 2019

Due: Wed. Sep. 25, 2019 (by 23:55)

Learning Objectives

- Work with realworld datasets that can be represented using linear data structures.
- Apply vectorization concepts to an iterative problem.
- Explore different programming paradigms to solve problems.

This is an individual homework assignment.

Please complete this homework assignment within the Jupypter notebook environment, making use of Markdown and Code cells to properly format your answers.

Your completed Jupyter notebook is to be submitted via the HW2 dropbox on D2L.

(Top)

Question 1 (8 points):

Visualizing trends in an index

This question asks you to plot the Bitcoin price index (BPI) along with 5-day and 10-day averages. Please execute the code cell below; it will read in a csv file containing the daily closing price from Sep. 1, 2018 to Aug. 31, 2019 (data obtained from https://www.coindesk.com/price/ (https://www.coindesk.com/price/)). Perform the following tasks. You may use any built-in Python functions as well as data strucutres and functions provided by the `numpy` library.

- Observe that the closing prices are at a daily interval. We therefore do not need the date information. Clean up the data and only retain the price information. Store the result in a list or a numpy array in floating point format.
- Recall that a simple moving average (https://en.wikipedia.org/wiki/Moving_average#Simple_moving_average) is defined as the (unweighted) mean over the previous N days.
  Perform a simple moving average of the price index. The number of days N to average over should be adjustable. If you are using `numpy`, you may find the function `np.convolve` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.convolve.html) helpful.
- Plot the raw price index data along with 5-day and 10-day simple moving average. Plot on the same figure in order to help you visually ascertain the effect of the filter.
- What is the effect of the moving average filter? In what circumstances would you *not* want to use a moving average?

Comments:
Using days since the epoch makes it very difficult to match times to events. The moving average is offset from what was expected in the question, but as it was otherwise calculated correctly no points were deducted.

In [1]:

```
import re

def fileToList( fname, regexp=r'\W+' ):
    file = open(fname, 'rt')
    text = file.read()
    file.close()
    # split based on provided regular expression and remove empty strings
    # By default, matches words.
    return [x for x in re.split(regexp, text) if x]

bfile = "coindesk-bpi-close-data.csv"
bpi = fileToList( bfile, regexp=r'[,\r\n]+' )
# Print the head and tail.
print(bpi[:10:1])
print(bpi[-10::1])
```

```
['2018-09-01T22:59:59.404Z', '7198.0584362259', '2018-09-02T22:59:59.965Z', '7282.8585138311', '2018-0
9-03T22:59:59.783Z', '7260.2515925593', '2018-09-04T22:59:59.331Z', '7361.2718642376', '2018-09-05T22:
59:59.788Z', '6913.1150129363']
['2019-08-27T23:00:00.000Z', '10122.1965329581', '2019-08-28T22:59:59.000Z', '9743.1620510451', '2019-
08-29T22:59:59.000Z', '9487.9764335192', '2019-08-30T22:59:59.000Z', '9590.7363369227', '2019-08-31T22
:59:59.000Z', '9624.9839105321']
```

(Top)

YOUR ANSWER HERE

Comments:
No response.

In [2]:

(Top)

```
# Load numpy and matplotlib
import numpy as np
import matplotlib.pyplot as plt

# Remove every second element starting at element 0
bpi_clean = np.array(bpi[1::2], float)

# Use the convolve function to create a moving average function for the BPI data
def bpi_moving_average(N):
    return np.convolve(bpi_clean, np.ones((N,))/N, mode='valid')

# Plot the index with N = 5 and N = 10
moving_avg_list = [5, 10]
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
for i in moving_avg_list:
    plt.plot(bpi_moving_average(i), label = 'N = ' + str(i))
plt.plot(bpi_clean, 'c-', label = 'raw', alpha = 0.4)
fig.set_size_inches(15,6)
plt.legend()
plt.ylabel('Bitcoin Price Index (BPI)')
plt.xlabel('Day')
plt.show()
```

```
<Figure size 1500x600 with 1 Axes>
```

(Top)

The effect of the moving average filter helps smooth the raw data in order for the viewer to better observe the trends in the data. If minor, but large frequency variations in data are important, then this filter would not be advantageous.

(Top)

Question 2 (12 points):

Vectorized Race Simulation

1. Eight athletes are competing in a 1500 m race. Using `numpy` , write a vectorized race simulation according to the following criteria:

   - The granularity of the simulation is 1 s, i.e. each iteration in your simulation represents 1 second.
   - During each iteration, each athelete can randomly take 1, 2, 3, or 4 steps. Each step is 1 m long.
   - When the race is complete, return the winner and the winning time. There should not be any ties. If there is a tie, select a winner at random.

   Please pay attention to the following:

   - There should only be one loop in your simulation: the loop that advances the simulation by a second.
   - All other operations should be done using vectorized array operations and boolean indexing.
   - The following numpy functions will be helpful:
     - `numpy.random.randint` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randint.html)
     - `numpy.sum` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.sum.html)
     - `numpy.random.choice` (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.choice.html)

1. Run your simulation 10,000 times. For each run, record the winner and the winning time. Produce a bar chart showing the number of times each athelete won. Also display a bar chart showing the average winning time for each athlete.

Comments:
This code doesn't run, as submitted. The Python interpreter complains of missing variables, some of which are available under different names, while others are not. Once defined, variables and libraries persist in the running Python kernel even if their definitions are later removed. I recommend that once you are ready to submit, you stop and restart the running kernel, then run each cell from the beginning.
The race simulation does not return the length of time the race took, making it impossible to generate the average length of each race for each athlete. At the same time, a graph of average run times is being presented; something is missing here.

(Top)

YOUR ANSWER HERE

Comments:
No response.

In [3]:

(Top)

```
# Create tie breaker function
def tie_test(x):
    if x.size > 1:
        return np.random.choice(x)
    else:
        return x

# Create race simulation function
def race(num_racers, length):
    racers = np.zeros(num_racers)
    time = 0
    while np.amax(racers) <= length:
        racers += np.random.randint(1, 5, num_racers)
        time += 1
    winners_index = tie_test(np.where(racers >= length)[0])
    return winners_index

# Create repeat simulation function
def repeat_race(num_racers, length, times):
    results = []
    winners_count = np.zeros((num_racers, 2))
    for i in range(times):
        winners_count[race(num_racers, length)] += np.array([1, time])
        results.append("Racer #" + str(int(winners_index + 1)) + " won with a time of " + str(time) + "s")
        # I didn't want to return 'results' but have stored it in this function in case it was of interest
    return winners_count
```

In [4]:

(Top)

```
race_counter = repeat_race(8, 1500, 10000)
print(race_counter)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-9913b0ce5c97> in <module>
----> 1 race_counter = repeat_race(8, 1500, 10000)
      2 print(race_counter)

<ipython-input-3-c551565b1faa> in repeat_race(num_racers, length, times)
     21     winners_count = np.zeros((num_racers, 2))
     22     for i in range(times):
---> 23         winners_count[race(num_racers, length)] += np.array([1, time])
     24         results.append("Racer #" + str(int(winners_index + 1)) + " won with a time of " + str(
time) + "s")
     25         # I didn't want to return 'results' but have stored it in this function in case it was
of interest

NameError: name 'time' is not defined
```

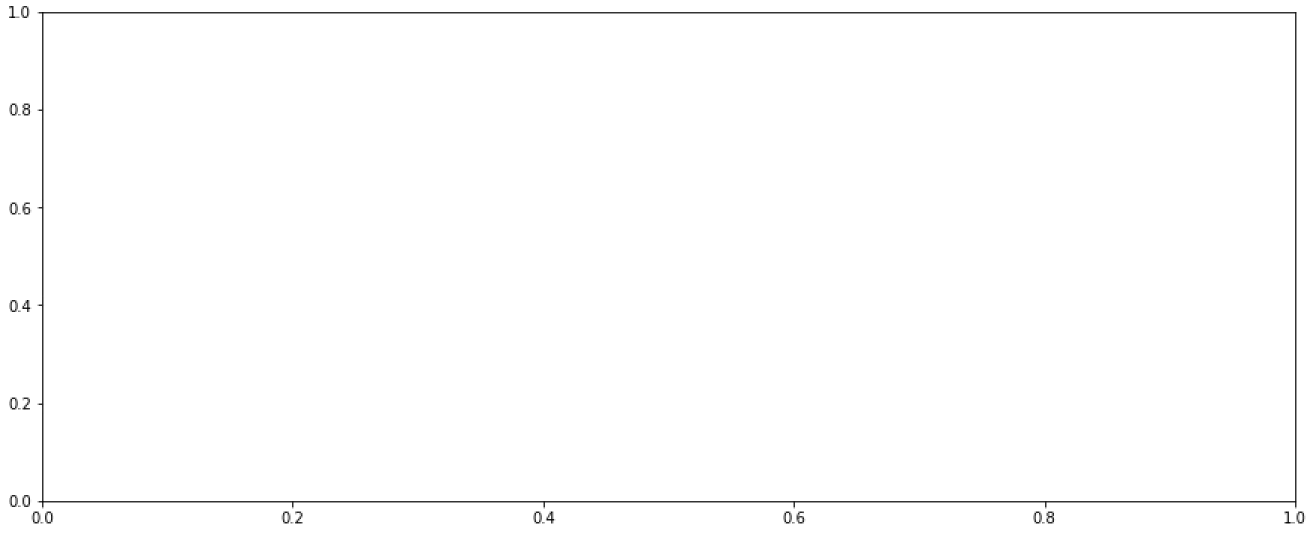
In [5]:

(Top)

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
fig.set_size_inches(15,6)
plt.bar(["1", "2", "3", "4", "5", "6", "7", "8"], race_counter[:, 1] / race_counter[:, 0])
plt.title('Distribution of Average Winning Time per Racer')
plt.ylabel('Average Winning Time (s)')
plt.xlabel('Racer #')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-998102bc8b40> in <module>
      2 ax = fig.add_subplot(1,1,1)
      3 fig.set_size_inches(15,6)
----> 4 plt.bar(["1", "2", "3", "4", "5", "6", "7", "8"], race_counter[:, 1] / race_counter[:, 0])
      5 plt.title('Distribution of Average Winning Time per Racer')
      6 plt.ylabel('Average Winning Time (s)')

NameError: name 'race_counter' is not defined
```



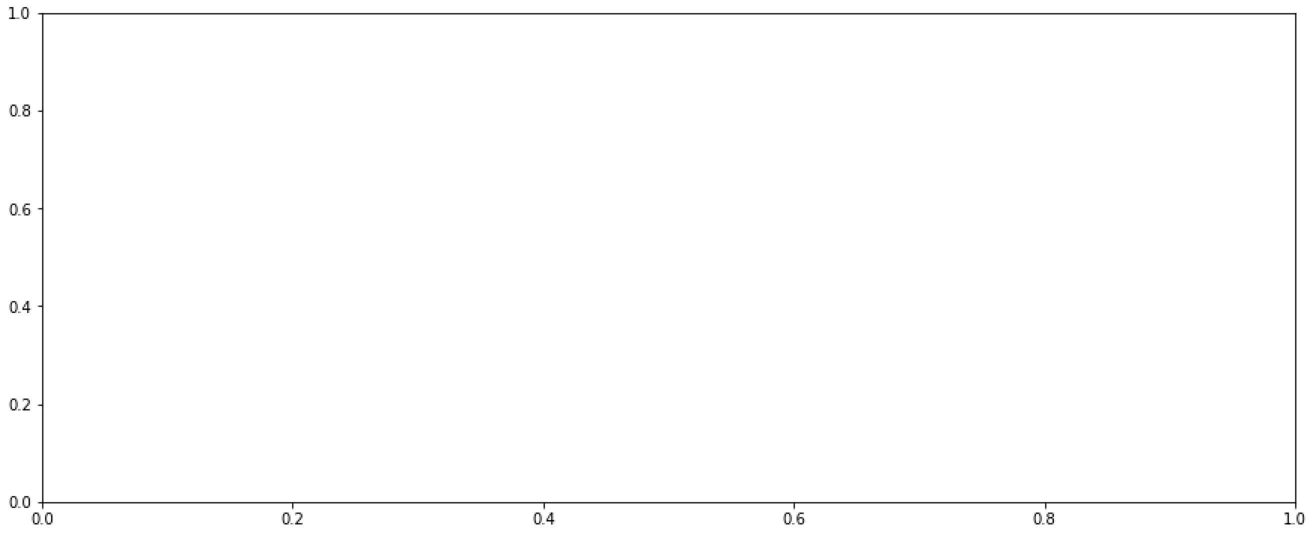
In [6]:

(Top)

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
fig.set_size_inches(15,6)
plt.bar(["1", "2", "3", "4", "5", "6", "7", "8"], race_counter[:, 0])
plt.title('Number of Race Wins per Racer')
plt.ylabel('Count')
plt.xlabel('Racer #')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-3e0b24c3c6fa> in <module>
      2 ax = fig.add_subplot(1,1,1)
      3 fig.set_size_inches(15,6)
----> 4 plt.bar(["1", "2", "3", "4", "5", "6", "7", "8"], race_counter[:, 0])
      5 plt.title('Number of Race Wins per Racer')
      6 plt.ylabel('Count')

NameError: name 'race_counter' is not defined
```



In [ ]:

(Top)