24. Data Visualization

Topics

How to define a useful class for manipulating sunrise/sunset data.

How to graphically display facts about that data using numpy and pyplot.
The Problem

For various cities around the world, we would like to examine the “Sun Up” time throughout the year.

How does it vary from day to day?

What are the monthly averages?

Sun Up Time = Sunset Time - Sunrise Time
How Does Sun-Up Depend on Latitude and Month?

Average Sun-Up (Hours):

<table>
<thead>
<tr>
<th>City</th>
<th>Latitude</th>
<th>June</th>
<th>September</th>
<th>December</th>
<th>March</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>51.50</td>
<td>16.55</td>
<td>12.64</td>
<td>7.93</td>
<td>11.89</td>
</tr>
<tr>
<td>Ithaca</td>
<td>42.43</td>
<td>15.24</td>
<td>12.47</td>
<td>9.13</td>
<td>11.95</td>
</tr>
<tr>
<td>New York</td>
<td>40.73</td>
<td>15.04</td>
<td>12.45</td>
<td>9.31</td>
<td>11.96</td>
</tr>
<tr>
<td>Cairo</td>
<td>30.05</td>
<td>14.05</td>
<td>12.34</td>
<td>10.25</td>
<td>11.99</td>
</tr>
<tr>
<td>Miami</td>
<td>25.78</td>
<td>13.72</td>
<td>12.29</td>
<td>10.56</td>
<td>12.02</td>
</tr>
<tr>
<td>Lagos</td>
<td>6.58</td>
<td>12.50</td>
<td>12.15</td>
<td>11.75</td>
<td>12.08</td>
</tr>
<tr>
<td>Johannesburg</td>
<td>-26.20</td>
<td>10.52</td>
<td>11.94</td>
<td>13.75</td>
<td>12.23</td>
</tr>
<tr>
<td>Sydney</td>
<td>-33.88</td>
<td>9.94</td>
<td>11.87</td>
<td>14.36</td>
<td>12.30</td>
</tr>
</tbody>
</table>
Visualization!
How Does Sun-Up Time Vary Day-to-Day?
How Does Sun-Up Time Vary Month-To-Month?
Recall the Motivating Problem

For various cities around the world, we would like to examine the “Sun Up” time throughout the year.

How does it vary from day to day?

What are the monthly averages?

Let’s define a class that makes this easy.
Our Plan

1. We define a class `Daylight` that facilitates data acquisition.

2. We introduce `numpy` arrays and show how to use the `pylab` for plotting.
The Class Daylight

5 Attributes

Name: name of the city [str]
Lat: latitude in degrees [float]
Long: longitude in degrees [float]
RiseTime: rise time in hours [length-365 numpy array]
SetTime: set time in hours [length-365 numpy array]
What the Constructor Does

It will have one argument: the name of a city as a string.

It will then read the .dat file associated with that city and proceed to set up the 5 attributes.
A Folder Called RiseSetData Has .dat Files for Each these Cities

<table>
<thead>
<tr>
<th>Anaheim</th>
<th>Anchorage</th>
<th>Arlington</th>
<th>Athens</th>
<th>Atlanta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baltimore</td>
<td>Bangkok</td>
<td>Beijing</td>
<td>Berlin</td>
<td>Bogota</td>
</tr>
<tr>
<td>Boston</td>
<td>Buenos Aires</td>
<td>Cairo</td>
<td>Chicago</td>
<td>Cincinnati</td>
</tr>
<tr>
<td>Cleveland</td>
<td>Denver</td>
<td>Detroit</td>
<td>Honolulu</td>
<td>Houston</td>
</tr>
<tr>
<td>Ithaca</td>
<td>Johannesburg</td>
<td>Kansas City</td>
<td>Lagos</td>
<td>London</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Mexico City</td>
<td>Miami</td>
<td>Milwaukie</td>
<td>Minneapolis</td>
</tr>
<tr>
<td>Moscow</td>
<td>New Delhi</td>
<td>New York</td>
<td>Oakland</td>
<td>Paris</td>
</tr>
<tr>
<td>Philadelphia</td>
<td>Phoenix</td>
<td>Pittsburgh</td>
<td>Rio De Janeiro</td>
<td>Rome</td>
</tr>
<tr>
<td>San Francisco</td>
<td>Seattle</td>
<td>Seoul</td>
<td>Sydney</td>
<td>Tampa</td>
</tr>
<tr>
<td>Teheran</td>
<td>Tokyo</td>
<td>Toronto</td>
<td>Washington</td>
<td>Wellington</td>
</tr>
</tbody>
</table>

For us, .dat files are the same as .txt files.

Downloaded from:  http://www.usno.navy.mil/
What do the lines in Ithaca.dat look like?
There Are 33 Lines

Ithaca
W07629N4226

1  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
2  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
3  R S R S R S R S R S R S R S R S R S R S R S R S R S R S

28  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
29  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
30  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
31  R S R S R S R S R S R S R S R S R S R S R S R S R S R S
The Data for a Particular City is Housed in a 33-line .dat file

Ithaca

W07629N4226


Line 2 encodes its longitude and latitude
Helper Function: LongLat

A latlong string has length 11

W08140N4129

def LongLat(s):
    Long = float(s[1:4])+float(s[4:6])/60
    if s[0] == 'E':
        Long = -Long
    Lat = float(s[7:9])+float(s[9:11])/60
    if s[6] == 'S':
        Lat = -Lat
    return (Lat,Long)
The Data for a Particular City is Housed in a 33-line .dat file

Ithaca
W07629N4226

| 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 28|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 29|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 30|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 31|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

The remaining lines house the rise-set data. Each R and S is a length-4 string: '0736'
Helper Function: ConvertTime

```python
def ConvertTime(s):
    x = float(s[:2])+float(s[2:])/60
    return x
```

In comes a length-4 string and back comes a float that encodes the time in hours

'0736'  ---->  7 + 36/60 hours  ----> 7.6
The Data for a Particular City is Housed in a 33-line .dat file

Ithaca
W07629N4226

Day-Number followed by 12 rise-set pairs, one pair for each month

Rise/Set data for April 3
The Class Daylight

Attributes:

City: name of the city [str]
Lat: latitude in degrees [float]
Long: longitude in degrees [float]
RiseTime: length-365 numpy array of sunrise times
SetTime: length-365 numpy array of sunset times
The Constructor

Sample Call

    C = Daylight(‘Ithaca’)

Reads the file Ithaca.dat into a list of 33 strings. Each string is deciphered.

Creates the Daylight object that house’s Ithaca’s name, latitude, longitude, the 365 sunrise times and the 365 sunset times.
We Need Some New Tools To Graphically Display the Data

```python
from numpy import *  
from pylab import *

We use numpy for arrays and pylab for plotting.
```
A Simple Plot

A = Daylight('Ithaca')
D = A.SunUp()
plot(D)
show()
A Simple Plot

```python
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)
show()

def SunUp(self):
    """returns a length-365 numpy array of sun-up times. """
    return self.SetTime - self.RiseTime

You can subtract one numpy array from another.
```
How about a title and a labeling of the y-axis?
A Simple Plot

```python
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)

titlestr = '%s  Lat = %6.2f  Long = %6.2f' % (A.City,A.Lat,A.Long)
title(titlestr,fontsize=16)
ylabel('Hours of Sunlight',fontsize=16)
show()
```
Modify the x range and the y range
A Simple Plot

```python
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)

titlestr = '%s  Lat = %6.2f  Long = %6.2f' % (A.City, A.Lat, A.Long)
title(titlestr, fontsize=16)
ylabel('Hours of Sunlight', fontsize=16)
xlim(0, 364)
ylim(5, 20)
show()
```
Label the x-axis with month names
Ithaca  Lat = 42.43  Long = 76.48

Hours of Sunlight

Add a Grid
def MonthAves(self):
    x = zeros((12,1))
    D = self.SunUp()
    start  = [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334]
    finish = [30, 58, 89, 119, 150, 180, 211, 242, 272, 303, 333,364]
    for k in range(12):
        z = D[start[k]:finish[k]]
        x[k] = sum(z)/len(z)
    return x
A = Daylight('Ithaca')
M = A.MonthAves()

bar(range(12), M, facecolor='magenta')
xlim(-.2,12)
ylabel('Average Hours of Sunlight')
title(A.City, fontsize=16)
show()
More on Numpy Arrays
1-dimensional Array Basics

```python
>>> from numpy import *
>>> x = array([1,2,3])
>>> x
array([1, 2, 3])
```

X is a 1d array. (2d arrays soon!)

It has 3 entries

The entries are floats.
1-dimensional Array Basics

```python
>>> y = array([1,2,3], dtype='int')
>>> z = y[2]/y[1]
>>> z
1
```

This is how you create an array of ints.
1-dimensional Array Basics

```python
>>> a = array([10, 20, 30])
>>> b = array([5, 4, 15])
>>> a+b
array([15, 24, 45])
>>> a-b
array([5, 16, 15])
>>> a/b
array([2, 5, 2])
>>> a*b
array([50, 80, 450])
```

You can add, subtract, divide, and multiply arrays.
1-dimensional Array Basics

```python
>>> f = array([10,20])
>>> g = array([1,2,3])
>>> f+g
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (2,) (3,)
```

But they better be the same size!
1-dimensional Array Basics

```python
>>> u = [1,2,3]
>>> type(u)
<type 'list'>
>>> v = array([10,20,30])
>>> type(v)
<type 'numpy.ndarray'>
>>> z = u+v
>>> z
array([11, 22, 33])
>>> type(z)
<type 'numpy.ndarray'>
```

You can mix "regular" lists of numbers with numpy arrays
1-dimensional Array Basics

```python
>>> x = array([-10.3, 12.6, -89.7])
>>> y = abs(x)
>>> y
array([ 10.3,  12.6,  89.7])
```

You can apply a function to an array if it is ok to apply the function to each entry in the array.
The numpy `linspace` function

\[ x = \text{linspace}(1,3,5) \]

\[ x : 1.0 \ 1.5 \ 2.0 \ 2.5 \ 3.0 \]

`linspace(a,b,n)` is a length `-n` list of values that are equally spaced from \( x = a \) to \( x = b \).
Assume:

```python
from numpy import *
from pylab import *
```
Displaying an Array

Assume:

from numpy import *  
from pylab import *  

U = Daylight('Ithaca')
D = U.SunUP()
plot(D)
<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0</td>
</tr>
<tr>
<td>1.57</td>
<td>1.0</td>
</tr>
<tr>
<td>3.14</td>
<td>0.0</td>
</tr>
<tr>
<td>4.71</td>
<td>-1.0</td>
</tr>
<tr>
<td>6.28</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Plot based on 5 points

Table $\rightarrow$ Plot
# Table → Plot

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.784</td>
<td>0.707</td>
</tr>
<tr>
<td>1.571</td>
<td>1.000</td>
</tr>
<tr>
<td>2.357</td>
<td>0.707</td>
</tr>
<tr>
<td>3.142</td>
<td>0.000</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.707</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.000</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.707</td>
</tr>
<tr>
<td>6.283</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Plot based on 9 points
Plot based on 200 points—looks smooth
### Generating Tables and Plots

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.784</td>
<td>0.707</td>
</tr>
<tr>
<td>1.571</td>
<td>1.000</td>
</tr>
<tr>
<td>2.357</td>
<td>0.707</td>
</tr>
<tr>
<td>3.142</td>
<td>0.000</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.707</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.000</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.707</td>
</tr>
<tr>
<td>6.283</td>
<td>0.000</td>
</tr>
</tbody>
</table>

```python
x = linspace(0, 2*pi, 9)
y = sin(x)
plot(x, y)
show()
```
plot(x,y)

x, y 1-dim arrays of numbers
That have the same length

plot(x,y) "connects the dots":

(x[0], y[0]), ..., (x[n-1], y[n-1])
for k in range(6,20):
    # Draw horizontal line from (0,k) to (365,k)
    plot(array([0,365]),array([[k,k]]),
         color='red',linestyle=':')
for k in range(6,20):
    # Draw horizontal line from (0,k) to (365,k)
    plot(array([0,365]),array([[k,k]]),
         color='red',linestyle=':')

Connect two dots
A Note on subplot

```
subplot(2,1,1)
<code>
subplot(2,1,2)
<code>
Show()
```

When you want more than one plot in the window.
A Note on subplot

```python
subplot(2,2,1)
<code>
subplot(2,2,2)
<code>
subplot(2,2,3)
<code>
subplot(2,2,4)
<code>
Show()
```