

# Data Visualization

BBM 101 - Introduction to Programming I

Hacettepe University

Fall 2016

Fuat Akal, Aykut Erdem, Erkut Erdem

# Today

- **A Little Bit on Data Science**
  - What is Data Science?
  - Why learn Data Science?
  - How do we learn Data Science?
- **Plotting with Matplotlib**
  - How to read data from a file?
  - How to work with that data?
  - How to graphically display facts about that data using **numpy** and **pyp1ot**?

# Today

- **A Little Bit on Data Science**
  - What is Data Science?
  - Why learn Data Science?
  - How do we learn Data Science?
  
- **Plotting with Matplotlib**
  - How to read data from a file?
  - How to work with that data?
  - How to graphically display facts about that data using `numpy` and `pyplot`?

# 20<sup>th</sup> Century Innovation

Engineering and Computer Science played key role

- Cars
- Airplanes
- Power grid
- Television
- Air conditioning and central heating
- Nuclear power
- Digital computers
- The internet

For more:

<http://camdp.com/blogs/21st-century-problems>

# But how about these 20th Century questions?

- Does fertilizer increase crop yields?
- Does Streptomycin cure Tuberculosis?
- Does smoking cause lung-cancer?

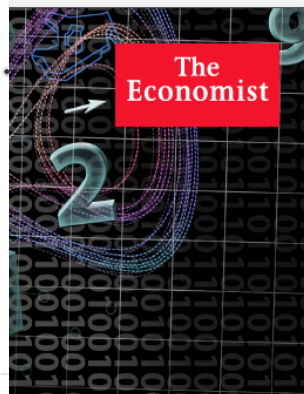
# What is the difference

- Deterministic versus random
- Deductive versus empirical
- Solutions deduced mostly from theory versus solutions deduced from mostly from **data**

# Data

- Does fertilizer increase crop yields? Answer: Collect and analyze agricultural experimental **data**
- Does Streptomycin cure Tuberculosis? Collect and analyze randomized trials **data**
- Does smoking cause lung-cancer? Collect and analyze observational studies **data**
- Analyzing these was the job of: boring ol' **statisticians**

# 21<sup>st</sup> Century





# 21<sup>st</sup> Century

“I keep saying the sexy job in the next ten years will be statisticians. People think I'm joking, but who would've guessed that computer engineers would've been the sexy job of the 1990s?”

- Hal Varian, Google's Chief Economist

# Hal Varian Explains...

“The ability to take **data** – to be able to **understand** it, to **process** it, to **extract value** from it, to **visualize** it, to **communicate** it's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids. Because now we really do have essentially free and ubiquitous data.”

– Hal Varian

# Data Science Success Stories

BRAD PITT



# MONEYBALL

JONAH HILL PHILIP SEYMOUR HOFFMAN

BASED ON A TRUE STORY

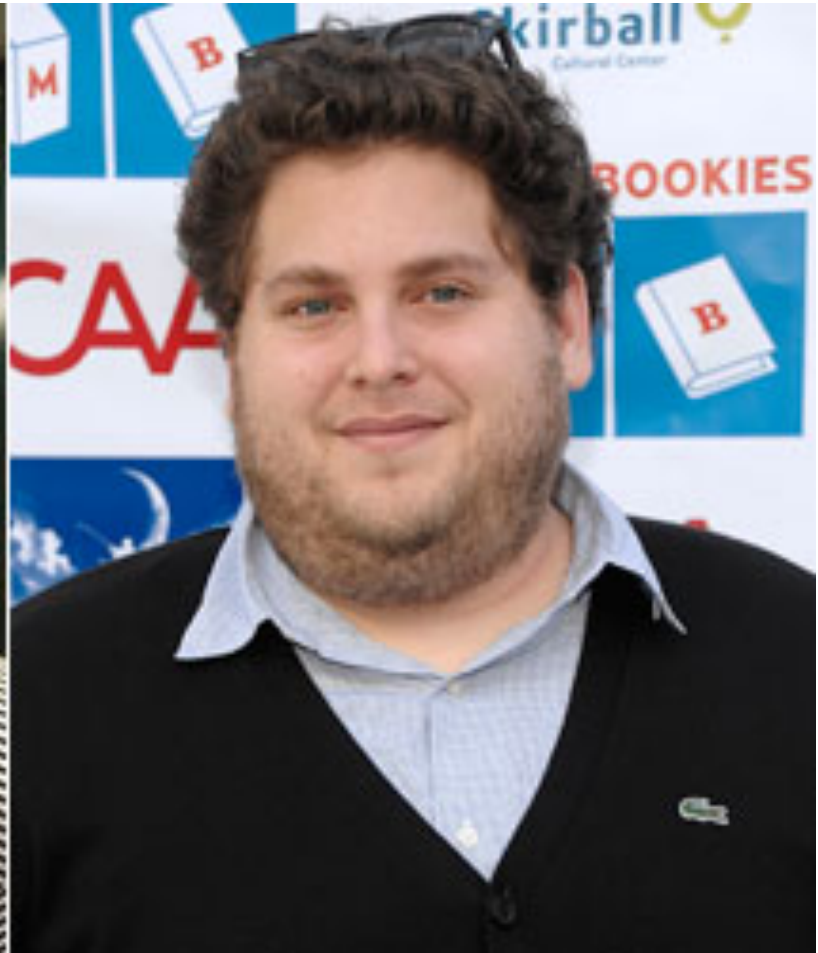
COLUMBIA PICTURES PRESENTS A SCOTT ROBIN / WICKED DE LUCA / BACKLASH / ROBBYTT PRODUCTION A FILM BY BENNETT MILLER  
"MONEYBALL" CASTING BY DANIELA JACOBSON AND JESSICA WILLY PASTER AND  
JONAH SCOTT ROBIN ANDREW BARACK ZORNY KANDEL MARK BACHIN \*\*\*A WICKED DE LUCA / BACKLASH / ROBBYTT PRODUCTION  
SONY PICTURES CLASSICS PRESENTS A WICKED DE LUCA / BACKLASH / ROBBYTT PRODUCTION A FILM BY BENNETT MILLER  
COMING SOON

# The Data Scientist

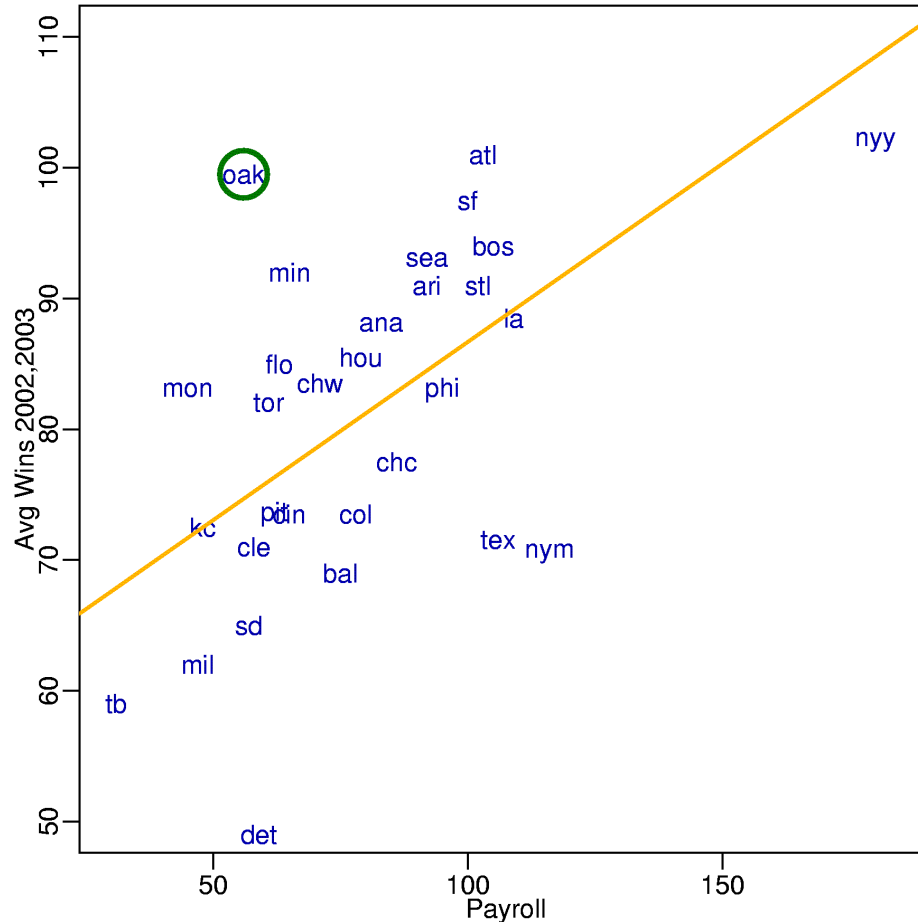
Actual



Hollywood



# Money Ball



Start around 2001, the Oakland A's picked players that scouts thought no good but data said otherwise

# Netflix Challenge

**The New York Times**  
Wednesday, October 14, 2009

## Technology

WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION

Search Technology  Go

Inside Technology  
[Internet](#) [Start-Ups](#) [Business Computing](#) [Compu](#)


### Bits

Business ■ Innovation ■ Technology ■ Society

September 21, 2009, 10:15 AM

## Netflix Awards \$1 Million Prize and Starts a New Contest

By STEVE LOHR



Jason Kempin/Getty Images

Netflix prize winners, from left: Yehuda Koren, Martin Chabbert, Martin Plotte, Michael Jahrer, Andreas Toscher, Chris Volinsky and Robert Bell.

In Sept 2009 a team lead by Chris Volinsky from Statistics Research AT&T Research was announced as winner!

# Ad-targeting

Ads ⓘ

Yacht Inbox x



1:19 PM (1 minute ago) ☆



Suit yourself. I'll send you pictures from my yacht.

## **Making Sense of Big Data**

A Big Data Guide for Small & Medium Businesses. Get the Free eMagazine!

[www.tableausoftware.com/big-data](http://www.tableausoftware.com/big-data)

## **Dell™ Computer Outlet**

Shop Dell™ Outlet For Discounted Computer Refurbs, w/ Intel® Core™

[www.Dell.com/Outlet](http://www.Dell.com/Outlet)

## **Luxury BVI Cruise**

7 Night Small Ship BVI Cruise from \$2,595. Book Now & Save 50%

[pgcruises.com/BVI-Cruise](http://pgcruises.com/BVI-Cruise)

## **BVI Yacht Charter**

Yacht Charter in the BVI bareboat and with great crews.

[www.ViSailing.com](http://www.ViSailing.com)



# How do we do Data Science?

- **Science:** determining what questions can be answered with data and what are the best datasets for answering them
- **Computer programming:** using computers to analyze data
- **Data wrangling:** getting data into analyzable form on our computers
- **Statistics:** separating signal from noise
- **Machine learning:** making predictions from data
- **Communication:** sharing findings through visualization, stories and interpretable summaries

# Today

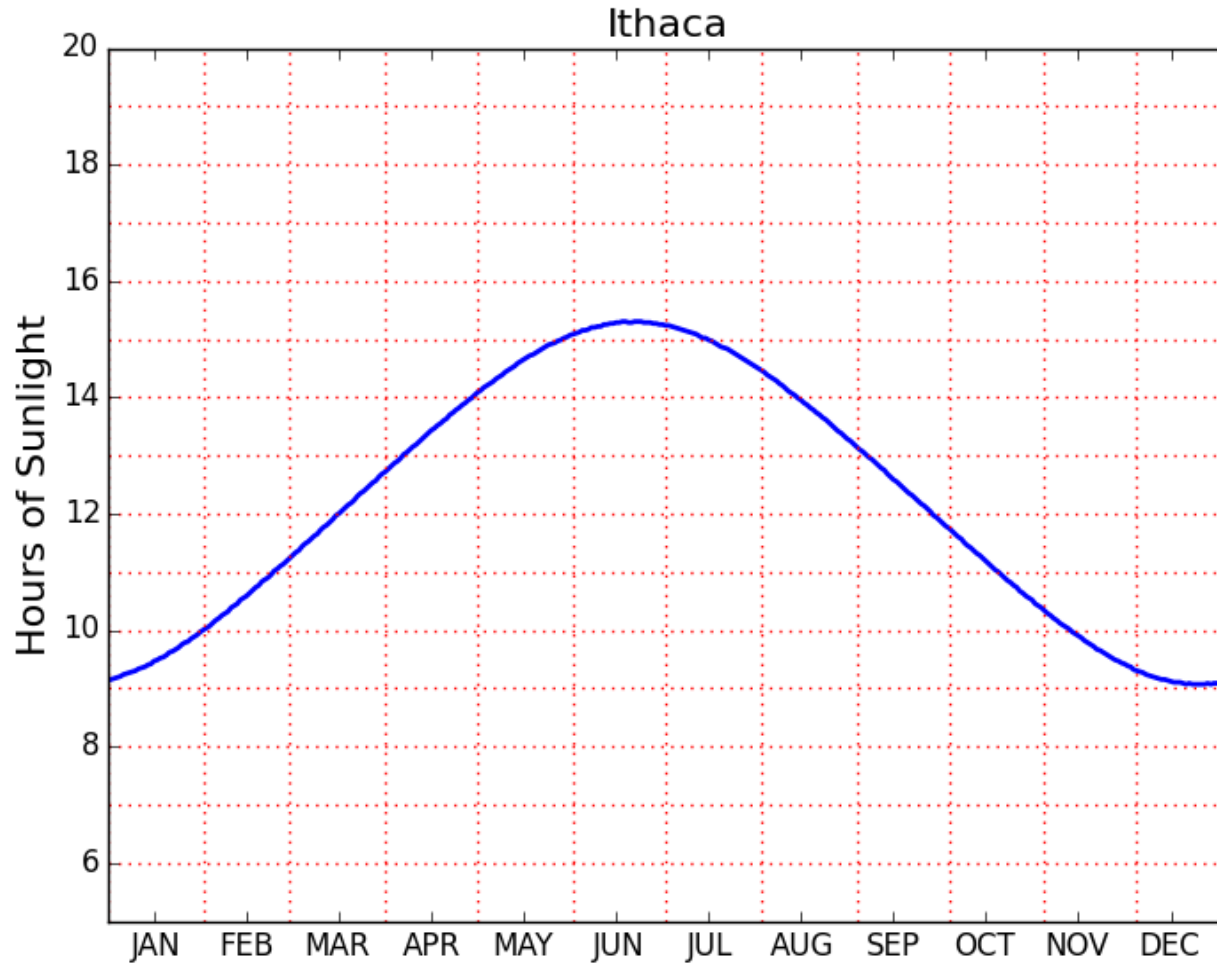
- **A Little Bit on Data Science**
  - What is Data Science?
  - Why learn Data Science?
  - How do we learn Data Science?
  
- **Visualization**
  - How to read data from a file?
  - How to work with that data?
  - How to graphically display facts about that data using **numpy** and **pyp1ot**?

# The Problem

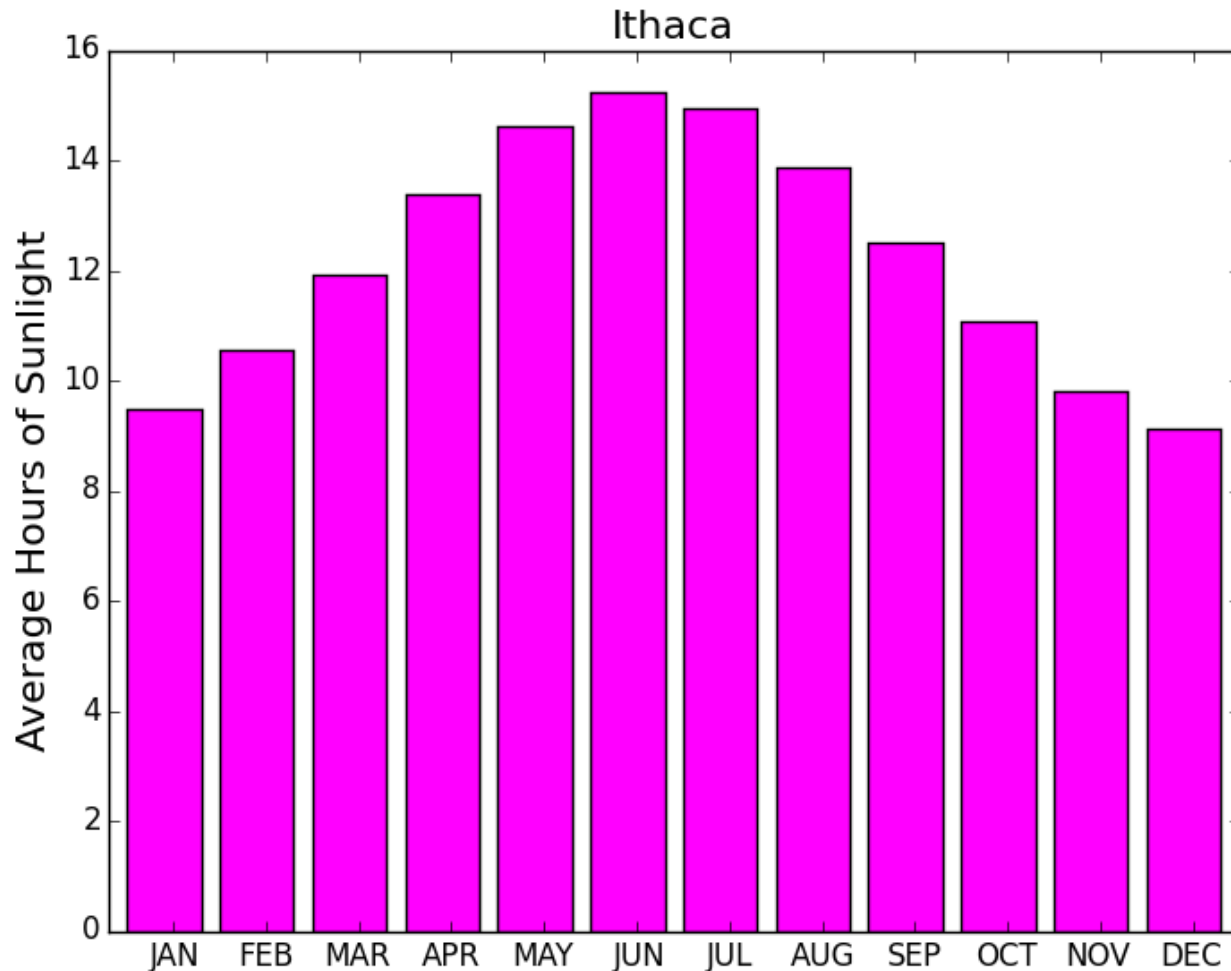
- For various cities around the world, we would like to examine the “Sun Up” time throughout the year.
- How does it vary from day to day?
- What are the monthly averages?

$$\text{Sun Up Time} = \text{Sunset Time} - \text{Sunrise Time}$$

# How Does Sun-Up Time Vary Day-to-Day?



# How Does Sun-Up Time Vary Month-to-Month?



# The Task Before Us...

1. Find a website where the data can be found.
2. Get that data into a file on our computer.
3. Understand how the data is laid out in the file.
4. Write python code that gets that data (or some aspect of it) into your Python environment.

# Where Do We Get the Data?

- Lots of choices. Google “Sunset Sunrise times”
- We will use the U.S. Naval Observatory data service:
- Visit:

<http://www.usno.navy.mil/>

# From the Website...

## Astronomical Applications

### [Data Services](#)

Sun and Moon rise and set times, Moon phases, eclipses, seasons, positions of solar system objects, and other data

[Complete Sun and Moon Data for One Day](#)

[Sun or Moon Rise/Set Table for One Year](#)

[Phases of the Moon](#)  
[more...](#)



# We Downloaded Rise/Set Data For a Number of Cities

Anaheim	Anchorage	Arlington	Athens	Atlanta
Baltimore	Bangkok	Beijing	Berlin	Bogata
Boston	BuenosAires	Cairo	Chicago	Cincinnati
Cleveland	Denver	Detroit	Honolulu	Houston
Ithaca	Johannesburg	KansasCity	Lagos	London
LosAngeles	MexicoCity	Miami	Milwaukee	Minneapolis
Moscow	NewDelhi	NewYork	Oakland	Paris
Philadelphia	Phoenix	Pittsburgh	RiodeJaneiro	Rome
SanFrancisco	Seattle	Seoul	Sydney	Tampa
Teheran	Tokyo	Toronto	Washington	Wellington

# One .dat File Per City

RiseSetData

Anaheim.dat

Anchorage.dat

Arlington.dat

:

Toronto.dat

Washington.dat

Wellington.dat

We put all these files in a directory called RiseSetData

.dat and .txt files are common ways to house simple data. Don't worry about the difference.

# .txt and .dat Files have Lines

```
MyFile.dat
```

```
abcd
```

```
123 abc d fdd
```

```
xyz
```

```
3.14159      2.12345
```

There is an easy way to read the data in such a file line-by-line

# Read and Print the Data in Ithaca.dat

FileIO.py

```
FileName = 'RiseSetData/Ithaca.dat'  
f = file(FileName, 'r')  
for s in f:  
    print s  
f.close()
```

**RiseSetData** and **FileIO.py** must be in the same folder.

# Ithaca.dat

- 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
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

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
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
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
31 R S  R S  R S  R S  R S  R S  R S  R S
```

The provider of the file typically tells  
you how the data is structured

# From the Naval Observatory Website

- The first line names the city and the second line encodes its latitude and longitude, e.g.,

**Ithaca**

**W07629N4226**

and ...

# From the Naval Observatory Website

- The rise and set times are then specified day-by-day with the data for each month housed in a pair of columns.
- In particular, columns  $2k$  and  $2k+1$  have the rise and set times for month  $k$  (Jan=1, Feb = 2, Mar = 3, etc.)
- Column 1 specifies day-of-the-month, 1 through 31. Blanks are used for nonexistent dates (e.g., April 31).







# Helper Function: LongLat

- A latlong string has length 11, e.g. **W08140N4129**

```
def LongLat(s):
```

```
    """ Returns a tuple (Long,Lat) of floats that are the
    equivalent (in degrees) of the longitude and latitude
    encoded by s.
```

```
    PredC: s an 11-character string of the form 'cdddmmCDDMM'
    where cdddmm specifies longitude in degrees and minutes with
    c = 'W' or 'E' and CDDMM species latitude in degrees and
    minutes with C = 'N' or '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   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
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

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
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
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
31  R S   R S   R S   R S   R S   R S   R S   R S
```

The remaining lines house the rise-set data.

Each R and S is a length-4 string: '0736'

# Helper Function: ConvertTime

```
def ConvertTime(s):
```

```
    """ Returns a float that is the equivalent (in
    hours) of the time encoded by s.
```

```
    '2145' means 9:45 pm.
```

```
    PredC: s a 4-character string of the form hhmm
            that specifies time.
```

```
    """
```

```
    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

1	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
3	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
29	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
31	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S

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

# The Data for a Particular City is Housed in a 33-line .dat file

Ithaca

W07629N4226

1	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
3	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
29	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
31	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S

Day -Number followed by 11 rise-set pairs, one pair for each month except February

# The Data for a Particular City is Housed in a 33-line .dat file

Ithaca

W07629N4226

1	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
3	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
29	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
31	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S	R	S

Day -Number followed by 7 rise-set pairs, one pair for each 31-day 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?



# Daylight

```
def SunUp(CityName):
    FileName = 'RiseSetData/'+CityName+'.dat'
    f = file(FileName, 'r');
    lineNumber = 0
    for s in f:
        parts = s.split()
        lineNumber+=1
        if lineNumber == 1:
            City = parts[0]
        elif lineNumber == 2:
            Lat, Long = LatLong(parts[0])
        else:
            Code that builds the RiseTime and SetTime arrays
    f.close()
    return (City, Lat, Long, SetTime - RiseTime)
```

Recall how split works...

```
s = '1 0535 0816 0542 0713'
x = s.split()
print x
['1', '0535', '0816', '0542', '0713']
```

# Building RiseTime and SetTime arrays

...

```
# Remaining lines have rise/set pairs
```

```
day = int(parts[0])
```

```
# Get all the rise and set times
```

```
RiseTimeList = ConvertTime(parts[1:len(parts):2])
```

```
SetTimeList = ConvertTime(parts[2:len(parts):2])
```

```
p = len(RiseTimeList)
```

```
for k in range(p):
```

```
    if day<=28:
```

```
        # All months have at least 28 days
```

```
        starts = [0,31,59,90,120,151,181,212,243,273,304,334]
```

```
        dayIndex = day + starts[k] - 1
```

```
    elif day==29 or day==30:
```

```
        # All months except February have a day 29 and a day 30
```

```
        starts = [0, 59,90,120,151,181,212,243,273,304,334]
```

```
        dayIndex = day + starts[k] - 1
```

```
    else:
```

```
        # Only January, March, May, July, August, October, and December have
```

```
        # a day 31.
```

```
        starts = [0,59,120,181,212,273,334]
```

```
        dayIndex = day + starts[k] - 1
```

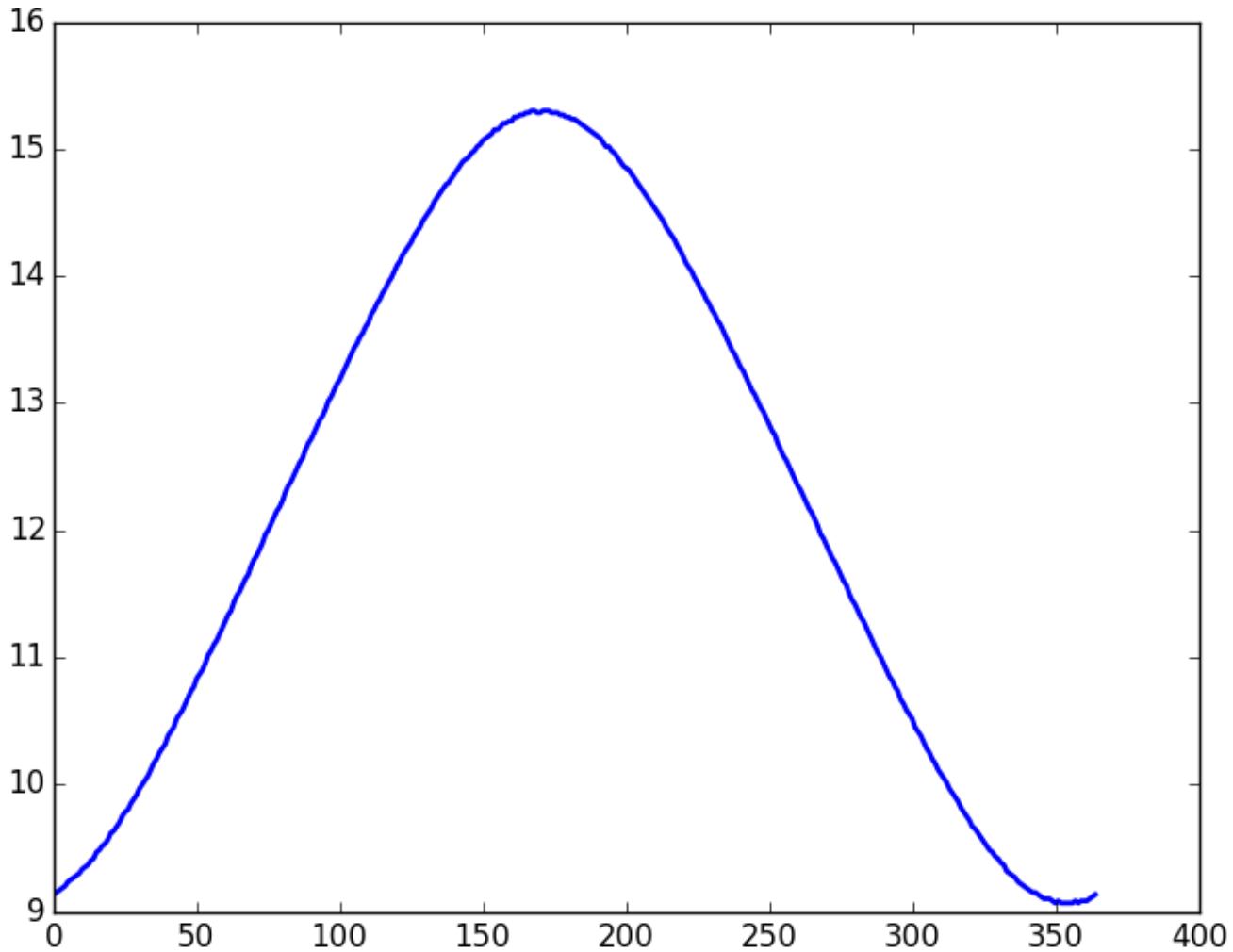
```
RiseTime[dayIndex] = RiseTimeList[k]
```

```
SetTime[dayIndex] = SetTimeList[k]
```

# A Simple Plot

```
from pylab import *  
  
# Plot a 1-dim numpy array  
City, Lat, Long, D = SunUp('Ithaca')  
plot(D)  
  
show()
```

This is how you display the values in a numpy array like D.



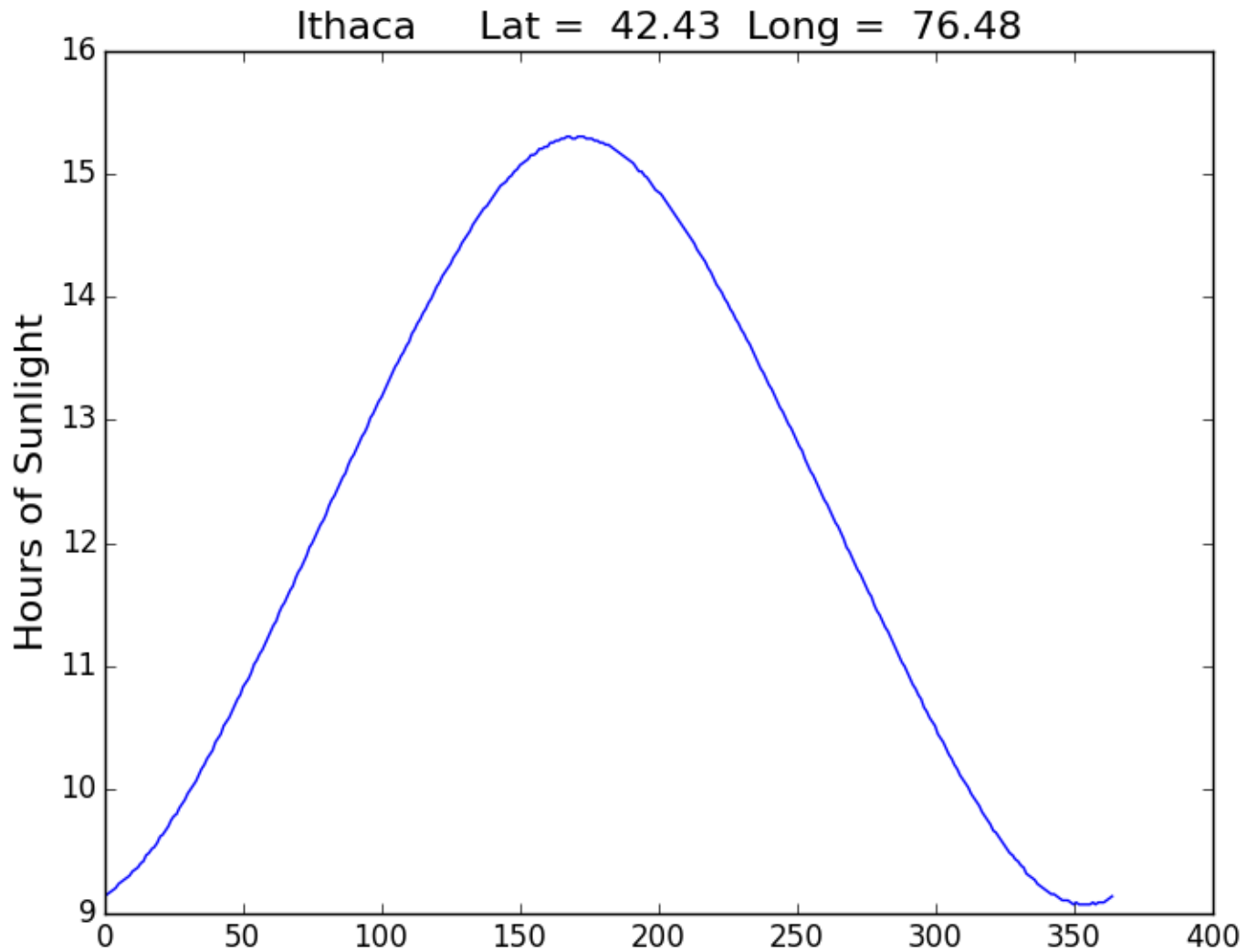
How about a title and a labeling of the y-axis?

# A Simple Plot

```
# Plot a 1-dim numpy array
City, Lat, Long, D = SunUp('Ithaca')
plot(D)

# The title
titlestr = '%s Lat = %6.2f Long = %6.2f' % (City, Lat, Long)
title(titlestr, fontsize=16)
# Label the y-axis
ylabel('Hours of Sunlight', fontsize=16)

show()
```



Modify the x range and the y range

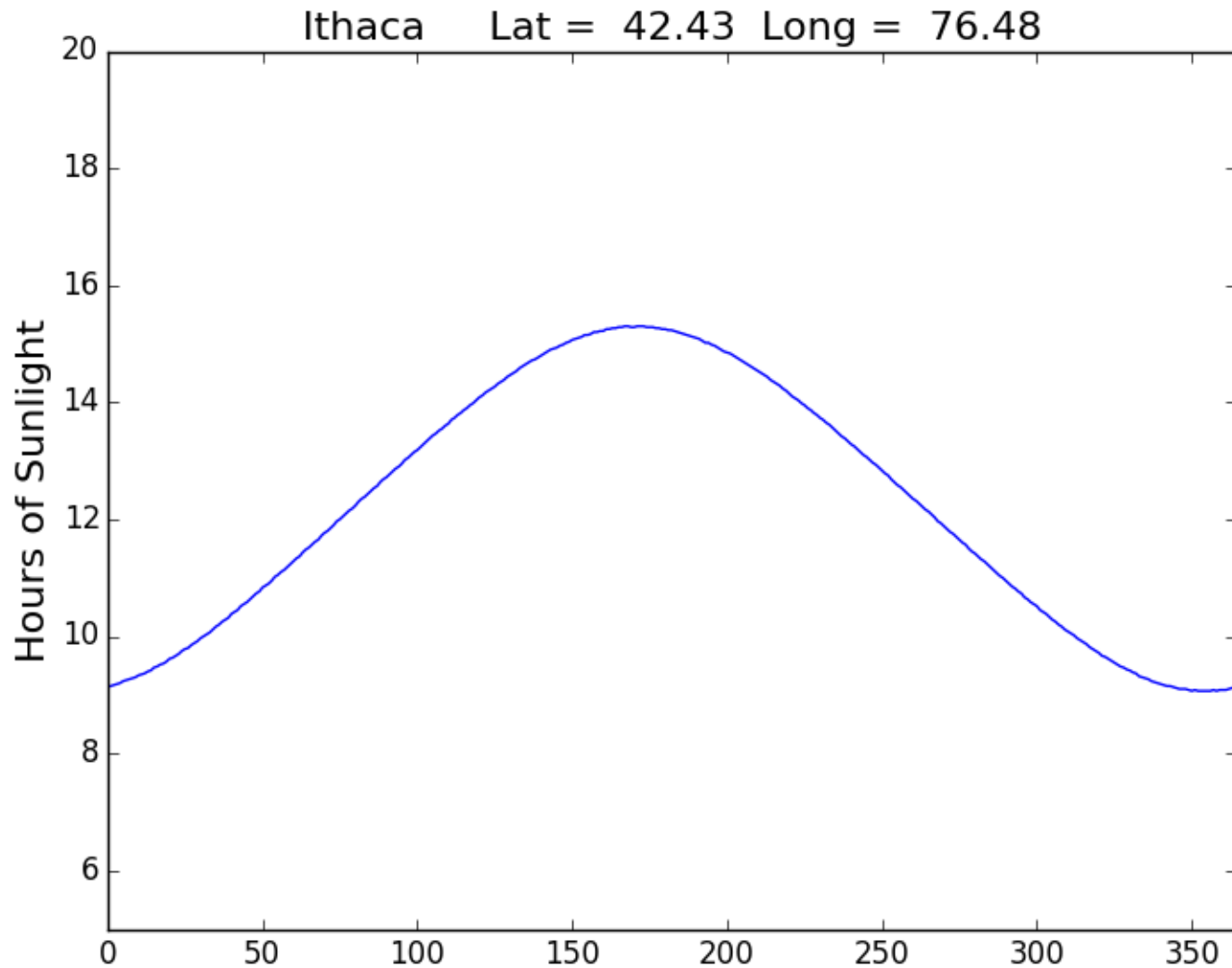
# A Simple Plot

```
# Plot a 1-dim numpy array
City, Lat, Long, D = SunUp('Ithaca')
plot(D)

# The title
titlestr = '%s Lat = %6.2f Long = %6.2f' % (City, Lat, Long)
title(titlestr, fontsize=16)
# Label the y-axis
ylabel('Hours of Sunlight', fontsize=16)

# set the range of x and the range of y
xlim(0, 364)
ylim(5, 20)

show()
```



Label the x-axis with month names



# A Simple Plot

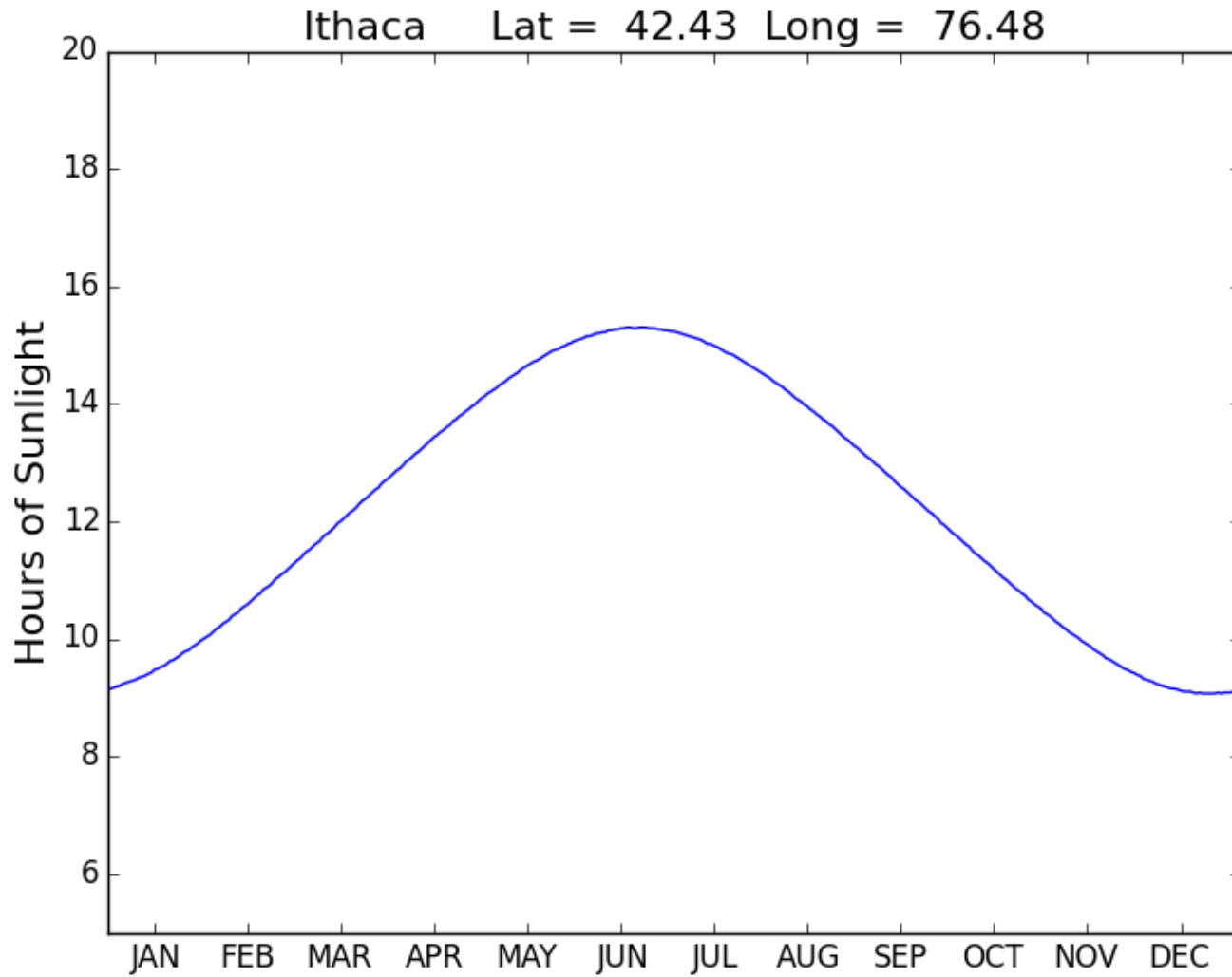
```
# Plot a 1-dim numpy array
City, Lat, Long, D = SunUp('Ithaca')
plot(D)

# The title
titlestr = '%s Lat = %6.2f Long = %6.2f' % (City,Lat,Long)
title(titlestr,fontsize=16)
# Label the y-axis
ylabel('Hours of Sunlight',fontsize=16)

# set the range of x and the range of y
xlim(0,364)
ylim(5,20)

# Position ticks along the x-axis and label them
c = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']
t = [15,45,75,105,135,165,195,225,255,285,315,345]
xticks( t,c)

show()
```



Add a Grid

# A Simple Plot

```
# Plot a 1-dim numpy array
City, Lat, Long, D = SunUp('Ithaca')
plot(D)

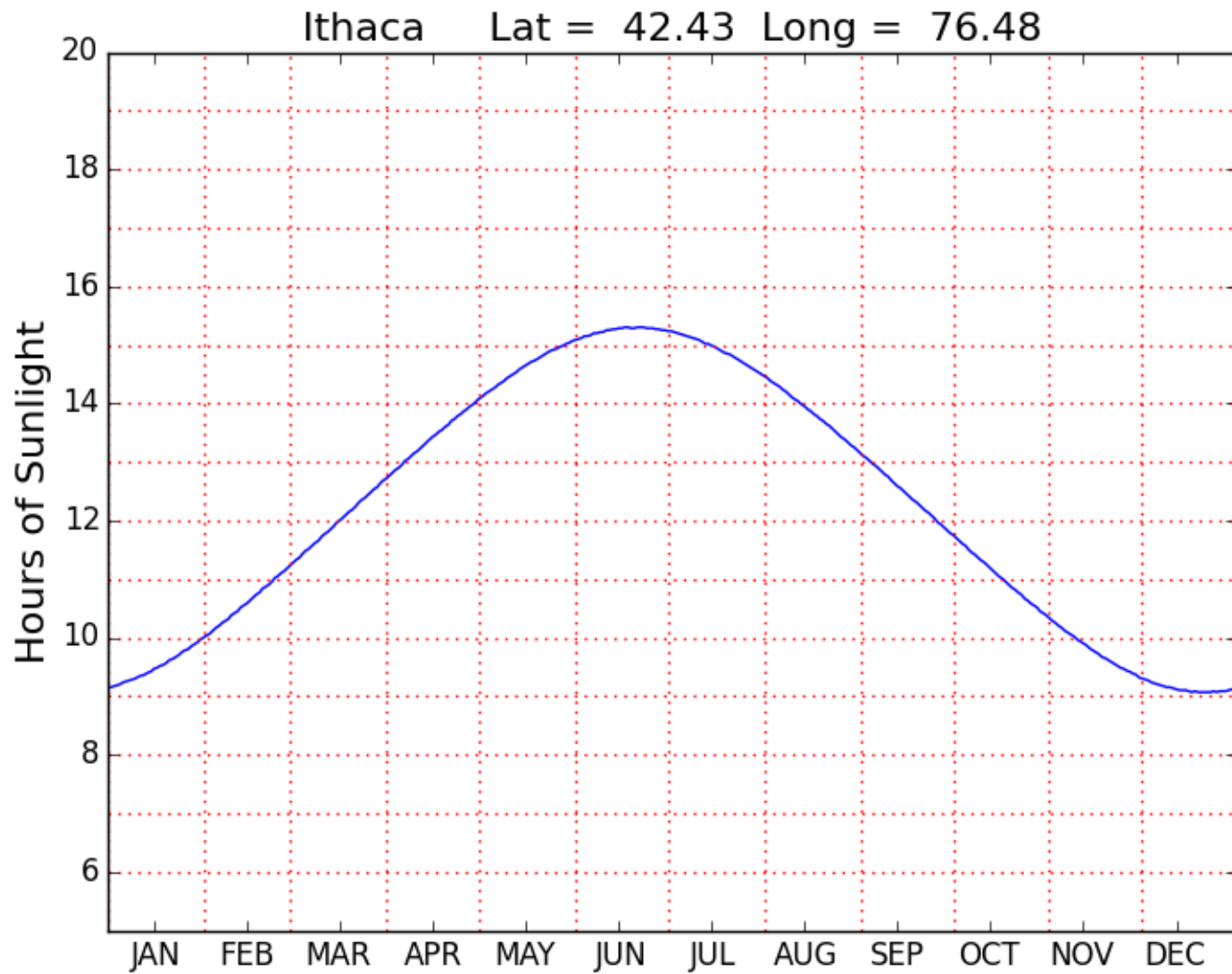
# The title
titlestr = '%s Lat = %6.2f Long = %6.2f' % (City,Lat,Long)
title(titlestr,fontsize=16)
# Label the y-axis
ylabel('Hours of Sunlight',fontsize=16)

# set the range of x and the range of y
xlim(0,364)
ylim(5,20)

# Position ticks along the x-axis and label them
c = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']
t = [15,45,75,105,135,165,195,225,255,285,315,345]
xticks( t,c)

# Draw a grid
for k in range(6,20):
    # Draw horizontal line from (0,k) to (65,k)
    plot(array([0,365]),array([k,k]),color='red',linestyle=':')
for k in [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334]:
    # Draw vertical line from (k,5) to (k,20)
    plot(array([k,k]),array([5,20]),color='red',linestyle=':')

show()
```



# Monthly Averages

```
def MonthAverages(CityName):  
    x = zeros((12,1))  
    City, Lat, Long, D = SunUp(CityName)  
    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 Bar Plot

```
M = MonthAverages('Ithaca')  
  
bar(range(12), M, facecolor='magenta')  
xlim(-.2, 12)  
ylabel('Average Hours of Sunlight')  
title(A.City, fontsize=16)  
show()
```

