In [2]:

```
%matplotlib inline
```

```
/opt/conda/envs/python2/lib/python2.7/site-packages/matplotlib/font_manager.py:
273: UserWarning: Matplotlib is building the font cache using fc-list. This may
 take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This may
 take a moment.')
```

We start by initializing the Earth Engine API.

In [3]:

```
import ee
ee.Initialize()
```

Then we import in other Python packages we need.

In [4]:

```
import datetime
from matplotlib import dates
import matplotlib.dates as mdates
from pylab import *
```

Next we define the Landsat bands that we would like to plot, along with the starting and ending times.

In [5]:

```
xBand = 'time'
yBandList = [
        'B1',
        u'B2',
        u'B3',
        u'B4',
        u'B5',
        u'B6_VCID_1',
        u'B6_VCID_2',
        u'B7',
        u'B8',
    ]
startTime = datetime.datetime(2000, 1, 1)
endTime = datetime.datetime(2004, 1, 1)
```

Next we contruct a filtered ImageCollection for the date range, and extract band information for a specified point location.

In [6]:

```
collection = ee.ImageCollection('LE7_L1T').filterDate(startTime, endTime)
point = {'type':'Point', 'coordinates':[ -116.88629,36.56122]};  # death valley (
info = collection.getRegion(point,500).getInfo()
```

We separate the information returned into column headers and data.

In [7]:

```
# extract the header column names
header = info[0]
# create a Numpy array of the data
data = array(info[1:])
```

Next we extract time information and convert it to at Python datatime data type.

In [8]:

```
# extract the time information
iTime = header.index('time')
# convert to Python datetime objects
time = [datetime.datetime.fromtimestamp(i/1000) for i in (data[0:,iTime].astype(i
```

Extract the data columns what we want to display on the plot.

In [9]:

```
iBands = [header.index(b) for b in yBandList]
yData = data[0:,iBands].astype(np.float)
```

Calculate NDVI

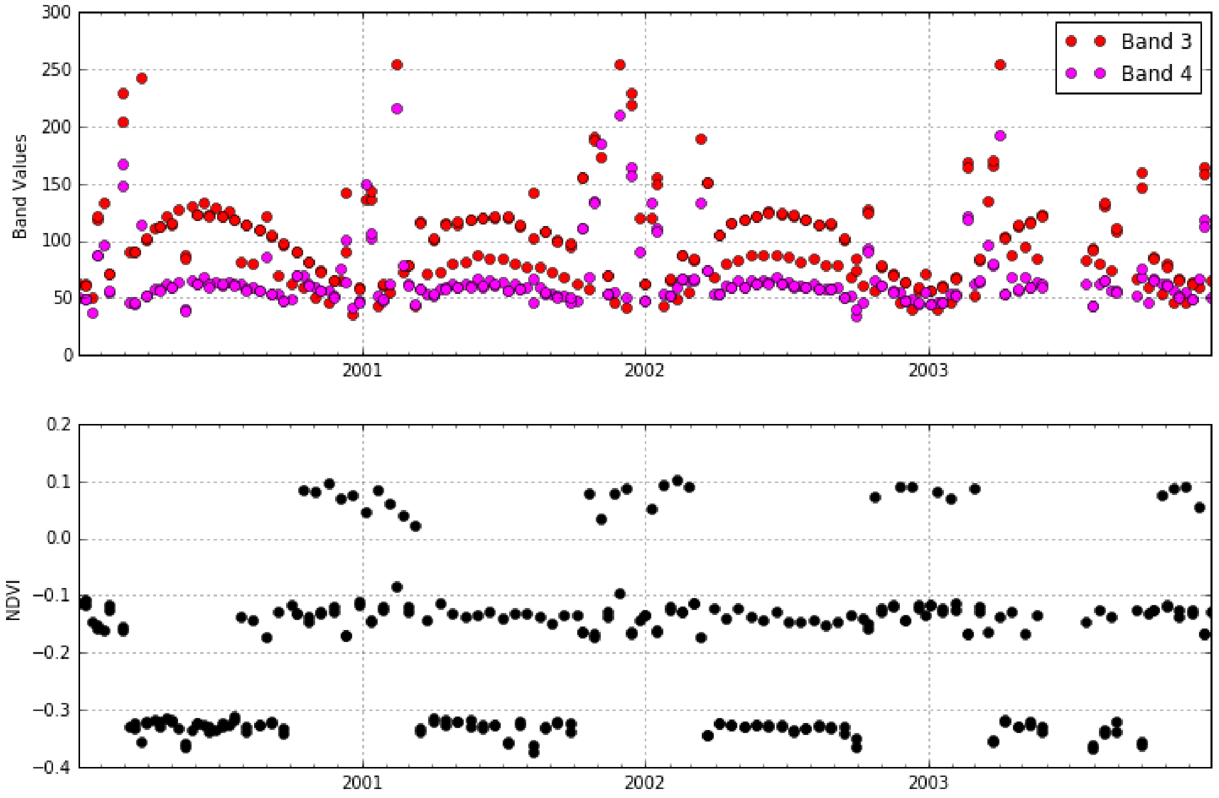
In [10]:

```
band3 = yData[:,2]
band4 = yData[:,3]
ndvi = (band4 - band3) / (band4 + band3)
```

And finally, we create a plot of MODIS band values as a function of time.

In [11]:

```
# matplotlib date format object

fig = figure(figsize=(12,8), dpi=80)

# plot the band values
ax1 = fig.add_subplot(211)
ax1.plot(time, yData[:,2], 'o', color="red", label="Band 3")
ax1.plot(time, yData[:,3], 'o', color="magenta",  label="Band 4")
ax1.legend(loc='best')
ax1.grid(True)

#plt.title('Band values as a function of time')
ax1.set_ylabel('Band Values')

# plot NDVI
ax2 = fig.add_subplot(212, sharex=ax1)
ax2.plot(time, ndvi, 'o', color="black", label="NDVI")
ax2.grid(True)
start, end = ax2.get_xlim()
ax2.xaxis.set_ticks(np.arange(start, end, 64.5))

# Format the ticks.
years    = mdates.YearLocator()   # every year
months   = mdates.MonthLocator()  # every month
yearsFmt = mdates.DateFormatter('%Y')

ax2.set_ylabel('NDVI')

ax2.xaxis.set_major_locator(years)
ax2.xaxis.set_major_formatter(yearsFmt)
ax2.xaxis.set_minor_locator(months)
```



```
In [12]: # Convert the timestamp to a numpy array
         t = np.array([i.toordinal() for i in time])
         t
```

```
Out[12]: array([730122, 730138, 730154, 730186, 730202, 730218, 730234, 730250,
                730266, 730282, 730298, 730314, 730330, 730346, 730362, 730378,
                730394, 730410, 730426, 730442, 730458, 730474, 730490, 730506,
                730522, 730538, 730554, 730570, 730586, 730602, 730618, 730634,
                730650, 730666, 730682, 730698, 730714, 730730, 730746, 730762,
                730778, 730794, 730810, 730826, 730842, 730858, 730874, 730890,
                730906, 730922, 730938, 730954, 730970, 730986, 731002, 731018,
                731034, 731050, 731066, 731082, 731098, 731114, 731130, 731146,
                731178, 731194, 731210, 731226, 731242, 731274, 731290, 731306,
                731322, 731338, 731354, 731418, 731434, 731450, 731482, 731498,
                731514, 731530, 731546, 731562, 731578, 730129, 730145, 730161,
                730177, 730193, 730209, 730225, 730241, 730257, 730273, 730289,
                730305, 730321, 730337, 730353, 730369, 730385, 730401, 730417,
                730433, 730449, 730465, 730481, 730497, 730513, 730545, 730561,
                730577, 730593, 730609, 730625, 730641, 730657, 730673, 730689,
                730705, 730721, 730737, 730753, 730769, 730785, 730801, 730833,
                730849, 730865, 730881, 730897, 730913, 730929, 730945, 730961,
                730977, 730993, 731009, 731025, 731041, 731057, 731073, 731089,
                731105, 731121, 731137, 731153, 731169, 731185, 731201, 731217,
                731233, 731249, 731265, 731281, 731297, 731313, 731329, 731345,
                731361, 731425, 731441, 731457, 731489, 731505, 731521, 731537,
                731553, 731569, 730129, 730145, 730161, 730177, 730193, 730209,
                730225, 730241, 730257, 730273, 730289, 730305, 730321, 730337,
                730353, 730369, 730385, 730401, 730417, 730433, 730449, 730465,
                730481, 730497, 730513, 730529, 730545, 730561, 730577, 730593,
                730609, 730625, 730641, 730657, 730673, 730689, 730705, 730721,
                730737, 730753, 730769, 730785, 730801, 730817, 730833, 730849,
                730865, 730881, 730897, 730913, 730929, 730945, 730961, 730977,
                730993, 731009, 731025, 731041, 731057, 731073, 731089, 731105,
                731121, 731137, 731153, 731169, 731185, 731201, 731217, 731233,
                731249, 731265, 731281, 731297, 731313, 731329, 731345, 731361,
                731425, 731441, 731457, 731489, 731505, 731521, 731537, 731553,
                731569])
```

$$\begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ \vdots & \vdots \\ t_n & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

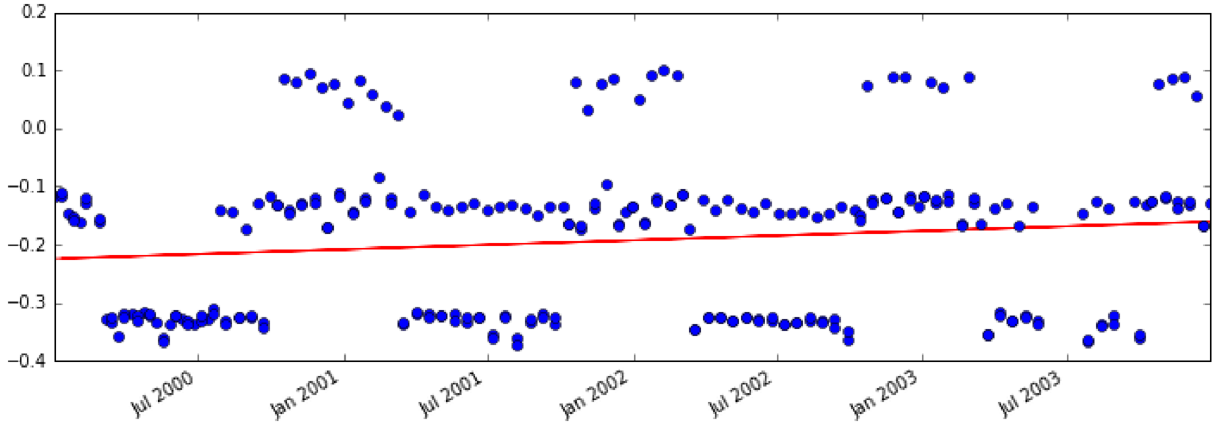
$$\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$$

```
In [13]: A = array([ t, ones(len(t))]).transpose()
         b = ndvi
         x = linalg.lstsq(A,b)[0] # obtaining the parameters
         x
```

```
Out[13]: array([  4.36365615e-05,  -3.20841879e+01])
```

```
In [14]: b_hat = A.dot(x)
```

```
In [15]: fig2 = figure(figsize=(12,4), dpi=80)
         plot(time,b_hat.transpose(),'r-',t,b,'o')
         fig2.fmt_xdata = mdates.DateFormatter('%Y-%m-%d')
         fig2.autofmt_xdate()
```



```
In [ ]:
```

```
In [16]: 18_image.getThumbUrl
```

```
  File "<ipython-input-16-c56611b87bef>", line 1
    18_image.getThumbUrl
           ^
SyntaxError: invalid syntax
```