Scientific Programming Assignment 2: Standard Deviations of Temperature

Instructions:

In [3]:
```
with open('hw2_todo.txt') as f:
    print "Instructions: use hw2.xls file \n" + f.read()
```

```
Instructions: use hw2.xls file
Column C: Year
Column D: Temperature in F

1. Program: Calculate standard deviation of temperature from arbitrary beginning year and ending year inp
ut by user.

2. Calibrate: report standard deviations for the years 1930 to 1960 and 1980 to 2010.

3. Plot: temperature averaged in 10 year increments and standard deviation in 10 year increments. Temp da
ta = red, stddev data = blue.

4. Submit as a single PDF via email by 10pm
```

0. Read in the data

Importing the data from .csv, but without the first two columns:

In [4]:
```
import numpy as np
import scipy as sp
```

```
data = np.genfromtxt('hw2.csv', delimiter = ',' , usecols = (2,3))
print data
```

```
[[ 1850.     57.7]
 [ 1851.     58. ]
 [ 1852.     58. ]
 [ 1853.     57.9]
 [ 1854.     57.9]
 [ 1855.     57.9]
 [ 1856.     57.8]
 [ 1857.     57.6]
 [ 1858.     57.6]
 [ 1859.     57.9]
 [ 1860.     57.8]
 [ 1861.     57.8]
 [ 1862.     57.5]
 [ 1863.     57.9]
 [ 1864.     57.6]
 [ 1865.     58. ]
 [ 1866.     58. ]
 [ 1867.     57.9]
 [ 1868.     58. ]
 [ 1869.     57.9]
 [ 1870.     58. ]
 [ 1871.     57.9]
 [ 1872.     58. ]
 [ 1873.     57.9]
 [ 1874.     57.8]
 [ 1875.     57.7]
 [ 1876.     57.8]
 [ 1877.     58.3]
 [ 1878.     58.5]
 [ 1879.     58. ]
 [ 1880.     58. ]
 [ 1881.     58.1]
 [ 1882.     58.1]
 [ 1883.     58. ]
```

```
[ 1884.     57.8]
[ 1885.     57.8]
[ 1886.     58. ]
[ 1887.     57.8]
[ 1888.     57.9]
[ 1889.     58.2]
[ 1890.     57.7]
[ 1891.     57.8]
[ 1892.     57.6]
[ 1893.     57.6]
[ 1894.     57.7]
[ 1895.     57.7]
[ 1896.     58.1]
[ 1897.     58.1]
[ 1898.     57.7]
[ 1899.     57.9]
[ 1900.     58.1]
[ 1901.     58. ]
[ 1902.     57.7]
[ 1903.     57.6]
[ 1904.     57.5]
[ 1905.     57.8]
[ 1906.     57.9]
[ 1907.     57.6]
[ 1908.     57.5]
[ 1909.     57.5]
[ 1910.     57.5]
[ 1911.     57.5]
[ 1912.     57.6]
[ 1913.     57.6]
[ 1914.     58. ]
[ 1915.     58.1]
[ 1916.     57.7]
[ 1917.     57.6]
[ 1918.     57.8]
[ 1919.     57.9]
```

```
[ 1920.     57.9]
[ 1921.     58. ]
[ 1922.     57.8]
[ 1923.     57.9]
[ 1924.     57.9]
[ 1925.     58. ]
[ 1926.     58.2]
[ 1927.     58. ]
[ 1928.     58. ]
[ 1929.     57.8]
[ 1930.     58.2]
[ 1931.     58.3]
[ 1932.     58.2]
[ 1933.     58. ]
[ 1934.     58.2]
[ 1935.     58.2]
[ 1936.     58.2]
[ 1937.     58.4]
[ 1938.     58.5]
[ 1939.     58.5]
[ 1940.     58.5]
[ 1941.     58.6]
[ 1942.     58.4]
[ 1943.     58.4]
[ 1944.     58.7]
[ 1945.     58.5]
[ 1946.     58.1]
[ 1947.     58.1]
[ 1948.     58.1]
[ 1949.     58.1]
[ 1950.     57.9]
[ 1951.     58.2]
[ 1952.     58.4]
[ 1953.     58.5]
[ 1954.     58. ]
[ 1955.     58. ]
```

```
[ 1956.     57.9]
[ 1957.     58.4]
[ 1958.     58.5]
[ 1959.     58.4]
[ 1960.     58.3]
[ 1961.     58.5]
[ 1962.     58.5]
[ 1963.     58.5]
[ 1964.     58. ]
[ 1965.     58.1]
[ 1966.     58.2]
[ 1967.     58.2]
[ 1968.     58.2]
[ 1969.     58.5]
[ 1970.     58.4]
[ 1971.     58.2]
[ 1972.     58.4]
[ 1973.     58.6]
[ 1974.     58.1]
[ 1975.     58.7]
[ 1976.     58.5]
[ 1977.     59. ]
[ 1978.     58.9]
[ 1979.     59.1]
[ 1980.     59.1]
[ 1981.     59.2]
[ 1982.     59. ]
[ 1983.     59.3]
[ 1984.     59. ]
[ 1985.     58.9]
[ 1986.     59.1]
[ 1987.     59.3]
[ 1988.     59.3]
[ 1989.     59.2]
[ 1990.     59.5]
[ 1991.     59.4]
```

```
[ 1992.     59.1]
[ 1993.     59.2]
[ 1994.     59.3]
[ 1995.     59.5]
[ 1996.     59.2]
[ 1997.     59.6]
[ 1998.     60. ]
[ 1999.     59.5]
[ 2000.     59.5]
[ 2001.     59.7]
[ 2002.     59.8]
[ 2003.     59.9]
[ 2004.     59.8]
[ 2005.     59.9]
[ 2006.     59.8]
[ 2007.     59.7]
[ 2008.     59.6]
[ 2009.     59.2]
[ 2010.     60.1]
[ 2011.     60.4]
[ 2012.     60.8]]
```

1. Program: Standard deviation calculation starting/ending at user-defined years in range

This works for the full range, so that's good.

In [56]:
```
# Computes the standard deviation of the temperature given a range of years
def annual_temp_stdev(data,begin,end):
    """data must be a 2-col array of year and temp, begin and end are years to define averaging"""
    #check to make sure begin and end are within range
    if begin not in data[:,0]:
        print "Beginning year not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", tr
y again."
```

```
    elif end not in data[:,0]:
        print "Ending year not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", try a
gain."
    else:
        #index the begin and end years
        ibegin = begin - data[0,0]
        iend = end - data[0,0]

        #compute standard deviation of the temperatures in the range indicated by years
        stdev = numpy.std(data[ibegin:iend+1,1])
        #round
        stdev = round(stdev,3)

        print "Standard deviation for temperatures in years " + str(begin) \
        + " to " + str(end) + " is " + str(stdev) + " degrees Fahrenheit."
    return stdev
```

In [57]:
```
annual_temp_stdev(data,1850,2012)
```

```
Standard deviation for temperatures in years 1850 to 2012 is 0.686 degrees Fahrenheit.
```

Out[57]:
```
0.686
```

In [59]:
```
numpy.std(data[:,1])
```

Out[59]:
```
0.68628455323746362
```

2. Calibrate: report standard deviations for the years 1930 to 1960 and 1980 to 2010.

In [60]:
```
annual_temp_stdev(data,1930,1960)
```

```
Standard deviation for temperatures in years 1930 to 1960 is 0.209 degrees Fahrenheit.
```

Out[60]:
```
0.209
```

In [61]:
```
annual_temp_stdev(data,1980,2010)
```

```
Standard deviation for temperatures in years 1980 to 2010 is 0.319 degrees Fahrenheit.
```

Out[61]:
```
0.319
```

3. Plot: temperature averaged in 10 year increments and standard deviation in 10 year increments.

Temp data = red, stddev data = blue.

Less interactive version of annual_temp_stdev, without rounding. How about one for the mean, too.

In [67]:
```
# Computes the standard deviation of the 2nd col given a range of items in the 1st col
def my_stdev(data,begin,end):
    """data must be a 2-col array, begin and end are endpoints to define averaging"""
    #check to make sure begin and end are within range
    if begin not in data[:,0]:
        print "begin not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", try again."
    elif end not in data[:,0]:
        print "end not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", try again."
    else:
        #index the begin and end points
        ibegin = begin - data[0,0]
        iend = end - data[0,0]

        #compute standard deviation of the data in the range indicated by [begin:end+1]
        stdev = numpy.std(data[ibegin:iend+1,1])
    return stdev
```

In [70]:
```
# Computes the mean of the 2nd col given a range of items in the 1st col
def my_mean(data,begin,end):
    """data must be a 2-col array, begin and end are endpoints to define averaging"""
    #check to make sure begin and end are within range
```

```
    if begin not in data[:,0]:
        print "begin not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", try again."
    elif end not in data[:,0]:
        print "end not in range " + str(data[0,0]) + " to " + str(data[len(data)-1,0]) + ", try again."
    else:
        #index the begin and end points
        ibegin = begin - data[0,0]
        iend = end - data[0,0]

        #compute standard deviation of the data in the range indicated by [begin:end+1]
        datamean = numpy.mean(data[ibegin:iend+1,1])
    return datamean
```

Now to make data to plot in bins.

In [252]:
```
def bin_plot(data,binsize,begin,end):
    #initialize lists to store binned averages and stdevs
    years = []
    means = []
    stdevs = []

    #loop in increments of binsize to populate lists, up to the last full decade (doesn't work for 2011,2
012 in this vsn)
    i = 0
    while i < int(round(2012-1850,-1)):
        years.append(begin+i)
        means.append(my_mean(data,begin+i,begin+i+binsize))
        stdevs.append(my_stdev(data,begin+i,begin+i+binsize))
        i += binsize
    return [years,means,stdevs]
```

Values to sanity check the first bin:

In [197]:
```
my_mean(data,1850,1860)
```

Out[197]:
```
57.827272727272721
```

In [206]:
```
my_stdev(data,1850,1860)
```

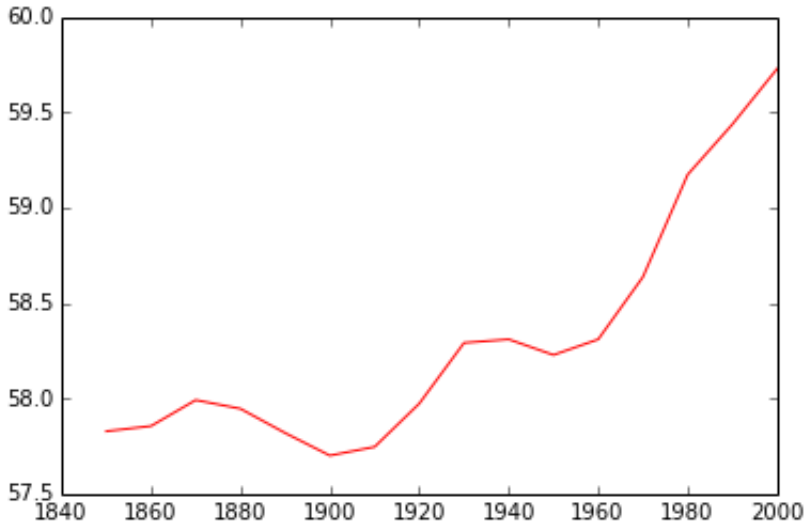
```
my_stdev(data,1850,1900)
```

Out[206]: 0.13545149477955676

In [253]:
```
plotdata=bin_plot(data,10,1850,2012)
```

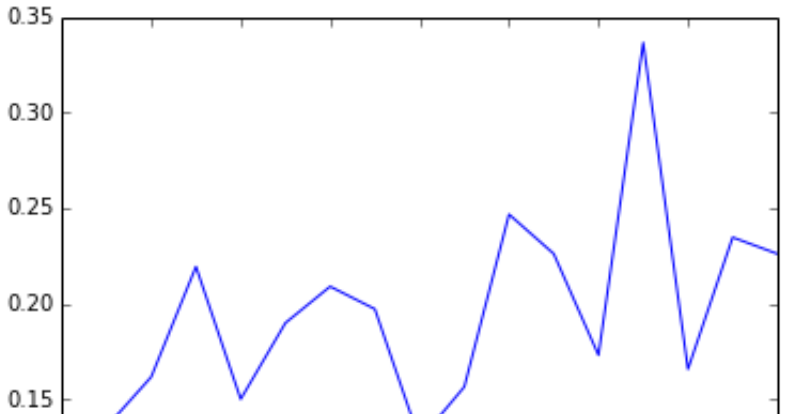
In [266]:
```
plt.plot(plotdata[0],plotdata[1],'r')
```

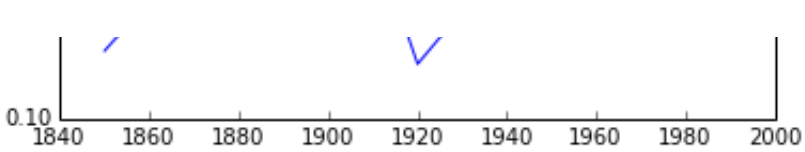
Out[266]: [<matplotlib.lines.Line2D at 0x106bc44d0>]



In [262]:
```
plt.plot(plotdata[0],plotdata[2],'b')
```

Out[262]: [<matplotlib.lines.Line2D at 0x106c22950>]





I still need to figure out how to combine with different vertical axes.... I am new to python and especially matplotlib.