Untitled5

May 20, 2015

In [190]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import exp
import math

temp1 = np.loadtxt("C:\Users\IL\Documents\SciComp\HWK6\minTemp.txt",usecols=[0])
temp2 = np.loadtxt("C:\Users\IL\Documents\SciComp\HWK6\minTemp.txt",usecols=[1,2,3,4,5,6,7,8,
temp3 = np.loadtxt("C:\Users\IL\Documents\SciComp\HWK6\maxTemp.txt",usecols=[1,2,3,4,5,6,7,8,

years = np.delete(temp1, (len(temp1)-1), axis=0)
minAve = np.delete(temp2, (len(temp2)-1), axis=0)
maxAve = np.delete(temp3, (len(temp3)-1), axis=0)

plt.plot(years,minAve[:,7])
plt.show()
```

Note: there is bad/incomplete data for year 2015 so I threw away the data for 2015
Box Car/Moving Average Method of width 5 years

In [84]:
```
minBoxCar = np.zeros((len(minAve)-4,13))
maxBoxCar = np.zeros((len(maxAve)-4,13))
for i in range(len(minAve)-4):
    minBoxCar[i,0] = years[i+2]
    maxBoxCar[i,0] = years[i+2]
    #Iterate Through each month
    for j in range(12):
        minBoxCar[i,j+1] = 1/5.*(minAve[i,j]+minAve[i+1,j]+minAve[i+2,j]+minAve[i+3,j]+minAve[
        maxBoxCar[i,j+1] = 1/5.*(maxAve[i,j]+maxAve[i+1,j]+maxAve[i+2,j]+maxAve[i+3,j]+maxAve[
print minBoxCar
```

```
[[  1.88000000e+03  -6.20000000e-01   2.36000000e+00 ...,   5.56000000e+00
    2.82000000e+00   2.00000000e-01]
 [  1.88100000e+03  -7.00000000e-01   2.30000000e+00 ...,   5.50000000e+00
    3.16000000e+00   1.18000000e+00]
 [  1.88200000e+03   7.60000000e-01   2.62000000e+00 ...,   5.44000000e+00
    3.28000000e+00   2.04000000e+00]
 ...,
 [  2.01000000e+03   1.34000000e+00   1.44000000e+00 ...,   7.30000000e+00
    4.72000000e+00   5.20000000e-01]
 [  2.01100000e+03   8.40000000e-01   1.30000000e+00 ...,   8.04000000e+00
    4.44000000e+00   1.02000000e+00]
 [  2.01200000e+03   1.42000000e+00   1.68000000e+00 ...,   8.26000000e+00
    4.42000000e+00   1.38000000e+00]]
```

Gaussian Kernel Smoothing, width 7 years

```
In [81]: minGauss = np.zeros((len(minAve)-6,13))
         maxGauss = np.zeros((len(maxAve)-6,13))

         #Need normalized Gaussian
         values = [-3,-2,-1,0,1,2,3]
         norm = sum(np.exp(-np.square(values)/7.))


         for i in range(len(minAve)-6):
             #First 3 and last 3 years ignored
             minGauss[i,0] = years[i+3]
             maxGauss[i,0] = years[i+3]

             #Iterate Through each month
             for j in range(12):
                 #Iterate through 7 Gaussian weights
                 for k in range(7):
                     minGauss[i,j+1] += minAve[i+k,j]*exp(-(k-3)**2/7.)/norm
                     maxGauss[i,j+1] += maxAve[i+k,j]*exp(-(k-3)**2/7.)/norm

         print minGauss
```

```
[[  1.88100000e+03  -3.29402363e-01   2.38742551e+00 ...,   5.44790143e+00
    3.18977656e+00   1.16465926e+00]
 [  1.88200000e+03   4.87704663e-01   2.54102419e+00 ...,   5.54780345e+00
    3.33003540e+00   1.64334130e+00]
 [  1.88300000e+03   1.35101408e+00   2.51285239e+00 ...,   5.84448648e+00
    3.31548750e+00   1.68576515e+00]
 ...,
 [  2.00900000e+03   1.58320295e+00   1.53873948e+00 ...,   7.47427393e+00
    4.65220577e+00   5.86870915e-01]
 [  2.01000000e+03   1.20758243e+00   1.49295246e+00 ...,   7.58245402e+00
    4.58401124e+00   5.55395020e-01]
 [  2.01100000e+03   1.19052851e+00   1.53860317e+00 ...,   7.89616346e+00
    4.58822801e+00   1.03524606e+00]]
```

Exponential Smoothing

```
In [93]: minExp = np.zeros((len(minAve),13))
         maxExp = np.zeros((len(maxAve),13))

         #Initialize
         minExp[0,0] = years[0]
         maxExp[0,0] = years[0]
         a = .8

         for i in range(12):
             minExp[0,i+1] = minAve[0,i]
             maxExp[0,i+1] = maxAve[0,i]

         for i in range(1,len(minAve),1):
             minExp[i,0] = years[i]
             maxExp[i,0] = years[i]

             for j in range(12):
```

```
                minExp[i,j+1] = a*minAve[i-1,j]+(1-a)*minExp[i-1,j+1]
                maxExp[i,j+1] = a*maxAve[i-1,j]+(1-a)*maxExp[i-1,j+1]

        print minExp
```

```
[[  1.87800000e+03   2.60000000e+00   3.10000000e+00 ...,   6.90000000e+00
    1.10000000e+00  -2.60000000e+00]
 [  1.87900000e+03   2.60000000e+00   3.10000000e+00 ...,   6.90000000e+00
    1.10000000e+00  -2.60000000e+00]
 [  1.88000000e+03  -1.96000000e+00   1.42000000e+00 ...,   6.26000000e+00
    1.66000000e+00  -2.20000000e+00]
 ...,
 [  2.01200000e+03   8.49766024e-01   3.03566161e+00 ...,   8.61661974e+00
    5.98745696e+00   2.05391465e+00]
 [  2.01300000e+03   2.16995320e+00   1.16713232e+00 ...,   6.92332395e+00
    4.31749139e+00   1.93078293e+00]
 [  2.01400000e+03   1.47399064e+00   6.33426464e-01 ...,   9.06466479e+00
    3.34349828e+00   3.02615659e+00]]
```

Plot of all smoothing waveforms for given month

```
In [110]: plt.plot(minBoxCar[:,0],minBoxCar[:,8],label = 'Gauss')
          plt.plot(minExp[:,0],minExp[:,8], label = 'Exp')
          plt.plot(minGauss[:,0],minGauss[:,8], label = 'BoxCar')
          plt.legend(loc='upper left')
          plt.show()
```

```
In [111]: plt.plot(maxBoxCar[:,0],maxBoxCar[:,8],label = 'Gauss')
          plt.plot(maxExp[:,0],maxExp[:,8], label = 'Exp')
          plt.plot(maxGauss[:,0],maxGauss[:,8], label = 'BoxCar')
          plt.legend(loc='upper left')
          plt.show()
```

Baseline: 30 years from 1961-1990. 100 years from 1880-1980. Choosing an arbitrary month.

```
In [203]: month = 0 # 0=Jan,..

          b1Min=0
          b1Max=0
          b2Min=0
          b2Max=0

          for i in range(1961-int(years[0]),1990-int(years[0]),1):
              b1Min += minAve[i,month]
              b1Max += maxAve[i,month]
          b1Min = b1Min/30.
          b1Max = b1Max/30.

          for i in range(1881-int(years[0]),1980-int(years[0]),1):
              b2Min += minAve[i,month]
              b2Max += maxAve[i,month]
          b2Min = b2Min/100.
          b2Max = b2Max/100.

          plt.bar(years,minAve[:,month]-b1Min)
```

```
        #Try a polynomial fit
        z = np.polyfit(years, minAve[:,month],8)
        fit = np.poly1d(z)

        plt.plot(years,fit(years),color = 'red')
        plt.show()
        plt.bar(years,maxAve[:,month]-b1Max)
        plt.show()
        plt.bar(years,minAve[:,month]-b2Min)
        plt.show()
        plt.bar(years,maxAve[:,month]-b2Max)
        plt.show()
```

```
C:\Users\IL\Anaconda\lib\site-packages\numpy\lib\polynomial.py:588: RankWarning: Polyfit may be poorly c
  warnings.warn(msg, RankWarning)
```

Blind Parameter Searching. This can be done systematically to find the highest temperature anomaly for years 2000-2010, since there are a finite number of intervals of length >= 30 we can choose. This is again done for an arbitrary month. I will look for the lowest possible baseline for both data sets.

In [204]:
```
month = 0
t1Min = 0
t2Min = 0

t1Max = 0
t2Max = 0
baseMin = 10
baseMax = 10
tempMin = 0
tempMax = 0

#Iterate through possible interval lengths
for i in range(30, len(minAve)+1,1):
    #Iterate through first element of new interval
    for j in range(0,len(minAve)-i+1,1):
        #Get the average
        for k in range(j,j+i,1):
            tempMin += minAve[k,month]
            tempMax += maxAve[k,month]
        tempMin = tempMin/float(i)
        tempMax = tempMax/float(i)
        if(tempMin < baseMin):
            baseMin = tempMin
            t1Min = j+years[0]
            t2Min = j+years[0] + i
        if(tempMax < baseMax):
            baseMax = tempMax
            t1Max = j+years[0]
            t2Max = j+years[0] + i
        tempMin =0
        tempMax = 0
print(t1Min,t2Min,t1Max,t2Max)

plt.bar(years,maxAve[:,month]-baseMax)
```

```
plt.show()
plt.bar(years,minAve[:,month]-baseMin)
plt.show()
```

```
(1939.0, 1969.0, 1879.0, 1909.0)
```

Seasonal Data

```
In [180]: seaMin=np.zeros((len(minAve),2))
          seaMax=np.zeros((len(minAve),2))
          #Get average temp per season. col 1 will be summer, col 2 winter
          ctr = 0
          for i in range(len(minAve)):
              #Summer
              for j in range(5,8,1):
                  ctr += minAve[i,j]
              seaMin[i,0] = ctr/3
              ctr = 0
              for j in range(5,8,1):
                  ctr += maxAve[i,j]
              seaMax[i,0] = ctr/3
              ctr = 0

              #Winter
              for j in range(2):
                  ctr += minAve[i,j]
              for j in range(10,12,1):
                  ctr += minAve[i,j]
              seaMin[i,1] = ctr/4
              ctr = 0
              for j in range(2):
                  ctr += maxAve[i,j]
              for j in range(10,12,1):
                  ctr += maxAve[i,j]
              seaMax[i,1] = ctr/4
              ctr = 0
```

Optimize baseline parameter to get minimum baseline average temperature for each of the four data sets. I will again use a minimum interval of 30 years.

```
In [183]: t1SMin = 0 #years for summer
          t2SMin = 0
          t1SMax = 0
          t2SMax = 0

          t1WMin = 0 #years for winter
          t2WMin = 0
          t1WMax = 0
          t2WMax = 0

          baseWMin = 100
          baseWMax = 100
          baseSMin = 100
          baseSMax = 100
          temp1 = 0
```

```
temp2 = 0
temp3 = 0
temp4 = 0

#Iterate through possible interval lengths
for i in range(30, len(minAve)+1,1):
    #Iterate through first element of new interval
    for j in range(0,len(minAve)-i+1,1):
        #Get the average
        for k in range(j,j+i,1):
            temp1 += seaMin[k,0]
            temp2 += seaMax[k,0]
            temp3 += seaMin[k,1]
            temp4 += seaMax[k,1]

        temp1 = temp1/float(i)
        temp2 = temp2/float(i)
        temp3 = temp3/float(i)
        temp4 = temp4/float(i)

        #New minimum found?
        if(temp1 < baseSMin):
            baseSMin = temp1
            t1SMin = j+years[0]
            t2SMin = j+years[0] + i
        if(temp2 < baseSMax):
            baseSMax = temp2
            t1SMax = j+years[0]
            t2SMax = j+years[0] + i
        if(temp3 < baseWMin):
            baseWMin = temp3
            t1SMin = j+years[0]
            t2SMin = j+years[0] + i
        if(temp1 < baseWMax):
            baseWMax = temp4
            t1SMax = j+years[0]
            t2SMax = j+years[0] + i
        temp1 =0
        temp2=0
        temp3 =0
        temp4 = 0
print(baseSMin,baseSMax, baseWMin, baseWMax)

plt.bar(years,seaMin[:,0]-baseSMin)
plt.show()
plt.bar(years,seaMax[:,0]-baseSMax)
plt.show()
plt.bar(years,seaMin[:,1]-baseWMin)
plt.show()
plt.bar(years,seaMax[:,1]-baseWMax)
plt.show()
```

```
(10.636507936507936, 19.126666666666669, 1.7054687500000003, 6.8558333333333339)
```

Poisson Statistics: Determine average number of negative average min temp months per decade.

```
In [210]: #Start at 1880, end at 1960
          data = np.zeros((8,2))
          for i in range(8):
              data[i,0] = 1880+10*i

          while(years[i] < 1960):
              for j in range(12):
                  if(minAve[i,j]<0):
                      data[int(years[i]-1880)/10,1] += 1

              i += 1
          print data
          print np.average(data[:,1])
          print exp(-np.average(data[:,1])) #Chance of 0 events in 2000-2009
```

```
[[ 1880.     7.]
 [ 1890.    10.]
 [ 1900.     6.]
 [ 1910.     6.]
 [ 1920.     3.]
 [ 1930.     3.]
 [ 1940.     8.]
 [ 1950.     7.]]
6.25
0.00193045413623
```

```
In [195]: print math.ceil(len(minAve)/10.)+1
```

```
15.0
```

```
In [ ]:
```