Lab 6

# I. Data Set.

The 'historical\_temperature\_philly.csv' contains 5 columns, which is 'Month, Day, Year, Max, Min'. It records the Max and Min temperature of each day from year 1874 – 2014. The daily Max and Min temperature of 2015 January is also recorded. Therefore there are 141 whole years' temperature + 1 month temperature.

# II. Matrix generation

Two ways of matrix representation generation are provided for a give list of years based on the 'historical\_temperature\_philly.csv' data in matrix\_generation.py file.

1). monthly\_ave\_mtx(years, colname)

Create a matrix with size N by 12.

Each row contains monthly average min or max temperature of a whole year.

arguments:

years -- N length list-like years, e.g., [2012, 2011, 2013], or np.array([2012,2011,2013])

colname -- a string either "Max" or "Min".

outputs:

month\_mtx -- N by 12 matrix

usage example:

# To create a monthly average of max temperature matrix of years 1874 - 2014

import matrix\_generation

monthly\_maxave\_mtx = matrix\_generation.monthly\_ave\_mtx(range(1874,2015), 'Max')

2). daily\_mtx(years, month, colname)

Create a matrix with size N by (number of days in month)

Each row contains daily max or min temperature of the month in the year.

arguments:

years -- a N length list-like years, e.g., [2012, 2011, 2013], or np.array([2012,2011,2013])

month -- an interger between [1, 12].

colname -- a string either "Max" or "Min".

outputs:

daily\_mtx -- a N by 30 or N by 31 or N by 28 table, where N is the length of years. \

Column number varies with different month selection

usage example:

# To create a daily max temperature matrix in Jan of years from 1874-2015

import matrix\_generation

daily\_max\_mtx\_jan = matrix\_generation.daily\_mtx(range(1874, 2016), 1, "Max")

# III. Analysis Tasks

## 1). Find the nearest neighbors of 2014 with January daily max or min temperature.

|  |
| --- |
| # Create the required matrix  import matrix\_generation  import numpy as np  import matplotlib.pyplot as plt  years = np.arange(1874, 2015)  daily\_max\_mtx\_jan = matrix\_generation.daily\_mtx(years, 1, "Max")  # Find the nearest neighbors of 2014 based on the daily January max temperature matrix.  from scipy.spatial.distance import pdist, squareform  data\_dist\_flat = pdist(daily\_max\_mtx\_jan, 'euclidean') # computing the distance  data\_dist\_square = squareform(data\_dist\_flat) # make the flat distance as a matrix  nearest\_neighbors\_indices = data\_dist\_square.argsort(axis = 1) # sort each row in ascending order  k = 5  knn\_years = years[nearest\_neighbors\_indices[-1, 2:(2+k)]]  print('The 5 nearest years of 2014 based on January daily max temperature is: ' + str(knn\_years)) |

This piece of code printed the following line:

The 5 nearest years of 2014 based on January daily max temperature is: [1987 1882 1961 1905 1895]

## 2). Find the nearest neighbors of 2014 with whole year monthly average of max or min temperature.

|  |
| --- |
| # Create the required matrix  years = np.arange(1874, 2015)  monthly\_maxave\_mtx = matrix\_generation.monthly\_ave\_mtx(years, 'Max')  # Find the nearest neighbors of 2014 based on the monthly average of max temperature matrix.  from scipy.spatial.distance import pdist, squareform  data\_dist\_flat = pdist(monthly\_maxave\_mtx, 'euclidean') # computing the distance  data\_dist\_square = squareform(data\_dist\_flat) # make the flat distance as a matrix  nearest\_neighbors\_indices = data\_dist\_square.argsort(axis = 1) # sort each row in ascending order  k = 5  knn\_years = years[nearest\_neighbors\_indices[-1, 2:(2+k)]]  print('The 5 nearest years of 2014 based on monthly average of max temperature matrix is: ' + str(knn\_years)) |

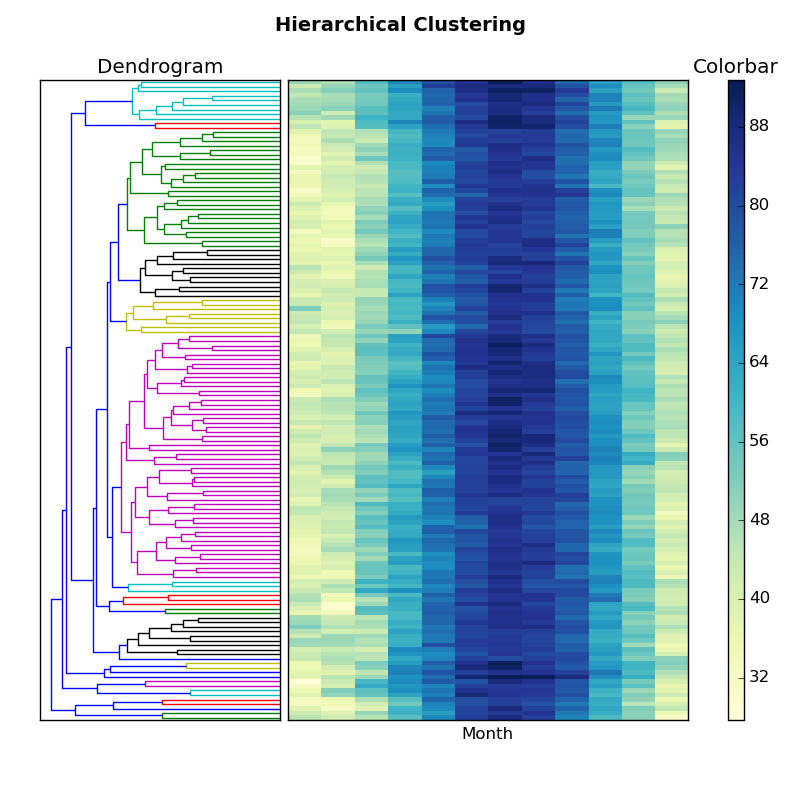
This piece of code printed the following line:

The 5 nearest years of 2014 based on monthly average of max temperature matrix is: [1959 1957 1971 1996 1951]

## 3). Cluster the 141 years (except 2015) with their monthly average of max or min temperature, using hierarchical clustering. And draw heatmap with the rows sorted according to the leaf order of HC

|  |
| --- |
| # Create the required matrix  years = np.arange(1874,2015)  monthly\_maxave\_mtx = matrix\_generation.monthly\_ave\_mtx(years, 'Max')  # Hierarchical Clustering  from scipy.spatial.distance import pdist, squareform  from scipy.cluster.hierarchy import linkage, dendrogram  data\_dist\_flat = pdist(monthly\_maxave\_mtx, 'euclidean') # computing the distance  data\_linkage = linkage(data\_dist\_flat, method='average')  # Plot dendrogram and the heatmap of temperatures.  # Dendrogram  fig = plt.figure(figsize=(8,8))  ax1 = fig.add\_axes([0.05,0.1,0.3,0.8])  Z = dendrogram(data\_linkage,orientation='right')  plt.xticks([])  plt.yticks([])  plt.title('Dendrogram')  # Heatmap  ax2 = fig.add\_axes([0.36, 0.1,0.5,0.8])  idx = Z['leaves']  data\_ordered = monthly\_maxave\_mtx[idx, :]  im = ax2.matshow(data\_ordered, aspect='auto', origin='lower', cmap=plt.cm.YlGnBu)  plt.xticks([])  plt.yticks([])  #plt.title('Heatmap')  plt.xlabel('Month')  # Colorbar  ax3 = fig.add\_axes([0.91,0.1,0.02,0.8])  plt.colorbar(im, cax=ax3)  plt.title('Colorbar')  plt.suptitle('Hierarchical Clustering', fontweight='bold', fontsize=14) |

This code created the figure as below:



## 4). Find the nearest neighbors of 2015 with January daily max or min temperature. Predict the max or min temperature in June.

|  |
| --- |
| years = np.arange(1874, 2016)  daily\_max\_mtx\_jan = matrix\_generation.daily\_mtx(years, 1, "Max")  from scipy.spatial.distance import pdist, squareform  data\_dist\_flat = pdist(monthly\_maxave\_mtx, 'euclidean') # computing the distance  data\_dist\_square = squareform(data\_dist\_flat) # make the flat distance as a matrix  nearest\_neighbors\_indices = data\_dist\_square.argsort(axis = 1) # sort each row in ascending order  k = 5  knn\_years = years[nearest\_neighbors\_indices[-1, 2:(2+k)]]  prediction = matrix\_generation.daily\_mtx(knn\_years, 6, "Max")  prediction = np.mean(prediction, axis = 0)  print("The prediction for the Max temperature of year 2015 in July is: " + str(prediction)) |

The above code printed the following line:

The prediction for the Max temperature of year 2015 in July is: [ 78.2 77.4 76.2 82.4 77.8 80.2 85. 80. 81. 79.8 81. 84.4 81. 77.6 80.4 82.8 83. 82.8 80.8 83.8 86.4 88.2 86. 87.8 88.8 86.6 88.2 88. 84.4 83.2]