WeatherData descriptive statistics.

September 27, 2017

0.1 Import

```
In [3]: # necessary imports for the descriptive statistics
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
```

0.2 General description

The following dataset is the rain and temperature data from January to December for the period of 1961-2000, which is from four county meteorological stations representing Mengla, Menglun, Menghai, Jinghong.

Variable tag description:

Mla=Mengla; Mlun=Menglun; Mhai=Menghai; Jhong=Jinghong.

R=monthly rainfall; AT=monthly temperature; 01,02,...,12 means 12 months from January through December.

```
In [41]: import pandas as pd
         df=pd.read_csv('C:/Users/Administrator/Desktop/Python/exercise/WeatherData
         df.head()
```

Out[41]:

	MlaR01	MlaR02	MlaR03	MlaR04	MlaR05	MlaR06	MlaR07	MlaR08	MlaR09
0	4.8	46.8	41.4	97.7	262.8	205.4	215.5	415.2	245.1
1	12.9	45.0	16.8	105.3	166.0	258.6	91.0	330.9	93.0
2	3.3	5.2	42.0	114.7	90.0	314.1	363.8	350.6	72.9
3	6.1	9.7	46.5	88.7	125.4	335.7	448.6	241.5	104.6
4	19.2	28.3	19.8	44.1	176.2	228.2	340.3	235.8	194.2

	MlaR10	...	JhongAT3	JhongAT4	JhongAT5	JhongAT6	JhongAT7	\
0	21.5	...	21.3	24.2	25.8	26.0	25.6	
1	115.8	...	19.4	24.4	26.1	25.1	25.6	
2	228.1	...	20.4	24.0	26.3	25.9	25.3	
3	155.6	...	20.5	23.7	24.8	25.1	24.2	
4	128.6	...	20.2	24.3	25.9	25.3	25.4	

	JhongAT8	JhongAT9	JhongAT10	JhongAT11	JhongAT12
0	24.8	24.1	22.9	19.0	16.7
1	24.5	23.9	22.1	18.8	15.4

```
2      24.6      25.0       22.0       19.4       15.4
3      24.9      24.7       23.3       18.8       15.1
4      24.3      24.1       22.2       18.5       17.4

[5 rows x 96 columns]
```

```
In [6]: df.shape
```

```
Out[6]: (40, 96)
```

We can see that the index of the DataFrame consists of 40 rows representing 40 years(1961-2000), and there are 96 columns in the DataFrame, which consist of the rain and temperature data from January to December.

```
In [7]: df.index #examine the index
```

```
Out[7]: RangeIndex(start=0, stop=40, step=1)
```

```
In [8]: df.columns #get the columns
```

```
Out[8]: Index(['MlaR01', 'MlaR02', 'MlaR03', 'MlaR04', 'MlaR05', 'MlaR06', 'MlaR07'
               'MlaR08', 'MlaR09', 'MlaR10', 'MlaR11', 'MlaR12', 'MlaAT01', 'MlaAT0
               'MlaAT03', 'MlaAT04', 'MlaAT05', 'MlaAT06', 'MlaAT07', 'MlaAT08',
               'MlaAT09', 'MlaAT10', 'MlaAT11', 'MlaAT12', 'MlunR1', 'MlunR2',
               'MlunR3', 'MlunR4', 'MlunR5', 'MlunR6', 'MlunR7', 'MlunR8', 'MlunR9'
               'MlunR10', 'MlunR11', 'MlunR12', 'MlunAT1', 'MlunAT2', 'MlunAT3',
               'MlunAT4', 'MlunAT5', 'MlunAT6', 'MlunAT7', 'MlunAT8', 'MlunAT9',
               'MlunAT10', 'MlunAT11', 'MlunAT12', 'MhaiR1', 'MhaiR2', 'MhaiR3',
               'MhaiR4', 'MhaiR5', 'MhaiR6', 'MhaiR7', 'MhaiR8', 'MhaiR9', 'MhaiR10
               'MhaiR11', 'MhaiR12', 'MhaiAT1', 'MhaiAT2', 'MhaiAT3', 'MhaiAT4',
               'MhaiAT5', 'MhaiAT6', 'MhaiAT7', 'MhaiAT8', 'MhaiAT9', 'MhaiAT10',
               'MhaiAT11', 'MhaiAT12', 'JhongR1', 'JhongR2', 'JhongR3', 'JhongR4',
               'JhongR5', 'JhongR6', 'JhongR7', 'JhongR8', 'JhongR9', 'JhongR10',
               'JhongR11', 'JhongR12', 'JhongAT1', 'JhongAT2', 'JhongAT3', 'JhongAT
               'JhongAT5', 'JhongAT6', 'JhongAT7', 'JhongAT8', 'JhongAT9', 'JhongAT
               'JhongAT11', 'JhongAT12'],
              dtype='object')
```

```
In [9]: df.head() #peek at the first 5 rows of the data using the .head() method
```

```
Out[9]:    MlaR01  MlaR02  MlaR03  MlaR04  MlaR05  MlaR06  MlaR07  MlaR08  MlaR09
        0     4.8    46.8    41.4    97.7   262.8   205.4   215.5   415.2   245.1
        1    12.9    45.0    16.8   105.3   166.0   258.6    91.0   330.9    93.0
        2     3.3     5.2    42.0   114.7    90.0   314.1   363.8   350.6    72.9
        3     6.1     9.7    46.5    88.7   125.4   335.7   448.6   241.5   104.6
        4    19.2    28.3    19.8    44.1   176.2   228.2   340.3   235.8   194.2

           MlaR10    ...      JhongAT3  JhongAT4  JhongAT5  JhongAT6  JhongAT7  \
        0    21.5    ...          21.3      24.2      25.8      26.0      25.6
```

```
1   115.8    ...          19.4      24.4      26.1      25.1      25.6
2   228.1    ...          20.4      24.0      26.3      25.9      25.3
3   155.6    ...          20.5      23.7      24.8      25.1      24.2
4   128.6    ...          20.2      24.3      25.9      25.3      25.4

   JhongAT8  JhongAT9  JhongAT10  JhongAT11  JhongAT12
0      24.8      24.1       22.9       19.0       16.7
1      24.5      23.9       22.1       18.8       15.4
2      24.6      25.0       22.0       19.4       15.4
3      24.9      24.7       23.3       18.8       15.1
4      24.3      24.1       22.2       18.5       17.4

[5 rows x 96 columns]
```

The first 5 rows representing the rain and air temperature monthly data from the 4 count meteorological stations for the period of 1961-1965. Likewise, the last 5 rows representing the data for the period of 1996-2000.

```
In [10]: df.tail() #peek at the last 5 rows of the data using the .tail() method
```

```
Out[10]:     MlaR01  MlaR02  MlaR03  MlaR04  MlaR05  MlaR06  MlaR07  MlaR08  MlaR0
         35     0.0    41.5    72.7    67.0   153.0   137.8   354.4   316.5   134.5
         36     0.5     0.0    63.8   122.5    29.6   107.9   547.8   186.1   273.2
         37     0.0     4.5    30.2   183.2   176.7   103.0   351.3   184.3   127.2
         38    46.1     0.1    61.1    79.4   182.9   305.3   179.2   281.0   211.3
         39    11.8    42.5    79.9    85.4   241.3   242.9   390.7   388.4   221.3

             MlaR10    ...      JhongAT3  JhongAT4  JhongAT5  JhongAT6  JhongAT7  \
         35    74.3    ...          22.6      24.5      26.0      26.0      25.3
         36    56.3    ...          22.4      23.0      26.9      26.6      25.7
         37    61.3    ...          22.7      24.2      26.3      26.8      25.4
         38    74.0    ...          22.0      25.8      24.4      26.5      25.7
         39   213.5    ...          20.6      24.6      24.2      25.7      25.7

             JhongAT8  JhongAT9  JhongAT10  JhongAT11  JhongAT12
         35      25.4      25.2       23.0       21.2       18.3
         36      26.0      24.0       23.4       20.5       19.4
         37      25.6      24.9       23.6       20.2       18.2
         38      25.3      24.8       23.4       19.7       14.6
         39      25.8      24.4       23.6       19.3       18.6

         [5 rows x 96 columns]
```

0.3 Summary statistics

The following code returns summary statistics on the rain and temperature data, including the count of items that are not part of NaN; the mean and standard deviation; minimum and maximum values; and the values of the 25,50,and 75 percentiles:

```
In [11]: df.describe() # returns summary statistics
```

Out[11]:

	MlaR01	MlaR02	MlaR03	MlaR04	MlaR05	MlaR
count	40.000000	40.000000	40.000000	40.000000	40.000000	40.0000
mean	19.562500	22.697500	37.550000	96.352500	167.657500	225.6800
std	24.024557	24.924578	28.946662	45.086094	69.049926	76.8601
min	0.000000	0.000000	0.000000	25.700000	29.600000	103.0000
25%	3.300000	4.225000	12.975000	62.925000	120.950000	163.6750
50%	12.150000	11.900000	34.000000	91.250000	170.950000	219.9000
75%	28.875000	38.200000	53.800000	122.900000	202.675000	278.9000
max	111.400000	96.700000	130.800000	202.100000	377.200000	410.7000

	MlaR07	MlaR08	MlaR09	MlaR10	...	JhongAT3
count	40.00000	40.000000	40.000000	40.000000	...	40.000000
mean	303.86000	308.070000	158.290000	93.927500	...	21.477500
std	100.31874	90.486716	72.134205	52.928239	...	0.881283
min	91.00000	173.800000	40.300000	13.300000	...	19.400000
25%	234.95000	236.400000	108.350000	55.650000	...	20.700000
50%	291.80000	302.350000	140.350000	84.600000	...	21.400000
75%	356.75000	360.050000	210.850000	129.675000	...	22.100000
max	547.80000	508.100000	321.700000	228.100000	...	23.400000

	JhongAT4	JhongAT5	JhongAT6	JhongAT7	JhongAT8	JhongAT9	\
count	40.000000	40.0000	40.00000	40.000000	40.00000	40.000000	
mean	24.377500	25.7775	25.91500	25.507500	25.22000	24.600000	
std	1.041323	0.8980	0.59938	0.460984	0.52242	0.458537	
min	22.300000	24.2000	24.90000	24.200000	24.00000	23.400000	
25%	23.800000	25.1000	25.37500	25.300000	24.97500	24.300000	
50%	24.400000	25.7000	26.00000	25.550000	25.30000	24.650000	
75%	24.925000	26.3000	26.30000	25.725000	25.60000	24.900000	
max	26.800000	27.7000	27.20000	26.700000	26.40000	25.500000	

	JhongAT10	JhongAT11	JhongAT12
count	40.00000	40.000000	40.000000
mean	22.79750	19.622500	16.472500
std	0.74644	0.914691	1.201706
min	21.10000	17.100000	13.300000
25%	22.27500	19.000000	15.625000
50%	22.85000	19.450000	16.550000
75%	23.40000	20.200000	17.125000
max	24.10000	21.400000	19.400000

[8 rows x 96 columns]

```
In [12]: MlaRmin=df.min()
         MlaRmin[0:12]
```

```
Out[12]: MlaR01      0.0
         MlaR02      0.0
```

```
MlaR03      0.0
MlaR04     25.7
MlaR05     29.6
MlaR06    103.0
MlaR07     91.0
MlaR08    173.8
MlaR09     40.3
MlaR10     13.3
MlaR11      0.0
MlaR12      0.0
dtype: float64
```

We can see that the smallest rainfall occurred in January, February, March, November and December within the 40 years in Mla.

```
In [14]: MlaATmin=df.min()
         MlaATmin[12:24]
```

```
Out[14]: MlaAT01    13.1
         MlaAT02    15.6
         MlaAT03    18.0
         MlaAT04    21.3
         MlaAT05    23.4
         MlaAT06    24.2
         MlaAT07    23.8
         MlaAT08    23.2
         MlaAT09    22.7
         MlaAT10    20.4
         MlaAT11    16.6
         MlaAT12    12.4
         dtype: float64
```

We can see that the temperature are lower than 20 degrees for 5 months, which also have the smallest rainfall.

0.4 Selecting DataFrame

Selecting the rainfall data from January to December within 40 years in Mla:

```
In [16]: # rows and columns can be retrieved by location using .iloc[]
         # get rows from 0 to 39, and columns from 0 to 12
         MlaR=df.iloc[:40,0:12]
         MlaR.head() # show the first 5 years' data
```

```
Out[16]:    MlaR01  MlaR02  MlaR03  MlaR04  MlaR05  MlaR06  MlaR07  MlaR08  MlaR09
         0     4.8    46.8    41.4    97.7   262.8   205.4   215.5   415.2   245.1
         1    12.9    45.0    16.8   105.3   166.0   258.6    91.0   330.9    93.0
         2     3.3     5.2    42.0   114.7    90.0   314.1   363.8   350.6    72.9
         3     6.1     9.7    46.5    88.7   125.4   335.7   448.6   241.5   104.6
```

```
4    19.2    28.3    19.8    44.1   176.2   228.2   340.3   235.8   194.2

   MlaR10  MlaR11  MlaR12
0    21.5    47.9    53.6
1   115.8    12.3     5.4
2   228.1   101.9    11.3
3   155.6    33.4    47.1
4   128.6   140.2    68.7
```

The following code returns summary statistics on the rainfall data from January to December within 40 years in Mla:

```
In [17]: MlaR.describe() # returns summary statistics
```

```
Out[17]:            MlaR01     MlaR02      MlaR03      MlaR04      MlaR05      MlaR
         count   40.000000  40.000000   40.000000   40.000000   40.000000   40.0000
         mean    19.562500  22.697500   37.550000   96.352500  167.657500  225.6800
         std     24.024557  24.924578   28.946662   45.086094   69.049926   76.8601
         min      0.000000   0.000000    0.000000   25.700000   29.600000  103.0000
         25%      3.300000   4.225000   12.975000   62.925000  120.950000  163.6750
         50%     12.150000  11.900000   34.000000   91.250000  170.950000  219.9000
         75%     28.875000  38.200000   53.800000  122.900000  202.675000  278.9000
         max    111.400000  96.700000  130.800000  202.100000  377.200000  410.7000

                   MlaR07      MlaR08      MlaR09      MlaR10      MlaR11      MlaR
         count   40.00000   40.000000   40.000000   40.000000   40.000000   40.0000
         mean   303.86000  308.070000  158.290000   93.927500   54.047500   29.5425
         std    100.31874   90.486716   72.134205   52.928239   49.021225   32.1075
         min     91.00000  173.800000   40.300000   13.300000    0.000000    0.0000
         25%    234.95000  236.400000  108.350000   55.650000   19.700000    4.2750
         50%    291.80000  302.350000  140.350000   84.600000   41.500000   17.8500
         max    547.80000  508.100000  321.700000  228.100000  193.500000  130.7000
```

The following example selects all line data whose index label is 4, performing the monthly rainfall data for 12 months in the year 1965:

```
In [19]: # using .ix() method to select rows by index label or location
         MlaR1965=MlaR.ix[4]
         MlaR1965
```

```
Out[19]: MlaR01     19.2
         MlaR02     28.3
         MlaR03     19.8
         MlaR04     44.1
         MlaR05    176.2
         MlaR06    228.2
         MlaR07    340.3
         MlaR08    235.8
```

```
MlaR09    194.2
MlaR10    128.6
MlaR11    140.2
MlaR12     68.7
Name: 4, dtype: float64
```

0.5 Boolean selection

The following code demonstrates identifying items in a Series where the rainfall are greater than 200:

```
In [20]: # A Boolean selection applies a logical expression
         #to the value of the Series
         MlaR1965>200

Out[20]: MlaR01    False
         MlaR02    False
         MlaR03    False
         MlaR04    False
         MlaR05    False
         MlaR06     True
         MlaR07     True
         MlaR08     True
         MlaR09    False
         MlaR10    False
         MlaR11    False
         MlaR12    False
         Name: 4, dtype: bool
```

The following code extracts subsets whose values are greater than 200:

```
In [21]: # using the name of the Series inside of the [] operator
         # to extract subsets of data based on its contents:
         MlaR1965[MlaR1965>200]

Out[21]: MlaR06    228.2
         MlaR07    340.3
         MlaR08    235.8
         Name: 4, dtype: float64
```

We can see that the rainfall values in June, July and August are great than 200.
The following code extracts subsets whose values are greater than 100 and less than 200:

```
In [22]: MlaR1965[(MlaR1965>100)&(MlaR1965<200)]

Out[22]: MlaR05    176.2
         MlaR09    194.2
         MlaR10    128.6
         MlaR11    140.2
         Name: 4, dtype: float64
```

It is convenient for us to see that the rainfall values in May, September, October and November are great than 100 and less than 200.

The following code extracts the air temperature monthly data in Mla in the year 1961:

```
In [24]: # get row in location 1, and columns from 12 to 24
         MlaAT1961=df.iloc[:1,12:24]
         MlaAT1961
```

Out[24]:

	MlaAT01	MlaAT02	MlaAT03	MlaAT04	MlaAT05	MlaAT06	MlaAT07	MlaAT08
0	15.8	18.7	20.5	22.9	24.5	24.4	24.8	24.2

	MlaAT09	MlaAT10	MlaAT11	MlaAT12
0	23.6	22.2	18.7	16.2

```
In [25]: # acquire the mean of the series
         MlaAT1961.ix[0].mean()
```

```
Out[25]: 21.375
```

We can see the annual air temperature value is 21.375 degree in the year 1961.

Taking the rainfall data of the four places in January within 40 years as an example, I want to do some summary statistics and make further comparisons. The codes are as follows:

```
In [31]: # or type: df[[0,24,48,72]]
         # to look up the rows with special index labels
         dfR1=df[['MlaR01','MlunR1','MhaiR1','JhongR1']]
         dfR1
```

Out[31]:

	MlaR01	MlunR1	MhaiR1	JhongR1
0	4.8	4.2	4.2	5.6
1	12.9	11.9	8.0	6.0
2	3.3	4.1	2.3	2.0
3	6.1	4.3	3.7	5.0
4	19.2	31.8	10.9	9.6
5	65.2	32.9	13.0	9.6
6	30.6	22.8	12.8	28.7
7	51.4	85.6	89.9	88.1
8	16.1	4.6	11.3	6.4
9	20.4	19.6	8.7	8.3
10	9.6	5.6	7.1	10.8
11	14.0	25.2	20.6	21.9
12	3.2	6.1	4.0	3.4
13	3.0	3.5	1.0	2.3
14	111.4	79.1	62.4	62.0
15	9.4	34.7	24.5	31.0
16	20.1	0.1	14.1	9.9
17	77.6	52.0	76.2	51.7
18	28.3	1.6	10.1	19.4
19	0.5	1.4	0.0	0.3

```
20    50.0    35.5     6.2     17.4
21    12.7    30.2    21.2     14.9
22    33.2    22.0    12.7     17.1
23     2.0     3.2     0.0      0.9
24    31.4    10.1     0.0      0.0
25     0.5     4.0     0.6      1.6
26    33.9     8.0    14.1     16.3
27     3.7     1.9     0.0      0.0
28    10.4    15.4     3.1     17.3
29    12.5     2.6     0.5      0.0
30     3.3     0.3     1.7      0.1
31    13.5    23.3     7.7      6.5
32     2.0     1.7     1.3      3.3
33     4.0     0.3     0.9     17.0
34     3.9     3.8     1.1      0.7
35     0.0     0.5     0.0      0.0
36     0.5     5.0     2.5      0.0
37     0.0     0.0     0.2      0.0
38    46.1    61.8    29.5     35.6
39    11.8     7.0     9.2     11.6
```

```
In [32]: dfR1.describe()# returns summary statistics
```

```
Out[32]:            MlaR01     MlunR1     MhaiR1    JhongR1
         count   40.000000  40.000000  40.000000  40.000000
         mean    19.562500  16.692500  12.432500  13.557500
         std     24.024557  21.381988  20.021327  18.480161
         min      0.000000   0.000000   0.000000   0.000000
         25%      3.300000   3.050000   1.075000   1.425000
         50%     12.150000   5.850000   6.650000   7.400000
         75%     28.875000  23.775000  12.850000  17.150000
         max    111.400000  85.600000  89.900000  88.100000
```

We can see that for the period of 1961-2000, January rainfall in Mla is larger than the other three counties, and the degree of dispersion in which is the largest too. In addition, the average rainfall in January is as low as 0 in the four counties.

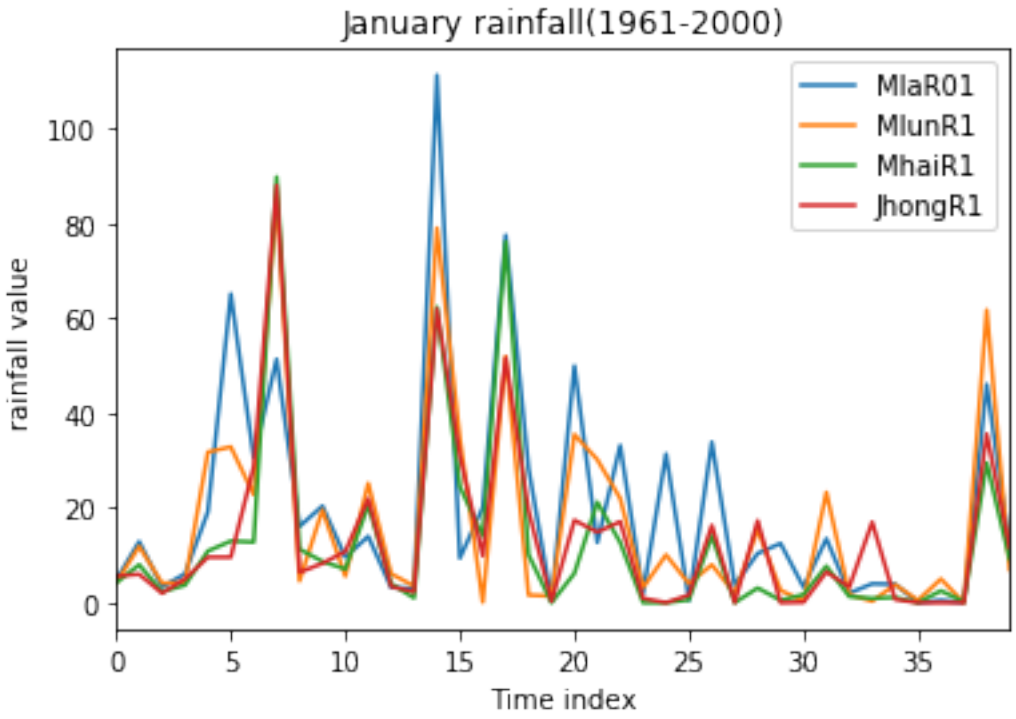
```
In [33]: dfAT1=df[['MlaAT01','MlunAT1','MhaiAT1','JhongAT1']]
         dfAT1.head()
```

```
Out[33]:    MlaAT01  MlunAT1  MhaiAT1  JhongAT1
         0     15.8     16.4     12.8      16.3
         1     14.8     15.3     12.4      15.7
         2     13.2     14.0     11.3      14.3
         3     15.7     15.7     11.8      15.4
         4     14.3     14.3     11.5      14.6
```

0.6 Visualization

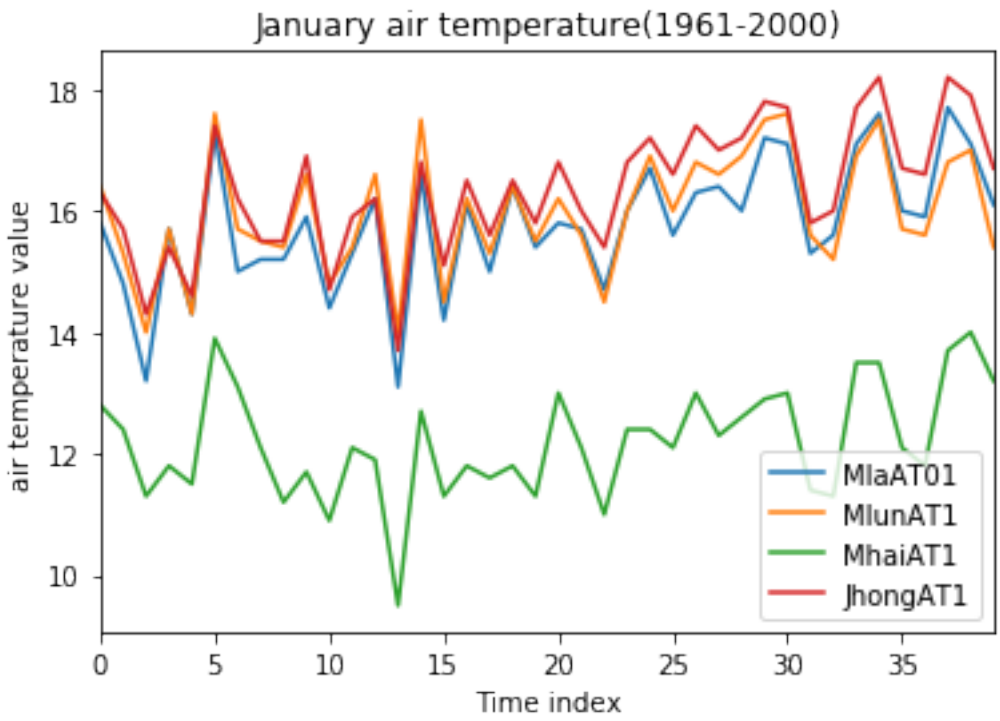
The .plot() method makes plots of pandas data very easy to create, and the .hist() method is useful for visualizing distributions of data.

```
In [34]: dfR1.plot(title='January rainfall(1961-2000)')
         plt.xlabel('Time index')
         plt.ylabel('rainfall value')

Out[34]: <matplotlib.text.Text at 0x89cc510>
```



The picture indicates that there are four peaks of rainfall from 1961 to 2000.

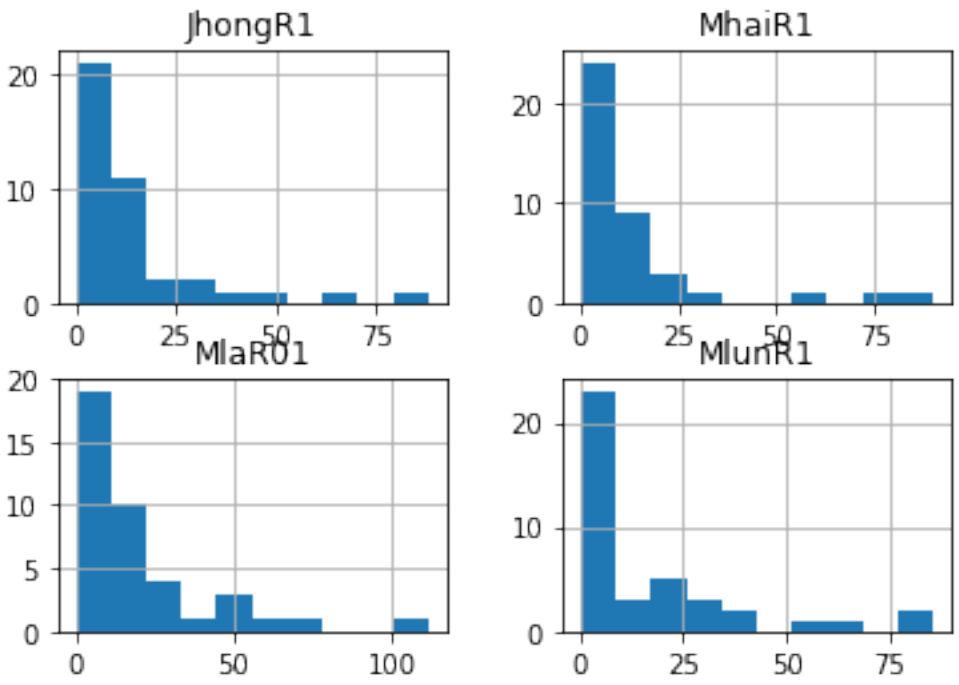
```
In [35]: dfAT1.plot(title='January air temperature(1961-2000)')
         plt.xlabel('Time index')
         plt.ylabel('air temperature value')

Out[35]: <matplotlib.text.Text at 0x89b2df0>
```



The picture indicates that January air temperature values in Mhai are lower than the others within 40 years, while the temperature values of the other three counties are close.

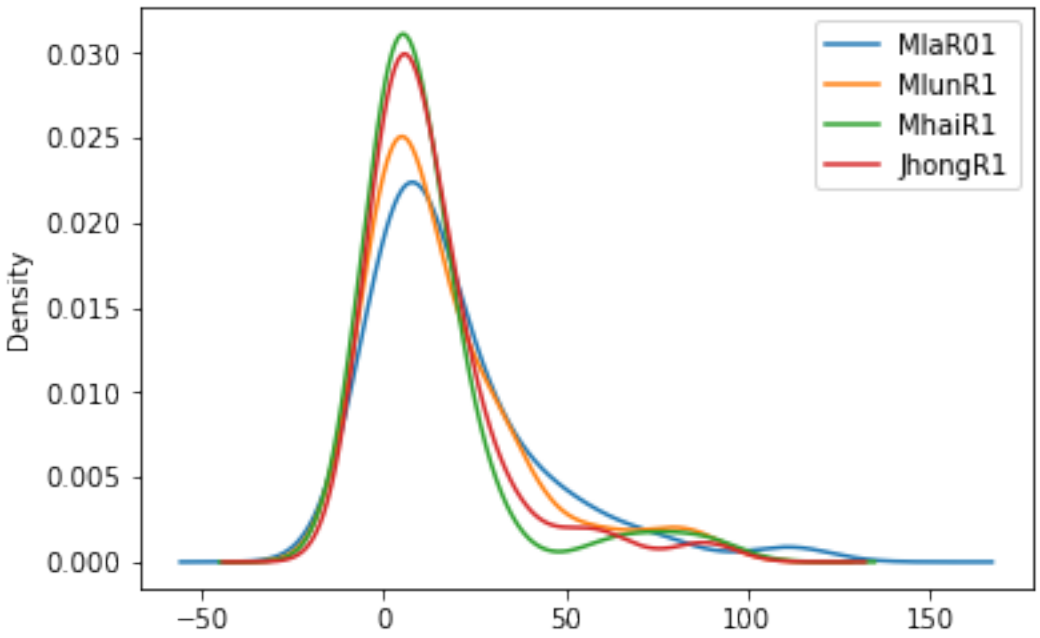
If the data has multiple series, the histogram function will automatically generate multiple histograma, one for each series:

```
In [36]: dfR1.hist()

Out[36]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x08AAC0B0>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x08AFAE90>],
                [<matplotlib.axes._subplots.AxesSubplot object at 0x08B3AF70>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x08B716F0>]], dt
```




```
In [37]: # using the kind='kde' parameter to create kernel density estimation plots
         dfR1.plot(kind='kde')
```

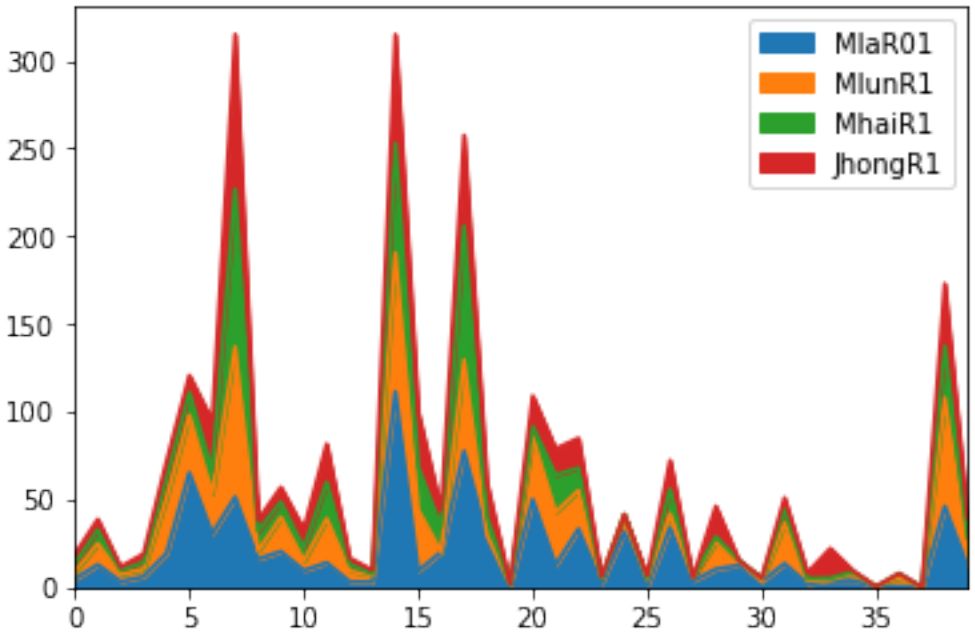
```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x8bd95d0>
```



A kernel density estimate plot makes an attempt and estimates the true distribution of the data, and hence smoothes it into a continuous plot.

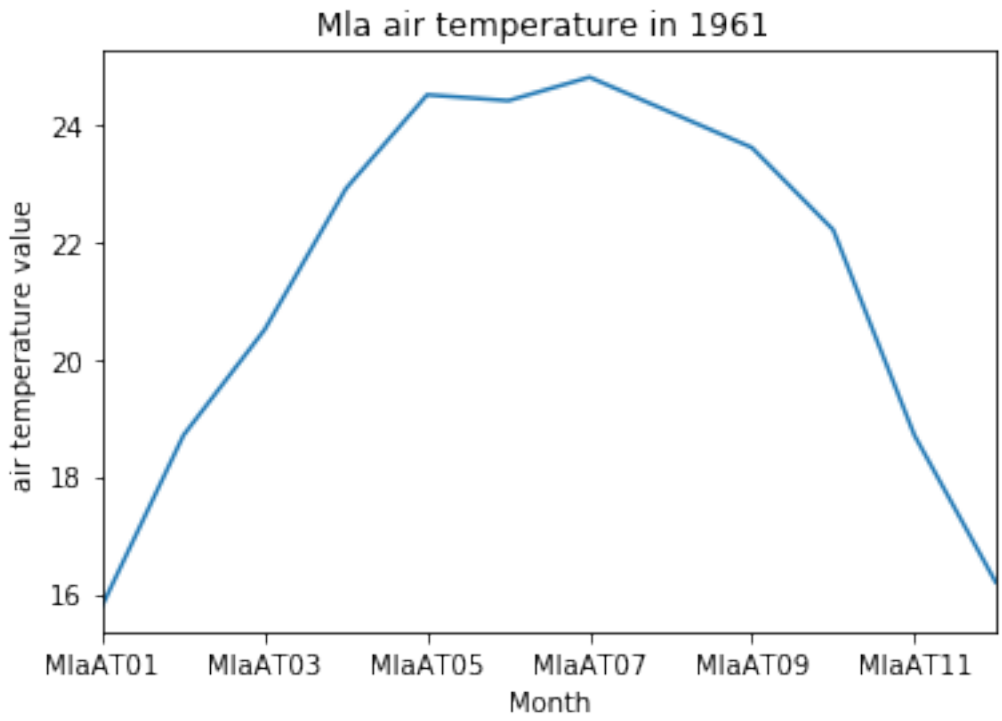
```
In [38]: # area plots are generated by specifying kind='area'
         dfR1.plot(kind='area')
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x9314dd0>
```




```
In [40]: # plotting a DataFrame with a single row
         MlaAT1961.ix[0].plot(title='Mla air temperature in 1961 ')
         plt.xlabel('Month')
         plt.ylabel('air temperature value')
```

```
Out[40]: <matplotlib.text.Text at 0x93ef3f0>
```



The picture indicates that Mla temperature in 1961 has began to fall continously from July.

In [ ]:

In [ ]: