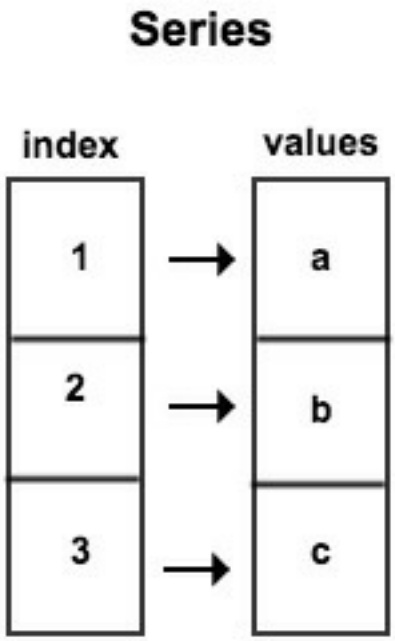
数据分析工具 Pandas 介绍

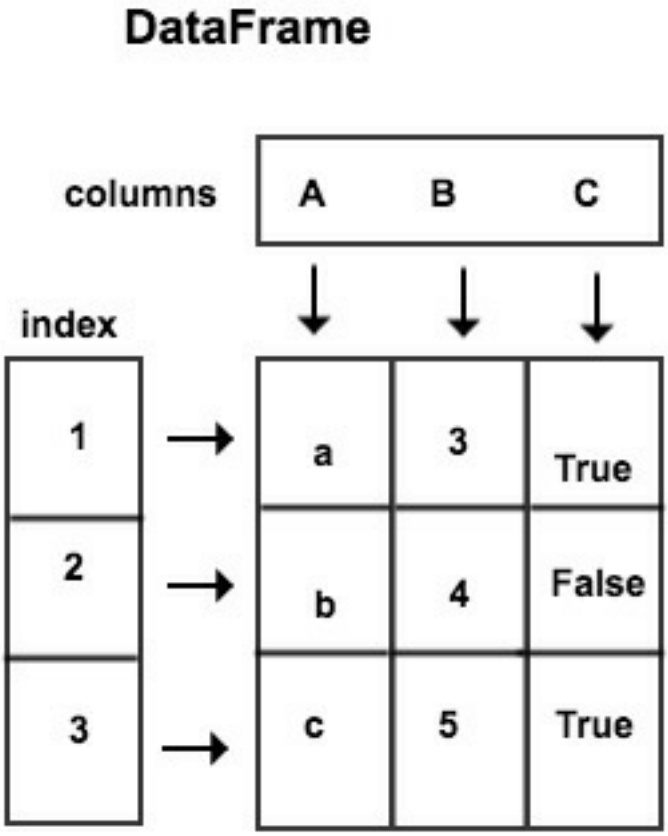
什么是Pandas

Pandas 是做数据分析的基础包，提供了灵活的数据结构和其它方便进行向量化计算的工具和函数, 使得 Python 也能够像 R 语言一样方便地用于数据分析和处理。在 Pandas 中有两种常见数据结构，分别是 Series 和 DataFrame。

Series 是一种增强型的一维数组，与 Python 中的列表相似，由 index（索引）和 values（值）组成， Series 中的值是相同的数据类型。



而 DataFrame 是增强型的二维数组，就像 Excel 中的表格，有行标签和列表索引，这种数据结构在Pandas 中最为常用。



在做数据分析前，我们会约定俗成地引入 Numpy 、Pandas、Matplotlib 三个工具包，并使用其简称 np，pd，plt。numpy 是科学计算基础包，pandas 依赖于 numpy，而 matplitlib 是绘图工具。(以下代码均在 IPython 中完成，如果你已经成功安装了 Anoconda，那么可以直接运行 ipython 命令进入)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Series

用列表可创建Series对象

```
In [30]: s = pd.Series(['a','b','c'])

In [31]: s
Out[31]:
0    a
1    b
2    c
dtype: object
```

Series 和列表一样每个元素有对应的索引，默认是0到n（n是列表的长度），也手动指定索引名字

```
In [42]: s = pd.Series(['a','b','c'], index=
['x','y','z'])

In [43]: s
Out[43]:
x    a
y    b
z    c
dtype: object
```

可以通过索引获取元素

```
In [50]: s['x']
Out[50]: 'a'
```

像列表一样，支持切片

```
In [51]: s[:2]
Out[51]:
x    a
y    b
dtype: object
```

可以使用字典创建Series（Series也可以看做是一个特殊的字典对象，都是索引到值的映射）

```
In [272]: s2 = pd.Series({1:"a",2:"b",3:"c"})

In [273]: s2
Out[273]:
1    a
2    b
3    c
dtype: object
```

DataFrame

有很多方法可以创建 DataFrame 对象，可以通过用相等长度的列表组成的字典对象来构建 DataFrame

```
In [52]: data = {'state': ['Ohio', 'Ohio',
'Ohio', 'Nevada', 'Nevada'],
    ...:         'year': [2000, 2001, 2002, 2001,
2002],
    ...:         'pop': [1.5, 1.7, 3.6, 2.4,
2.9]}
    ...:


In [54]: df = pd.DataFrame(data)

In [55]: df
Out[55]:
   pop    state  year
0  1.5     Ohio  2000
1  1.7     Ohio  2001
2  3.6     Ohio  2002
3  2.4   Nevada  2001
4  2.9   Nevada  2002
```

也可以通过 Numpy 的二维数组来构建 DataFrame

```
# 随机生成6行4列的二维数组
In [58]:  df = pd.DataFrame(np.random.randn(6,4))

In [59]: df
Out[59]:
          0         1         2         3
0  0.447964 -0.486327 -1.593023 -0.314114
1  1.004132 -0.058186  0.076479  0.076231
2  0.445284  0.592718  0.214101 -0.322876
3 -0.006924 -0.738673  0.277461  0.448946
4  0.100352  1.416282  0.353527  0.640276
5  0.804352 -0.374634  0.734836  0.247061
```

还可以从 csv 文件、数据库中获取，现在先来熟悉 DataFrame 中常用属性和操作方法，以便后续能够灵活运用 Pandas。

DataFrame 既有行索引（index）也有列索引（columns），构建 DataFrame 时可以指定每行的名字和每列的名字，例如下面的 DataFrame 用时间作为行索引，字母 A、B、C、D 作为列索引。

```
In [61]: dates =
pd.date_range('20130101',periods=6)

In [62]: dates
Out[62]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-
01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [63]: df =
pd.DataFrame(np.random.randn(6,4),index=dates,col
umns=list('ABCD'))

In [64]: df
Out[64]:
                   A         B         C
D
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
2013-01-06  0.888156  1.932291  0.998568
0.143846
```

DataFrame 其实就是由3部分组成的，分别是 index、columns、values

```
In [79]: df.columns
Out[79]: Index(['A', 'B', 'C', 'D'],
dtype='object')

In [80]: df.index
Out[80]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-
01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [81]: df.values
Out[81]:
array([[ 0.51328621, -1.475824  ,  1.93987575,
-0.16394233],
       [-0.51829132,  1.34522999,  0.51074601,
1.2847668 ],
       [-0.43486491, -0.46422712,  1.83025914,
-0.71928957],
       [ 0.65441841, -0.99424111,  0.16270488,
2.81662335],
       [ 1.54027403, -0.22712424,  1.84340078,
-2.97787999],
       [ 0.88815632,  1.93229088,  0.99856774,
0.14384553]])
```

head()

head() 返回 DataFrame 的头部数据（默认返回表格中的前5行数据），也可以指定返回的行数

```
In [70]: df.head(3)
Out[70]:
                   A          B          C
D
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
```

tail()

tail() 返回 DataFrame 的尾部数据（默认返回表格中的最后5行数据）

```
In [72]: df.tail()
Out[72]:
                   A          B          C
D
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
2013-01-06  0.888156  1.932291  0.998568
0.143846
```

按索引排序

```
# 按照列索引的降序排列：D->C->B->A
In [96]: df.sort_index(axis=1, ascending=False)
Out[96]:
                   D         C         B
A
2013-01-01 -0.163942  1.939876 -1.475824
0.513286
2013-01-02  1.284767  0.510746  1.345230
-0.518291
2013-01-03 -0.719290  1.830259 -0.464227
-0.434865
2013-01-04  2.816623  0.162705 -0.994241
0.654418
2013-01-05 -2.977880  1.843401 -0.227124
1.540274
2013-01-06  0.143846  0.998568  1.932291
0.888156

# 按照行索引的降序排列：2012-01-06->...->2013-01-01
In [97]: df.sort_index(axis=0, ascending=False)
Out[97]:
                   A         B         C
D
2013-01-06  0.888156  1.932291  0.998568
0.143846
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
2013-01-02 -0.518291  1.345230  0.510746
1.284767
```

```
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
```

按值排序

```
# 根据B列的值的升序排列
In [99]: df.sort_values(by='B')
Out[99]:
                   A         B         C
D
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-06  0.888156  1.932291  0.998568
0.143846

# 先按A的升序排，再按B的降序排
In [161]: df.sort_values(by=['A','B'], ascending=
[True, False])
Out[161]:
                   A         B         C
D
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
```

```
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-06  0.888156  1.932291  0.998568
0.143846
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
```

选择数据

```
# 选择一列，返回 Series 对象
In [100]: df['A']
Out[100]:
2013-01-01    0.513286
2013-01-02   -0.518291
2013-01-03   -0.434865
2013-01-04    0.654418
2013-01-05    1.540274
2013-01-06    0.888156
Freq: D, Name: A, dtype: float64

# 选择多列，返回 DataFrame 对象
In [102]: df[['A','B']]
Out[102]:
                   A         B
2013-01-01  0.513286 -1.475824
2013-01-02 -0.518291  1.345230
2013-01-03 -0.434865 -0.464227
2013-01-04  0.654418 -0.994241
2013-01-05  1.540274 -0.227124
2013-01-06  0.888156  1.932291
```

切片操作

```
In [101]: df[0:3]
Out[101]:
                   A         B         C
D
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-02 -0.518291  1.345230  0.510746
1.284767
2013-01-03 -0.434865 -0.464227  1.830259
-0.719290
```

通过loc、.iloc 高效获取数据

```
# 通过行索引切片获取指定列数据
In [145]: df.loc["2013-01-01":"2013-01-03",
['A','B']]
Out[145]:
                   A         B
2013-01-01  0.513286 -1.475824
2013-01-02 -0.518291  1.345230
2013-01-03 -0.434865 -0.464227

# 通过行号切片获取（1到4行，0到2列）数据
In [149]: df.iloc[1:4, 0:2]
Out[149]:
                   A         B
2013-01-02 -0.518291  1.345230
2013-01-03 -0.434865 -0.464227
2013-01-04  0.654418 -0.994241
```

通过条件过滤数据

```
In [104]: df[df.A>0]
Out[104]:
                   A         B         C
D
2013-01-01  0.513286 -1.475824  1.939876
-0.163942
2013-01-04  0.654418 -0.994241  0.162705
2.816623
2013-01-05  1.540274 -0.227124  1.843401
-2.977880
2013-01-06  0.888156  1.932291  0.998568
0.143846
```

求和

```
In [115]: df.sum()
Out[115]:
A    2.642979
B    0.116104
C    7.285554
D    0.384124
```

求平均值

```
In [121]: df.mean()
Out[121]:
A    0.440496
B    0.019351
C    1.214259
D    0.064021
dtype: float64
```

求最大/小值

```
In [157]: df.max()
Out[157]:
A    1.540274
B    1.932291
C    1.939876
D    2.816623

In [159]: df.min()
Out[159]:
A   -0.518291
B   -1.475824
C    0.162705
D   -2.977880
dtype: float64
```

分组 groupby

```
In [252]: df.groupby('A').size()
Out[252]:
A
-0.518291    1
-0.434865    1
 0.513286    1
 0.654418    1
 0.888156    1
 1.000000    1
 1.540274    1
```

groupby 的参数还可以是函数

```
# 添加E为时间列，根据时间的年进行分组
In [255]: df['E'] = df.index

In [256]: df
Out[256]:
                                A          B          C
D           E
2013-01-01 00:00:00  0.513286 -1.475824  1.939876
-0.163942 2013-01-01
2013-01-02 00:00:00 -0.518291  1.345230  0.510746
1.284767 2013-01-02
2013-01-03 00:00:00 -0.434865 -0.464227  1.830259
-0.719290 2013-01-03
2013-01-04 00:00:00  0.654418 -0.994241  0.162705
2.816623 2013-01-04
2013-01-05 00:00:00  1.540274 -0.227124  1.843401
-2.977880 2013-01-05
2013-01-06 00:00:00  0.888156  1.932291  0.998568
0.143846 2013-01-06
2014-01-06 00:00:00  1.000000  1.000000  1.000000
1.000000 2014-01-06

In [257]: df.groupby(lambda x :
df.E[x].year).size()
Out[257]:
2013    6
2014    1
dtype: int64
```

关于Pandas更多详细的用法可以参考Pandas官方文档[https://pandas.pydata.org (https://pandas.pydata.org)](https://pandas.pydata.org)，下一节我们将正式进入数据分析环节。