数据分析

数据分析是一门强大的科学，整天和数据打交道的人要么觉得枯燥要么觉得有趣，完全看个人性格。

本人接触数据分析是工作使然，为了分析各种不同的数据并对数据进行各种处理(比如说分组，切片，采样等)后交给相应的开发人员(拿到分析好的数据可以画图展示或者做其他相关事宜)。

python的数据分析相关有三个库：numpy、pandas以及matplotlib。其中，平常使用最多的还是pandas, 用了大概一周的pandas之后感觉这个库还是挺强大的，基本可以满足日常的开发需求。numpy相对pandas来说重要性就没那么高了，但是它是三个库中最基础的，所以也得掌握一下，只不过在学习三者的时间分配上来说pandas>numpy>matplotlib(本人在工作中很少用matplotlib画图)。

numpy

python中的list其实并不是像C、golang中的数组，list中的元素类型可以不一样，python中的numpy数据类型就和其他语言的数组类似，只能存放一种数据类型。并且，list并不能直接用来做数据操作(可以迭代去做)，但是numpy却有很多数学操作可以使用。下面从几个方面去了解一下numpy

- 创建
- 属性
- 基本操作
  - 索引
  - 切片
  - 变形
  - 级联
  - 切割
  - 排序
  - 运算
  - 索引

创建

1. 使用np.array()创建

In [5]:

```
import numpy as np
```

- 一维数组创建

In [6]:

```
np.array([1,2,3])
```

Out[6]:

```
array([1, 2, 3])
```

- 二维数组创建

In [7]:

```
np.array([[1,2,3], [4,5,6]])
```

Out[7]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [8]:

```
np.array([1, 1.2, 'a'])
```

Out[8]:

```
array(['1', '1.2', 'a'], dtype='<U32')
```

注意：

- numpy默认ndarray的所有元素的类型是相同的
- 如果传进来的列表中包含不同的类型，则统一为同一类型，优先级：str>float>int

2. 使用内置的方法创建

使用内置的方法去创建numpy还是挺常用的，一般不会去手动写一个列表里再套列表的数据，这样太麻烦了。

- np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None) 等差数列,其中num表示要创建几个元素，内部会计算差值

In [9]:

```
np.linspace(1, 30, num=5)
```

Out[9]:

```
array([ 1.  ,  8.25, 15.5 , 22.75, 30.  ])
```

- np.arange([start,]stop, [step,]dtype=None) 和python的range一样，左闭右开。和np.linspace的区别是一个是知道要创建的数据总数，一个是知道要创建的数据间的差值，相同点是两者都只能创建一维数组

In [10]:

```
np.arange(10, 30, 2)
```

Out[10]:

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28])
```

- np.random.randint(low, high=None, size=None, dtype='l') 这里的size和shape是一个意思，表示要创建的数组的形状，这个方法用来创建size形状的数组，数组的元素是从low到high的随机值。多次调用这个方法得到的随机值肯定是不同的，可以使用np.random.seed方法使得在同一次程序调用中多次调用np.random.randint得到相同的随机值。

In [11]:

```
np.random.randint(1, 20, size=(3, 4))
```

Out[11]:

```
array([[ 6,  8, 16, 13],
       [19, 19,  2, 15],
       [ 1, 10,  4,  6]])
```

In [12]:

```
np.random.seed(2)
np.random.randint(1, 20, size=(3,4))
```

Out[12]:

```
array([[16,  6,  8,  4],
       [ 7,  5, 11, 12],
       [ 8,  7, 11,  2]])
```

- np.random.random(size=None) 生成0到1的随机数，左闭右开

In [14]:

```
np.random.random(size=(2,3))
```

Out[14]:

```
array([[0.57730807, 0.16782331, 0.36747137],
       [0.46867358, 0.65426618, 0.79308974]])
```

至于np.zeros，np.ones这些方法就没啥好说的，本人目前常用的也就以上几种。

属性

array的属性：shape(形状), size(大小), dtype(元素类型), ndim(维度)

In [15]:

```
arr = np.arange(5)
```

In [16]:

```
arr.shape
```

Out[16]:

```
(5,)
```

In [17]:

```
arr.size
```

Out[17]:

```
5
```

In [18]:

```
arr.dtype
```

Out[18]:

```
dtype('int64')
```

In [19]:

```
arr.ndim
```

Out[19]:

```
1
```

基本操作

1. 索引

In [20]:

```
arr = np.random.randint(1, 20, size=(5, 6))
```

In [21]:

```
arr
```

Out[21]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 1, 19,  9, 13, 11,  9],
       [ 3, 10,  6,  7,  7, 19],
       [ 2,  5,  9, 18,  7,  6],
       [ 2, 19, 12,  6, 11,  9]])
```

- 默认索引为行索引

In [22]:

```
arr[1]
```

Out[22]:

```
array([ 1, 19,  9, 13, 11,  9])
```

In [23]:

```
# 取第一行和第三行数据，中括号里再套中括号的形式
arr[[0, 2]]
```

Out[23]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 3, 10,  6,  7,  7, 19]])
```

- 列索引(使用最原始的逗号)

In [25]:

```
arr[:,0]
```

Out[25]:

```
array([18,  1,  3,  2,  2])
```

In [26]:

```
arr[:,[0, 2]]
```

Out[26]:

```
array([[18, 10],
       [ 1,  9],
       [ 3,  6],
       [ 2,  9],
       [ 2, 12]])
```

2. 切片

In [27]:

```
arr
```

Out[27]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 1, 19,  9, 13, 11,  9],
       [ 3, 10,  6,  7,  7, 19],
       [ 2,  5,  9, 18,  7,  6],
       [ 2, 19, 12,  6, 11,  9]])
```

- 默认切片是行切片

In [28]:

```
arr[0:3]
```

Out[28]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 1, 19,  9, 13, 11,  9],
       [ 3, 10,  6,  7,  7, 19]])
```

In [29]:

```
arr[0:4:2]
```

Out[29]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 3, 10,  6,  7,  7, 19]])
```

- 列切片(同索引，使用最原始的逗号)

In [30]:

```
arr[:, 0:2]
```

Out[30]:

```
array([[18, 10],
       [ 1, 19],
       [ 3, 10],
       [ 2,  5],
       [ 2, 19]])
```

In [33]:

```
arr[:, 0:5:2]
```

Out[33]:

```
array([[18, 10,  1],
       [ 1,  9, 11],
       [ 3,  6,  7],
       [ 2,  9,  7],
       [ 2, 12, 11]])
```

- 倒序

In [43]:

```
arr[::-1]
```

Out[43]:

```
array([[ 2, 19, 12,  6, 11,  9],
       [ 2,  5,  9, 18,  7,  6],
       [ 3, 10,  6,  7,  7, 19],
       [ 1, 19,  9, 13, 11,  9],
       [18, 10, 10, 15,  1, 12]])
```

In [44]:

```
arr[:, ::-1]
```

Out[44]:

```
array([[12,  1, 15, 10, 10, 18],
       [ 9, 11, 13,  9, 19,  1],
       [19,  7,  7,  6, 10,  3],
       [ 6,  7, 18,  9,  5,  2],
       [ 9, 11,  6, 12, 19,  2]])
```

3. 变形

所谓的变形指的是数组的维度发生变化，但是包含的数据个数不变. 使用arr.reshape()函数，注意参数是一个tuple

In [34]:

```
arr
```

Out[34]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 1, 19,  9, 13, 11,  9],
       [ 3, 10,  6,  7,  7, 19],
       [ 2,  5,  9, 18,  7,  6],
       [ 2, 19, 12,  6, 11,  9]])
```

In [35]:

```
# 2维变1维, 等价于arr.flatten()
arr.reshape((arr.size,))
```

Out[35]:

```
array([18, 10, 10, 15,  1, 12,  1, 19,  9, 13, 11,  9,  3, 10,  6,  7,
7,
       19,  2,  5,  9, 18,  7,  6,  2, 19, 12,  6, 11,  9])
```

In [38]:

```
# 1维变2维
arr2 = np.linspace(1, 13, num=12)
```

In [39]:

```
arr2
```

Out[39]:

```
array([ 1.        ,  2.09090909,  3.18181818,  4.27272727,  5.3636363
6,
        6.45454545,  7.54545455,  8.63636364,  9.72727273, 10.8181818
2,
       11.90909091, 13.        ])
```

In [41]:

```
arr2.reshape((3, 4))
```

Out[41]:

```
array([[ 1.        ,  2.09090909,  3.18181818,  4.27272727],
       [ 5.36363636,  6.45454545,  7.54545455,  8.63636364],
       [ 9.72727273, 10.81818182, 11.90909091, 13.        ]])
```

In [42]:

```
# (-1, 2) 中的2表示要变形为2列, -1表示让numpy帮我们去根据元素数目计算出要变形为多少行(无需人为计
arr2.reshape((-1, 2))
```

Out[42]:

```
array([[ 1.        ,  2.09090909],
       [ 3.18181818,  4.27272727],
       [ 5.36363636,  6.45454545],
       [ 7.54545455,  8.63636364],
       [ 9.72727273, 10.81818182],
       [11.90909091, 13.        ]])
```

4. 级联

np.concatenate((a1, a2, ...), axis=0, out=None)

1. 级联的参数是列表：一定要加中括号或小括号
2. 维度必须相同
3. 形状相符:在维度保持一致的前提下，如果进行横向（axis=1）级联，必须保证进行级联的数组行数保持一致。如果进行纵向（axis=0）级联，必须保证进行级联的数组列数保持一致。
4. 可通过axis参数改变级联的方向, 其中axis=0表示纵向, axis=1表示横向, 这是需要记忆的

In [45]:

```
arr1 = np.random.randint(1, 23, size=(2,3))
arr2 = np.random.randint(2, 13, size=(2,3))
```

In [46]:

```
np.concatenate((arr1, arr2), axis=0)
```

Out[46]:

```
array([[ 4,  1, 17],
       [22, 22, 12],
       [ 2, 10,  8],
       [12,  7,  3]])
```

In [47]:

```
np.concatenate((arr1, arr2), axis=1)
```

Out[47]:

```
array([[ 4,  1, 17,  2, 10,  8],
       [22, 22, 12, 12,  7,  3]])
```

5. 切割

np.split(ary, indices_or_sections, axis=0) 参数： ary:要切分的数组

indices_or_sections:如果是一个整数，就用该数平均切分，如果是一个数组，为沿轴切分的位置（左开右闭）

axis：沿着哪个维度进行切向，默认为0，横向切分。为1时，纵向切分。这里的axis的值理解刀切割所落的位置在哪个轴向。

与array_split的差别：split必须要均等分，否则会报错。array_split不会

In [49]:

```
arr3 = np.arange(16).reshape((4, 4))
```

In [50]:

```
arr3
```

Out[50]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [52]:

```
# 纵向切割平均为2份
np.split(arr3, 2, axis=1)
```

Out[52]:

```
[array([[ 0,  1],
        [ 4,  5],
        [ 8,  9],
        [12, 13]]), array([[ 2,  3],
        [ 6,  7],
        [10, 11],
        [14, 15]])]
```

In [63]:

```
# 3 表示 先数3个再一刀切
np.split(arr3, [3,])
```

Out[63]:

```
[array([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]]), array([[12, 13, 14, 15]])]
```

In [64]:

```
arr3
```

Out[64]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [65]:

```
# 先找2行切一刀，再找到第3行再切一刀
np.split(arr3, [2,3])
```

Out[65]:

```
[array([[0, 1, 2, 3],
        [4, 5, 6, 7]]), array([[ 8,  9, 10, 11]]), array([[12, 13, 14,
15]])]
```

6. 排序

np.sort()与ndarray.sort()都可以，但有区别：

np.sort()不改变输入 ndarray.sort()本地处理，不占用空间，但改变输入

In [66]:

```
arr
```

Out[66]:

```
array([[18, 10, 10, 15,  1, 12],
       [ 1, 19,  9, 13, 11,  9],
       [ 3, 10,  6,  7,  7, 19],
       [ 2,  5,  9, 18,  7,  6],
       [ 2, 19, 12,  6, 11,  9]])
```

In [67]:

```
np.sort(arr, axis=1)
```

Out[67]:

```
array([[ 1, 10, 10, 12, 15, 18],
       [ 1,  9,  9, 11, 13, 19],
       [ 3,  6,  7,  7, 10, 19],
       [ 2,  5,  6,  7,  9, 18],
       [ 2,  6,  9, 11, 12, 19]])
```

In [68]:

```
arr.sort(axis=0)
```

In [69]:

```
arr
```

Out[69]:

```
array([[ 1,  5,  6,  6,  1,  6],
       [ 2, 10,  9,  7,  7,  9],
       [ 2, 10,  9, 13,  7,  9],
       [ 3, 19, 10, 15, 11, 12],
       [18, 19, 12, 18, 11, 19]])
```

7. 运算

求和np.sum、最大最小值：np.max/ np.min、平均值：np.mean()等等，这个用到的时候再查即可。

8. 索引

布尔型索引是我在接触numpy的时候给了我很大兴奋感的一个功能，布尔索引实现的是通过列向量中的每个元素的布尔量数值对一个与列向量有着同样行数的矩阵进行符合匹配。而这样的作用，其实是把列向量中布尔量为True的相应行向量给抽取了出来, 这一点同样在pandas中适用而且很常用

如果进行变量或者标定量的大数据处理，这种筛选功能的使用肯定会给程序的设计带来极大的便捷。

In [70]:

```
a = np.arange(12).reshape(3,4)
```

In [71]:

```
a
```

Out[71]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [72]:

```
b1 = np.array([False,True,True])
```

In [73]:

```
b2 = np.array([True,False,True,False])
```

In [74]:

```
# 选择2，3行
a[b1]
```

Out[74]:

```
array([[ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [75]:

```
# 选择1，3列
a[:, b2]
```

Out[75]:

```
array([[ 0,  2],
       [ 4,  6],
       [ 8, 10]])
```

一维布尔数组的长度必须和你要切片的维(或axis)的长度相同。在上面的例子中，b1是长度为3的一维数组，b2是长度为4，适合索引数组a的第二维。

In [ ]:

```

```