

数据存取与函数

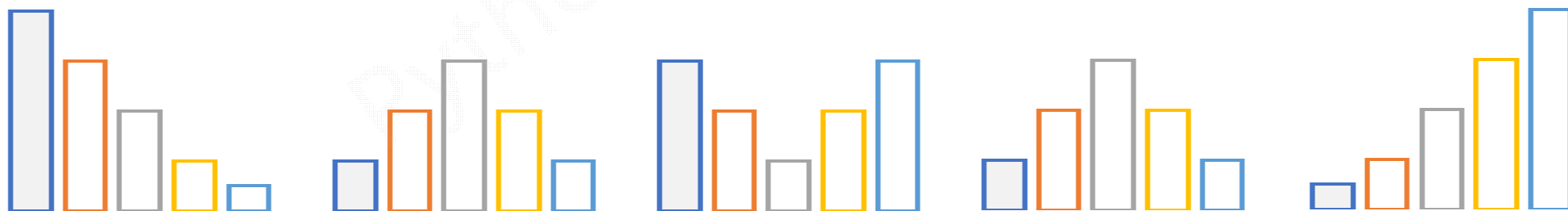
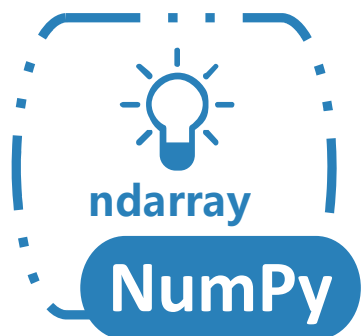


肖毅



Python数据分析与展示

掌握表示、清洗、统计和展示数据的能力





数据的CSV文件存取

CSV文件

CSV (Comma-Separated Value, 逗号分隔值)

CSV是一种常见的文件格式，用来存储批量数据

2016年7月部分大中城市新建住宅价格指数

城市	环比	同比	定基
北京	101.5	120.7	121.4
上海	101.2	127.3	127.8
广州	101.3	119.4	120.0
深圳	102.0	140.9	145.5
沈阳	100.1	101.4	101.6



城市,环比,同比,定基

北京,101.5,120.7,121.4

上海,101.2,127.3,127.8

广州,101.3,119.4,120.0

深圳,102.0,140.9,145.5

沈阳,100.1,101.4,101.6

CSV文件

```
np.savetxt(frame, array, fmt='%.18e', delimiter=None)
```

- **frame** : 文件、字符串或产生器，可以是.gz或.bz2的压缩文件
- **array** : 存入文件的数组
- **fmt** : 写入文件的格式，例如：%d %.2f %.18e
- **delimiter** : 分割字符串，默认是任何空格

CSV文件

```
In [91]: a = np.arange(100).reshape(5, 20)
```

```
In [92]: np.savetxt('a.csv', a, fmt='%d', delimiter=',')
```



```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19  
20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39  
40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59  
60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79  
80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99
```

CSV文件

```
In [93]: a = np.arange(100).reshape(5, 20)
```

```
In [94]: np.savetxt('a.csv', a, fmt='%.1f', delimiter=',')
```



```
0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0  
20.0,21.0,22.0,23.0,24.0,25.0,26.0,27.0,28.0,29.0,30.0,31.0,32.0,33.0,34.0,35.0,36.0,37.0,38.0,39.0  
40.0,41.0,42.0,43.0,44.0,45.0,46.0,47.0,48.0,49.0,50.0,51.0,52.0,53.0,54.0,55.0,56.0,57.0,58.0,59.0  
60.0,61.0,62.0,63.0,64.0,65.0,66.0,67.0,68.0,69.0,70.0,71.0,72.0,73.0,74.0,75.0,76.0,77.0,78.0,79.0  
80.0,81.0,82.0,83.0,84.0,85.0,86.0,87.0,88.0,89.0,90.0,91.0,92.0,93.0,94.0,95.0,96.0,97.0,98.0,99.0
```

CSV文件

```
np.loadtxt(frame, dtype=np.float, delimiter=None, unpack=False)
```

- `frame` : 文件、字符串或产生器，可以是.gz或.bz2的压缩文件
- `dtype` : 数据类型，可选
- `delimiter` : 分割字符串，默认是任何空格
- `unpack` : 如果True，读入属性将分别写入不同变量

CSV文件

```
In [97]: b = np.loadtxt('a.csv', delimiter=',')
```

```
In [98]: b
```

```
Out[98]:
```

```
array([[ 0.,  1.,  2., ..., 17., 18., 19.],  
       [20., 21., 22., ..., 37., 38., 39.],  
       [40., 41., 42., ..., 57., 58., 59.],  
       [60., 61., 62., ..., 77., 78., 79.],  
       [80., 81., 82., ..., 97., 98., 99.]])
```

```
In [95]: b = np.loadtxt('a.csv', dtype=np.int, delimiter=',')
```

```
In [96]: b
```

```
Out[96]:
```

```
array([[ 0,  1,  2, ..., 17, 18, 19],  
       [20, 21, 22, ..., 37, 38, 39],  
       [40, 41, 42, ..., 57, 58, 59],  
       [60, 61, 62, ..., 77, 78, 79],  
       [80, 81, 82, ..., 97, 98, 99]])
```

CSV文件的局限性

CSV只能有效存储一维和二维数组

`np.savetxt()` `np.loadtxt()` 只能有效存取一维和二维数组



多维数据的存取

任意维度数据如何存取呢？

```
a.tofile(frame, sep='', format='%s')
```

- `frame` : 文件、字符串
- `sep` : 数据分割字符串，如果是空串，写入文件为二进制
- `format` : 写入数据的格式

```
a.tofile(frame, sep=' ', format='%s')
```

```
In [125]: a = np.arange(100).reshape(5, 10, 2)
```

```
In [126]: a.tofile("b.dat", sep=",", format='%d')
```



```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70
,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99
```

a.tofile(frame, sep=' ', format='%s')

In [127]: a = np.arange(100).reshape(5, 10, 2)

In [128]: a.tofile("b.dat", format='%d')



NULNULNULNULSOHNULNULNULSTXNULNULNULETXNULNULNULEOTNULNULNULENONULNULNULACKNULNULNULBELNULNUL
NULBSNULNULNULNULNULNULNUL
NULNULNULVTNULNULNULFFNULNULNUL
NULNULNULSONULNULNULSINULNULNULDLNULNULNULDC1NULNULNULDC2NULNULNULDC3NULNULNULDC4NULNULNUL
NAKNULNULNULSYNNULNULNULETBNULNULNULCANULNULNULEMNULNULNULSUBNULNULNULESCNULNULNULFSNULNUL
NULGSNULNULNULRSNULNULNULUSNULNULNUL
NULNULNUL!NULNULNUL"NULNULNUL#NULNULNUL\$NULNULNUL%NULNULNUL&NULNULNUL'NULNULNUL(NULNULNUL)NUL
NULNUL*NULNULNUL+~NULNULNUL,NULNULNUL-NULNULNUL.NULNULNUL/NULNULNUL0NULNULNUL1NULNULNUL2NULNUL
NUL3NULNULNUL4NULNULNUL5NULNULNUL6NULNULNUL7NULNULNUL8NULNULNUL9NULNULNUL:NULNULNUL;NULNULNUL<
NULNULNUL=
NULNULNUL>NULNULNUL?NULNULNUL@NULNULNULANULNULNULBNULNULNULCNULNULNULdNULNULNULENUL
NULNULFNULNULNULGNULNULNULHNULNULNULINULNULNULJNULNULNULKNULNULNULLNULNULNULMNULNULNULNNULNUL
NULoNULNULNULPNULNULNULQNULNULNULRNULNULNULSNULNULNULTNULNULNULUNULNULNULVNULNULNULwNULNULNULx
NULNULNULYNULNULNULzNULNULNUL[NULNULNUL\NULNULNUL]NULNULNUL^NULNULNUL_NULNULNUL`NULNULNULaNUL
NULNULbNULNULNULcNULNULNUL

```
np.fromfile(frame, dtype=float, count=-1, sep='')
```

- **frame** : 文件、字符串
- **dtype** : 读取的数据类型
- **count** : 读入元素个数，-1表示读入整个文件
- **sep** : 数据分割字符串，如果是空串，写入文件为二进制

```
np.fromfile(frame, dtype=float, count=-1, sep='')
```

```
In [129]: a = np.arange(100).reshape(5, 10, 2)
```

```
In [130]: a.tofile("b.dat", sep=",", format='%d')
```

```
In [131]: c = np.fromfile("b.dat", dtype=np.int, sep=",")
```

```
In [132]: c
```

```
Out[132]: array([ 0,  1,  2, ..., 97, 98, 99])
```

```
In [133]: c = np.fromfile("b.dat", dtype=np.int, sep=",").reshape(5, 10, 2)
```

```
In [134]: c
```

```
Out[134]:
```

```
array([[[ 0,  1],  
        [ 2,  3],  
        [ 4,  5],  
        ...,  
        [14, 15],  
        [16, 17],  
        [18, 19]],
```



```
np.fromfile(frame, dtype=float, count=-1, sep='')
```

```
In [135]: a = np.arange(100).reshape(5, 10, 2)
```

```
In [136]: a.tofile("b.dat", format='%d')
```

```
In [137]: c = np.fromfile("b.dat", dtype=np.int).reshape(5, 10, 2)
```

```
In [138]: c
```

```
Out[138]:
```

```
array([[[ 0,  1],  
        [ 2,  3],  
        [ 4,  5],  
        ...,  
        [14, 15],  
        [16, 17],  
        [18, 19]],
```

需要注意

该方法需要读取时知道存入文件时数组的维度和元素类型

`a.tofile()`和`np.fromfile()`需要配合使用

可以通过元数据文件来存储额外信息

NumPy的便捷文件存取

`np.save(fname, array)` 或 `np.savez(fname, array)`

- `fname` : 文件名, 以.npy为扩展名, 压缩扩展名为.npz
- `array` : 数组变量

`np.load(fname)`

- `fname` : 文件名, 以.npy为扩展名, 压缩扩展名为.npz

NumPy的便捷文件存取

```
In [148]: a = np.arange(100).reshape(5, 10, 2)
```

```
In [149]: np.save("a.npy", a)
```

```
In [150]: b = np.load("a.npy")
```

In [151]: b

Out[151]:

```
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       ...,
       [14, 15],
       [16, 17],
       [18, 19]])
```



```

揔UMPYSOHNULFNU{'descr': '<i4', 'fortran_order': False, 'shape': (5, 10, 2), }
NULNULNULNULSOHNULNULNULSTXNULNULNULETXNULNULNULEOTNULNULNULENONNULNULNULACKNULNULNULBELNULNUL
NULBSNULNULNULNULNULNULNUL
NULNULNULVTNULNULNULFFNULNULNUL
NULNULNULSONULNULNULSTNULNULNULDLNULNULNULDC1NULNULNULDC2NULNULNULDC3NULNULNULDC4NULNULNUL
NAKNULNULNULSYNNULNULNULETBNULNULNULCANNULNULNULEMNULNULNULSUBNULNULNULESCNULNULNULEFSNULNUL
NULGSNULNULNULRSNULNULNULUSNULNULNUL
NULNULNUL!NULNULNUL"NULNULNUL#NULNULNUL$NULNULNUL%NULNULNUL&NULNULNUL'NULNULNUL(NULNULNUL)NUL
NULNUL*NULNULNUL+NULNULNUL,NULNULNUL-NULNULNUL.NULNULNUL/NULNULNULoNULNULNUL1NULNULNUL2NULNUL
NUL3NULNULNUL4NULNULNUL5NULNULNUL6NULNULNUL7NULNULNUL8NULNULNUL9NULNULNUL:NULNULNUL;NULNULNUL<
NULNULNUL= NULNULNUL>NULNULNUL?NULNULNUL@NULNULNULANULNULNULBNULNULNULCNULNULNULDNULNULNULENUL
NULNULFNULNULNULGNULNULNULHNULNULNULINULNULNULJNULNULNULKNULNULNULLNULNULNULMNULNULNULNNULNUL
NULoNULNULNULPNULNULNULQNULNULNULRNULNULNULSNULNULNULTNULNULNULUNULNULNULVNULNULNULWNULNULNULX
NULoNULNULYNULNULNULzNULNULNUL[NULNULNUL\NULNULNUL]NULNULNUL^NULNULNUL_NULNULNUL`NULNULNULaNUL
NULNULbNULNULNULcNULNULNUL

```



NumPy的随机数函数

NumPy的随机数函数子库

NumPy的random子库

`np.random.*`

`np.random.rand()`

`np.random.randn()`

`np.random.randint()`

np.random的随机数函数(1)

函数	说明
<code>rand(d0,d1,...,dn)</code>	根据d0-dn创建随机数数组，浮点数， $[0,1)$ ，均匀分布
<code>randn(d0,d1,...,dn)</code>	根据d0-dn创建随机数数组，标准正态分布
<code>randint(low[,high,shape])</code>	根据shape创建随机整数或整数数组，范围是 $[low, high)$
<code>seed(s)</code>	随机数种子，s是给定的种子值

函数小试

```
In [5]: import numpy as np
```

```
In [6]: a = np.random.rand(3, 4, 5)
```

```
In [7]: a
```

```
Out[7]:
```

```
array([[[ 0.93859387,  0.01070518,  0.3054871 ,  0.1932416 ,  0.82375036],
        [ 0.15110071,  0.34448139,  0.21612265,  0.43276404,  0.73471971],
        [ 0.00407316,  0.70711519,  0.05127404,  0.67731786,  0.14322067],
        [ 0.88997925,  0.96002098,  0.33277737,  0.59770084,  0.57604945]],

       [[ 0.52441722,  0.14175617,  0.08588264,  0.62617497,  0.72711516],
        [ 0.63504074,  0.21290387,  0.77465841,  0.47369419,  0.78394602],
        [ 0.68891405,  0.3880887 ,  0.60886227,  0.50600248,  0.31468346],
        [ 0.34277096,  0.15791136,  0.14749979,  0.25235406,  0.03123494]],

       [[ 0.6084322 ,  0.51827266,  0.2855457 ,  0.92409508,  0.15750942],
        [ 0.00218532,  0.13749523,  0.73366243,  0.33392875,  0.31355293],
        [ 0.18500307,  0.16201531,  0.66444529,  0.34702364,  0.17776621],
        [ 0.0063955 ,  0.61556899,  0.93334567,  0.97117129,  0.85570959]]])
```

```
In [8]: sn = np.random.randn(3, 4, 5)
```

```
In [9]: sn
```

```
Out[9]:
```

```
array([[[ 0.80082098,  0.81195729, -2.02593352, -0.43980132, -0.84428853],
        [ 0.40876378,  1.0471804 , -0.86464492,  0.20691501, -0.32795862],
        [-0.72705744, -0.95187079, -0.79302798,  1.38466351, -0.64609614],
        [ 0.41632154,  0.08465048, -0.14857112, -0.93631024,  0.13784205]],

       [[-1.08984876,  0.12145932,  1.07303554, -1.95695301, -1.02073863],
        [ 0.60733033,  0.9238731 , -2.17872264, -0.17299356,  0.37281589],
        [-0.09743098, -0.17689769,  0.29176532, -1.282326 , -0.67244174],
        [-0.62478187,  0.24169548, -0.98143442, -0.56389493,  1.15780524]],

       [[ 1.47318384,  0.96500145,  0.80124135, -0.85210635,  0.25740846],
        [ 1.10173455,  0.65815256,  0.10804733, -1.02128938, -0.52060164],
        [-0.86936026,  0.11153553, -0.30380139, -0.36047905,  0.35520664],
        [ 1.27520826,  1.0598634 , -1.5327553 , -0.55896011,  1.92053426]]])
```

```
In [10]: b = np.random.randint(100, 200, (3,4))
```

```
In [11]: b
```

```
Out[11]:
```

```
array([[195, 192, 173, 161],
       [104, 190, 110, 126],
       [198, 122, 191, 178]])
```


函数小试

```
In [10]: b = np.random.randint(100, 200, (3,4))
```

```
In [11]: b
```

```
Out[11]:
```

```
array([[195, 192, 173, 161],  
       [104, 190, 110, 126],  
       [198, 122, 191, 178]])
```

```
In [12]: np.random.seed(10)
```

```
In [13]: np.random.randint(100, 200, (3,4))
```

```
Out[13]:
```

```
array([[109, 115, 164, 128],  
       [189, 193, 129, 108],  
       [173, 100, 140, 136]])
```

```
In [14]: np.random.seed(10)
```

```
In [15]: np.random.randint(100, 200, (3,4))
```

```
Out[15]:
```

```
array([[109, 115, 164, 128],  
       [189, 193, 129, 108],  
       [173, 100, 140, 136]])
```

np.random的随机数函数(2)

函数	说明
<code>shuffle(a)</code>	根据数组a的第1轴进行随排列，改变数组x
<code>permutation(a)</code>	根据数组a的第1轴产生一个新的乱序数组，不改变数组x
<code>choice(a[,size,replace,p])</code>	从一维数组a中以概率p抽取元素，形成size形状新数组 replace表示是否可以重用元素，默认为False

函数小试

```
In [18]: import numpy as np
```

```
In [19]: a = np.random.randint(100, 200, (3,4))
```

```
In [20]: a
```

```
Out[20]:  
array([[116, 111, 154, 188],  
       [162, 133, 172, 178],  
       [149, 151, 154, 177]])
```

```
In [21]: np.random.shuffle(a)
```

```
In [22]: a
```

```
Out[22]:  
array([[116, 111, 154, 188],  
       [149, 151, 154, 177],  
       [162, 133, 172, 178]])
```

```
In [23]: np.random.shuffle(a)
```

```
In [24]: a
```

```
Out[24]:  
array([[162, 133, 172, 178],  
       [116, 111, 154, 188],  
       [149, 151, 154, 177]])
```

```
In [35]: import numpy as np
```

```
In [36]: a = np.random.randint(100, 200, (3,4))
```

```
In [37]: a
```

```
Out[37]:  
array([[117, 146, 107, 175],  
       [128, 133, 184, 196],  
       [188, 144, 105, 104]])
```

```
In [38]: np.random.permutation(a)
```

```
Out[38]:  
array([[128, 133, 184, 196],  
       [188, 144, 105, 104],  
       [117, 146, 107, 175]])
```

```
In [39]: a
```

```
Out[39]:  
array([[117, 146, 107, 175],  
       [128, 133, 184, 196],  
       [188, 144, 105, 104]])
```

a没有变化

函数小试

```
In [54]: import numpy as np
```

```
In [55]: b = np.random.randint(100, 200, (8,))
```

```
In [56]: b
```

```
Out[56]: array([193, 175, 186, 137, 111, 121, 133, 195])
```

```
In [57]: np.random.choice(b, (3,2))
```

```
Out[57]:  
array([[137, 193],  
       [193, 121],  
       [175, 193]])
```

```
In [58]: np.random.choice(b, (3,2), replace=False)
```

```
Out[58]:  
array([[111, 175],  
       [193, 195],  
       [186, 133]])
```

```
In [61]: np.random.choice(b, (3,2), p= b/np.sum(b))
```

```
Out[61]:  
array([[121, 175],  
       [193, 186],  
       [193, 175]])
```

np.random的随机数函数(3)

函数	说明
<code>uniform(low,high,size)</code>	产生具有均匀分布的数组,low起始值,high结束值,size形状
<code>normal(loc,scale,size)</code>	产生具有正态分布的数组,loc均值,scale标准差,size形状
<code>poisson(lam,size)</code>	产生具有泊松分布的数组,lam随机事件发生率,size形状

函数小试

```
In [11]: import numpy as np
```

```
In [12]: u = np.random.uniform(0, 10, (3,4))
```

```
In [13]: u
```

```
Out[13]:
```

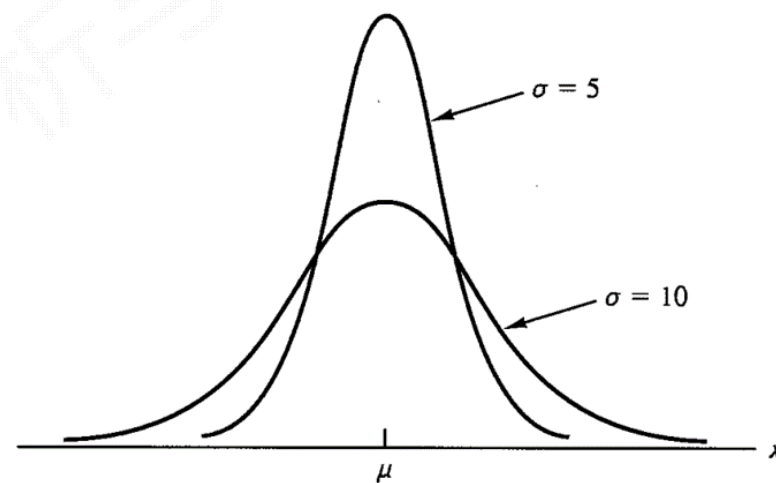
```
array([[ 2.05586231,  2.17210729,  8.23705995,  0.47651788],  
       [ 6.95560704,  6.79999967,  8.48924182,  4.39112863],  
       [ 4.25512913,  5.32043243,  5.28020392,  4.88650053]])
```

```
In [14]: n = np.random.normal(10, 5, (3,4))
```

```
In [15]: n
```

```
Out[15]:
```

```
array([[ 7.99574876,  5.02757645, 11.97332585,  5.48852456],  
       [12.45906894, 19.42856317, 19.11978806, 13.16322858],  
       [10.43019671,  8.46496502,  8.61987727, 15.6517832 ]])
```



正态分布



NumPy的统计函数

NumPy的统计函数

NumPy直接提供的统计类函数

`np.*`

`np.std()`

`np.var()`

`np.average()`

NumPy的统计函数(1)

函数	说明
<code>sum(a, axis=None)</code>	根据给定轴axis计算数组a相关元素之和，axis整数或元组
<code>mean(a, axis=None)</code>	根据给定轴axis计算数组a相关元素的期望，axis整数或元组
<code>average(a,axis=None,weights=None)</code>	根据给定轴axis计算数组a相关元素的加权平均值
<code>std(a, axis=None)</code>	根据给定轴axis计算数组a相关元素的标准差
<code>var(a, axis=None)</code>	根据给定轴axis计算数组a相关元素的方差

`axis=None` 是统计函数的标配参数

函数小试

```
In [20]: import numpy as np
```

```
In [21]: a = np.arange(15).reshape(3, 5)
```

```
In [22]: a
```

```
Out[22]:  
array([[ 0,  1,  2,  3,  4],  
       [ 5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14]])
```

```
In [23]: np.sum(a)
```

```
Out[23]: 105
```

```
In [24]: np.mean(a, axis=1)
```

```
Out[24]: array([ 2.,  7., 12.])
```

```
In [25]: np.mean(a, axis=0)
```

```
Out[25]: array([ 5.,  6.,  7.,  8.,  9.])
```

加权平均

```
In [26]: np.average(a, axis=0, weights=[10, 5, 1])
```

```
Out[26]: array([ 2.1875,  3.1875, 4.1875,  5.1875,  6.1875])
```

```
In [27]: np.std(a)
```

```
Out[27]: 4.3204937989385739
```

$$2*10+7*5+1*12/(10+5+1)=4.1875$$

```
In [28]: np.var(a)
```

```
Out[28]: 18.666666666666668
```

NumPy的统计函数(2)

函数	说明
<code>min(a)</code> <code>max(a)</code>	计算数组a中元素的最小值、最大值
<code>argmin(a)</code> <code>argmax(a)</code>	计算数组a中元素最小值、最大值的降一维后下标
<code>unravel_index(index, shape)</code>	根据shape将一维下标index转换成多维下标
<code>ptp(a)</code>	计算数组a中元素最大值与最小值的差
<code>median(a)</code>	计算数组a中元素的中位数（中值）

函数小试

```
In [34]: import numpy as np
```

```
In [35]: b = np.arange(15,0,-1).reshape(3, 5)
```

```
In [36]: b
```

```
Out[36]:
```

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

```
In [37]: np.max(b)
```

```
Out[37]: 15
```

```
In [38]: np.argmax(b)
```

```
Out[38]: 0
```

扁平化后的下标

```
In [39]: np.unravel_index(np.argmax(b), b.shape)
```

```
Out[39]: (0, 0)
```

重塑成多维下标

```
In [40]: np.ptp(b)
```

```
Out[40]: 14
```

```
In [41]: np.median(b)
```

```
Out[41]: 8.0
```



NumPy的梯度函数



NumPy的梯度函数

函数	说明
<code>np.gradient(f)</code>	计算数组f中元素的梯度，当f为多维时，返回每个维度梯度

梯度：连续值之间的变化率，即斜率

XY坐标轴连续三个X坐标对应的Y轴值：a，b，c，其中，b的梯度是： $(c-a)/2$

函数小试

```
In [49]: import numpy as np
```

```
In [50]: a = np.random.randint(0, 20, (5))
```

```
In [51]: a
```

```
Out[51]: array([15,  3, 12, 13, 14])
```

存在两侧值： $(12-15)/2$

```
In [52]: np.gradient(a)
```

```
Out[52]: array([-12. , -1.5,  5. ,  1. ,  1. ])
```

→ 只有一侧值： $(14-13)/1$

```
In [53]: b = np.random.randint(0, 20, (5))
```

```
In [54]: b
```

```
Out[54]: array([5, 7, 6, 1, 9])
```

```
In [55]: np.gradient(b)
```

```
Out[55]: array([ 2. ,  0.5, -3. ,  1.5,  8. ])
```

函数小试

```
In [56]: import numpy as np
```

```
In [57]: c = np.random.randint(0, 50, (3, 5))
```

```
In [58]: c
```

```
Out[58]:
```

```
array([[18, 49,  1,  5, 26],
       [40, 38, 39, 46, 47],
       [46, 23, 16, 31, 36]])
```

```
In [59]: np.gradient(c)
```

```
Out[59]:
```

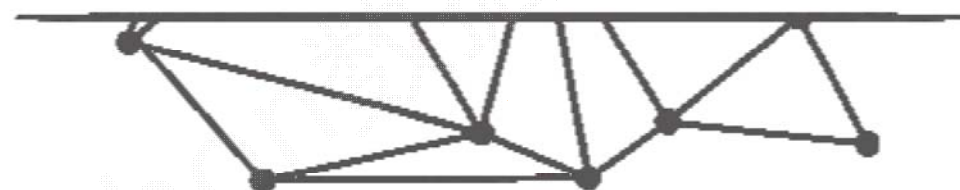
```
[array([[ 22. , -11. ,  38. ,  41. ,  21. ],
       [ 14. , -13. ,   7.5,  13. ,   5. ],
       [  6. , -15. , -23. , -15. , -11. ]]),
 array([[ 31. , -8.5, -22. , 12.5,  21. ],
       [ -2. , -0.5,  4. ,  4. ,  1. ],
       [-23. , -15. ,  4. , 10. ,  5. ]])]
```

最外层维度的梯度

第二层维度的梯度



单元小结



数据存取与函数

CSV文件

`np.loadtxt()`

`np.savetxt()`

多维数据存取

`a.tofile()` `np.fromfile()`

`np.save()` `np.savez()` `np.load()`

随机函数

`np.random.rand()`

`np.random.randn()`

`np.random.randint()`

`np.random.seed()`

`np.random.shuffle()`

`np.random.permutation()`

`np.random.choice()`

数据存取与函数

NumPy的统计函数

`np.sum()`

`np.mean()`

`np.average()`

`np.std()`

`np.var()`

`np.median()`

`np.min()`

`np.max()`

`np.argmin()`

`np.argmax()`

`np.unravel_index()`

`np.ptp()`

NumPy的梯度函数

`np.gradient()`