

统计函数

NumPy 能方便地求出统计学常见的描述性统计量。

```
1 import numpy as np
```

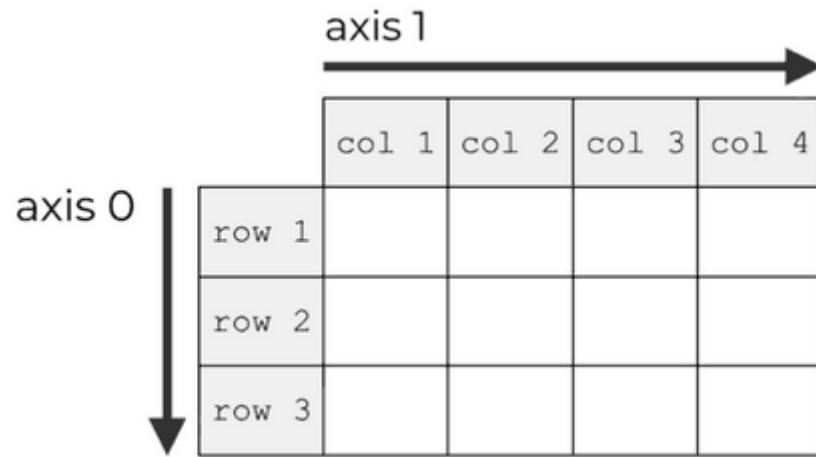
求平均值 `mean()`

```
1 m1 = np.arange(20).reshape((4,5))
2 print(m1)
3 # 默认求出数组所有元素的平均值
4 m1.mean()
5 np.mean(m1)
```

```
1 [[ 0  1  2  3  4]
2  [ 5  6  7  8  9]
3  [10 11 12 13 14]
4  [15 16 17 18 19]]
```

```
1 9.5
```

若想求某一维度的平均值，设置 **axis** 参数，多维数组的元素指定



- axis = 0, 将从上往下计算
- axis = 1, 将从左往右计算

```
1 m1 = np.arange(20).reshape((4,5))
2 print(m1)
3 # axis=0 将从上往下计算平均值
```

```
1 [[ 0  1  2  3  4]
2  [ 5  6  7  8  9]
3  [10 11 12 13 14]
4  [15 16 17 18 19]]
```

```
1 m1.mean(axis=0)
```

```
1 array([ 7.5,  8.5,  9.5, 10.5, 11.5])
```

```
1 # axis=1 将从左往右计算平均值
2 m1.mean(axis=1)
```

```
1 array([ 2.,  7., 12., 17.])
```

中位数 np.median

又称中点数，中值

是按顺序排列的一组数据中居于中间位置的数，代表一个样本、种群或概率分布中的一个数值

- 平均数：是一个"虚拟"的数，是通过计算得到的，它不是数据中的原始数据。
中位数：是一个不完全"虚拟"的数。
- 平均数：反映了一组数据的平均大小，常用来一代表数据的总体 "平均水平"。
中位数：像一条分界线，将数据分成前半部分和后半部分，因此用来代表一组数据的"中等水平"

```
1 ar1 = np.array([1,3,5,6,8])
2
3 np.median(ar1)
```

```
1 4.6
```

```
1 5.0
```

```
1 ar1 = np.array([1,3,5,6,8,9])
2 np.median(ar1)
```

```
1 5.5
```

```
1
```

求标准差 `ndarray.std`

在概率统计中最常使用作为统计分布程度上的测量,是反映一组数据离散程度最常用的一种量化形式，是表示精确度的重要指标

- 标准差定义是总体各单位 **标准值** 与其 **平均数** 离差平方的算术平均数的平方根。

简单来说，标准差是一组数据平均值分散程度的一种度量。

- 一个较大的标准差，代表大部分数值和其平均值之间差异较大；

- 一个较小的标准差，代表这些数值较接近平均值。

```
1 '''
2 例如，A、B两组各有6位学生参加同一次语文测验，
3 A组的分数为95、85、75、65、55、45，
4 B组的分数为73、72、71、69、68、67。
5 分析那组学生之间的差距大？
6 '''
7 a = np.array([95,85,75,65,55,45])
8 b = np.array([73,72,71,69,68,67])
9 print(np.std(a))
```


```
1 17.07825127659933
```

```
1 import math
2 # 按步骤计算下 a 标准差？
3
4 # 1.先求平均值：a_mean = np.mean(a)
5 a_mean = np.mean(a)
6 a_mean
7 # 标准值与平均值的差的平方....
8 math.sqrt(np.sum((a - a_mean) ** 2)/a.size)
9
10
```

```
1 17.07825127659933
```

```
1
```

```
1 2.160246899469287
```

标准差应用于投资上，可作为量度回报稳定性的指标。标准差数值越大，
 代表回报远离过去平均数值，回报较不稳定故风险越高。相反，标准差数值越小，代表回报较为稳定，风险亦较小。

方差ndarray.var()

衡量随机变量或一组数据时离散程度的度量

```
1 a = np.array([95,85,75,65,55,45])
2 b = np.array([73,72,71,69,68,67])
3 print('A组的方差为:',a.var())
4 print('B组的方差为:',b.var())
```

```
1 A组的方差为: 291.6666666666667
2 B组的方差为: 4.666666666666667
```



标准差有计量单位，而方差无计量单位，但两者的作用一样，虽然能很好的描述数据与均值的偏离程度，但是处理结果是不符合我们的直观思维的。

求最大值 `ndarray.max()`

```
1 print(m1)
2 print(m1.max())
3 print('axis=0,从上往下查找:',m1.max(axis=0))
4 print('axis=1,从左往右查找',m1.max(axis=1))
```

```
1 [[ 0  1  2  3  4]
2  [ 5  6  7  8  9]
3  [10 11 12 13 14]
4  [15 16 17 18 19]]
5 19
6 axis=0,从上往下查找: [15 16 17 18 19]
7 axis=1,从左往右查找 [ 4  9 14 19]
```

求最小值 `ndarray.min()`

```
1 print(m1)
2 print(m1.min())
3 print('axis=0,从上往下查找:',m1.min(axis=0))
4 print('axis=1,从左往右查找',m1.min(axis=1))
```

求和 `ndarray.sum()`

```

1 print(m1)
2 print(m1.sum())
3 print('axis=0,从上往下查找:',m1.sum(axis=0))
4 print('axis=1,从左往右查找',m1.sum(axis=1))

```

加权平均值 `numpy.average()`

即将各数值乘以相应的权数，然后加总求和得到总体值，再除以总的单位数

`numpy.average(a, axis=None, weights=None, returned=False)`

- **weights:** 数组，可选

与 `a` 中的值关联的权重数组。`a` 中的每个值都根据其关联的权重对平均值做出贡献。权重数组可以是一维的(在这种情况下，它的长度必须是沿给定轴的 `a` 的大小)或与 `a` 具有相同的形状。如果 `weights=None`，则假定 `a` 中的所有数据的权重等于 1。一维计算是：

`avg = sum(a * weights) / sum(weights)`

对权重的唯一限制是 `sum(weights)` 不能为 0。`

```

1 average_a1 = [20,30,50]
2
3 print(np.average(average_a1))
4 print(np.mean(average_a1))
5

```

```

1 33.333333333333336
2 33.333333333333336

```

实例

使用“示例—权重已知”中的数据，我们对比两位学生的考试成绩

姓名	平时测验	期中考试	期末考试
小明	80	90	95
小刚	95	90	80

学校规定的学科综合成绩的计算方式是：

平时测验占比	期中考试占比	期末考试占比
20%	30%	50%

要求 :比较谁的综合成绩更好

```
1 xiaoming = np.array([80,90,95])
2 xiaogang = np.array([95,90,80])
3 # 权重:
4
5 weights = np.array([0.2,0.3,0.5])
6 # 分别计算小明和小刚的平均值
7 xiaomingng_mean = np.mean(xiaoming)
8 print(xiaomingng_mean)
9 xiaogang_mean = np.mean(xiaogang)
10 print(xiaogang_mean)
11 # 分别计算小明和小刚的加权平均值
12 m_average = np.average(xiaoming,weights=weights)
13 print('小明加权平均数据:',m_average)
14 g_average = np.average(xiaogang,weights = weights)
15 print('小刚加权平均数据:',g_average)
16 # 对比得到结果
17
```

```
1 88.33333333333333
2 88.33333333333333
3 小明加权平均数据: 90.5
4 小刚加权平均数据: 86.0
```

股票价格的波动是股票市场风险的表现，因此股票市场风险分析就是对股票市场价格波动进行分析。波动性代表了未来价格取值的不确定性，这种不确定性一般用 方差 或 标准差 来刻画（Markowitz,1952）。

下表是中国和美国部分时段的股票统计指标，其中中国证券市场的数据由“钱龙”软件下载，美国证券市场的数据取自ECI的“WorldStockExchangeDataDisk”。表2股票统计指标

年份	业绩表现	业绩表现	波动率	波动率
年代	[上证综指]	[标准普尔指数]	[上证综指]	[标准普尔指数]
1996	110.93	16.46	0.2376	0.0573
1997	-0.13	31.01	0.1188	0.0836
1998	8.94	26.67	0.0565	0.0676
1999	17.24	19.53	0.1512	0.0433
2000	43.86	-10.14	0.097	0.0421
2001	-15.34	-13.04	0.0902	0.0732
2002	-20.82	-23.37	0.0582	0.1091

变异系数（Coefficient of Variation）：当需要比较两组数据离散程度大小的时候，如果两组数据的测量尺度相差太大，或者数据量纲的不同，直接使用标准差来进行比较不合适，此时就应当消除测量尺度和量纲的影响，而变异系数可以做到这一点，它是原始数据标准差与原始数据平均数的比

```

1  # 股票信息
2  stat_info = np.array([
3      [110.93, 16.46, 0.2376, 0.0573],
4      [-0.13, 31.01, 0.1188, 0.0836],
5      [8.94, 26.67, 0.0565, 0.0676],
6      [17.24, 19.53, 0.1512, 0.0433],
7      [43.86, -10.14, 0.097, 0.0421],
8      [-15.34, 13.04, 0.0902, 0.0732],
9      [-20.82, 23.37, 0.0582, 0.1091]
10 ])
11
12 # 先计算7年的期望值(平均值) axis = 0 = 1
13 stat_mean = np.mean(stat_info,axis=0)
14 stat_mean
15
16 # 计算7年的标准差
17 stat_std = np.std(stat_info,axis=0)
18
19
20 # 因为标准差是绝对值，不能通过标准差对中美直接进行对比，而变异系数可以直接比较
21 # 变异系数 = 原始数据标准差 / 原始数据平均数
22 stat_std/stat_mean

```



```
1 array([2.02686228, 0.72801021, 0.50570442, 0.32182211])
```

常用函数

数据类型

| 名称 | 描述 | 名称 | 描述 |

| ----- | :----- | ----- | :----- |

| `bool_` | 布尔型数据类型 (True 或者 False) | `float_` | `float64` 类型的简写 |

| `int_` | 默认的整数类型 (类似于 C 语言中的 long, `int32` 或 `int64`) | `float16/32/64` | 半精度浮点数:1 个符号位, 5 个指数位, 10个尾数位

单精度浮点数:1 个符号位, 8 个指数位, 23个尾数位

双精度浮点数,包括: 1 个符号位, 11 个指数位, 52个尾数位|

| `intc` | 和 C 语言的 `int` 类型一样, 一般是 `int32` 或 `int 64` | `complex_` | 复数类型, 与 `complex128` 类型相同 |

| `intp` | 用于索引的整数类型 (类似于 C 的 `ssize_t`, 通常为 `int32` 或 `int64`)

| `complex64/128` | 复数, 表示双 32 位浮点数 (实数部分和虚数部分)

复数, 表示双 64 位浮点数 (实数部分和虚数部分) |

| `int8/16/32/64` | 代表与1字节相同的8位整数

代表与2字节相同的16位整数

代表与4字节相同的32位整数

代表与8字节相同的64位整数 | `str_` | 表示字符串类型 |

| `uint8/16/32/64` | 代表1字节 (8位) 无符号整数

代表与2字节相同的16位整数

代表与4字节相同的32位整数

代表与8字节相同的64位整数 | `string_` | 表示字节串类型,也就是`bytes`类型 |

```
1 # 将数组中的类型存储为浮点型
2 a = np.array([1,2,3,4],dtype=np.float64)
3 a
```

```
1 # 将数组中的类型存储为布尔类型
2 a = np.array([0,1,2,3,4],dtype=np.bool_)
3 print(a)
4 a = np.array([0,1,2,3,4],dtype=np.float_)
5 print(a)
```

```
1 # str_和string_区别
2 #str1 = np.array([1,2,3,4,5,6],dtype=np.str_)
3 string1 = np.array([1,2,3,4,5,6],dtype=np.string_)
4
5 #str2 = np.array(['我们',2,3,4,5,6],dtype=np.string_)
6
7 #print(str1,str1.dtype)
8 print(string1,string1.dtype)
9 #print(str2,str2.dtype)
```

```
1 [b'1' b'2' b'3' b'4' b'5' b'6'] |S1
```

在内存里统一使用unicode，记录到硬盘或者编辑文本的时候都转换成了utf8 UTF-8 将Unicode编码后的字符串保存到硬盘的一种压缩编码方式

定义结构化数据

使用数据类型标识码

字符	对应类型	字符	对应类型	字符	对应类型	字符	对应类型
-----	:-----	-----	:-----	-----	:-----	-----	:-----
b	代表布尔型	i	带符号整型	u	无符号整型	f	浮点型
c	复数浮点型	m	时间间隔 (timedelta)	M	datetime (日期时间)	O	Python对象
S,a	字节串 (S) 与字符串 (a)	U	Unicode	V	原始数据 (void)		

还可以将两个字符作为参数传给数据类型的构造函数。此时，第一个字符表示数据类型，

第二个字符表示该类型在内存中占用的字节数（2、4、8分别代表精度为16、32、64位的

浮点数）：

```

1 # 首先创建结构化数据类型
2 dt = np.dtype([('age', 'i1')]) # int8
3 print(dt)
4 # 将数据类型应用于 ndarray 对象
5 students = np.array([(18), (19)], dtype=dt)
6 students
7 #dt['age']

```

```

1 [ ('age', 'i1')]

```

```

1 array([(18,), (19,)], dtype=[('age', 'i1')])

```

以下示例描述一位老师的姓名、年龄、工资的特征，该结构化数据其包含以下字段：

```

1 str 字段: name
2 int 字段: age
3 float 字段: salary

```

```

1 import numpy as np
2 teacher = np.dtype([('name', np.str_, 2), ('age', 'i2'), ('salary',
3 'f4')])
4 #输出结构化数据teacher
5 print(teacher)
6 #将其应用于ndarray对象
7 b = np.array([('wl', 32, 8357.50), ('lh', 28, 7856.80)], dtype =
8 teacher)
9 print(b)
10 b['name']
11 b['age']

```

```

1 [ ('name', '<U2'), ('age', '<i2'), ('salary', '<f4')]
2 [ ('wl', 32, 8357.5) ('lh', 28, 7856.8)]

```

```

1 array([32, 28], dtype=int16)

```

结构化数据操作

```
1 # 使用数组名[结构化名]
2 print(b)
3 # 取出数组中的所有名称
4 print(b['name'])
5 # 取出数据中的所有年龄
6 print(b['age'])
```

```
1 [('wl', 32, 8357.5) ('lh', 28, 7856.8)]
2 ['wl' 'lh']
3 [32 28]
```

操作文件 loadtxt

loadtxt读取txt文本、csv文件

```
loadtxt(fname, dtype=<type 'float'>, comments='#', delimiter=None,
converters=None, skiprows=0, usecols=None, unpack=False,
ndmin=0, encoding='bytes')
```

参数：

- fname: 指定文件名称或字符串。支持压缩文件，包括gz、bz格式。
- dtype: 数据类型。默认float。
- comments: 字符串或字符串组成的列表。表示注释字符集开始的标志，默认为#。
- delimiter: 字符串。分隔符。
- converters: 字典。将特定列的数据转换为字典中对应的函数的浮点型数据。例如将空值转换为0，默认为空。
- skiprows: 跳过特定行数据。例如跳过前1行（可能是标题或注释）。默认为0。
- usecols: 元组。用来指定要读取数据的列，第一列为0。例如（1， 3， 5），默认为空。
- unpack: 布尔型。指定是否转置数组，如果为真则转置，默认为False。
- ndmin: 整数型。指定返回的数组至少包含特定维度的数组。值域为0、1、2，默认为0。
- encoding: 编码，确认文件是gbk还是utf-8 格式

返回：从文件中读取的数组。

读取普通文件

如data1.txt存在数据:

0 1 2 3 4 5 6 7 8 9

...

20 21 22 23 24 25 26 27 28 29

```
1 # 读取普通文件文件 ,可以不用设置分隔符(空格 制表符)
2 data = np.loadtxt('data1.txt',dtype=np.int32,comments='#')
3 print(data,data.shape)
4 np.sum(data)
```

```
1 [[ 0  1  3  3  4  5  6  7  8  9]
2  [10 11 12 13 14 15 16 17 18 19]
3  [20 21 22 23 24 25 26 27 28 29]] (3, 10)
```

```
1 436
```

```
1 # 读取csv文件 ,取药设置分隔符,csv默认为,号
2 data = np.loadtxt('csv_test.csv',dtype=np.int32,delimiter=',')
3 print(data,data.shape)
```

```
1 [[ 0  1  2  3  4  5  6  7  8  9]
2  [10 11 12 13 14 15 16 17 18 19]
3  [20 21 22 23 24 25 26 27 28 29]] (3, 10)
```

不同列标识不同信息 数据读取

数据如下:

姓名 年龄 性别 身高

小王 21 男 170

.....

老王 50 男 180

文件:has_title.txt

```
1 # 1. 以上数据由于不同列数据标识的含义和类型不同,因此需要自定义数据类型
2 user_info = np.dtype([('name', 'U10'), ('age', 'i1'), ('gender', 'U1'),
3   ('height', 'i2')])
4 print(user_info)
5 # 2. 使用自定义的数据类型 读取数据,
6 data = np.loadtxt('has_title.txt', dtype=user_info, skiprows=1,
7   encoding='utf-8')
8 # 注意以上参数中:a.设置类型; b.跳过第一行; c.跳过第一行; d.编码
9 print(data)
```

```
1 [('name', '<U10'), ('age', 'i1'), ('gender', '<U1'), ('height',
2   '<i2')]
3 [('小王', 21, '男', 170) ('小张', 25, '女', 165) ('小花', 19, '女',
4   167)
5   ('老王', 40, '男', 180) ('小韩', 24, '男', 168) ('小白', 21, '女',
6   167)
7   ('小花1', 19, '女', 159) ('小刘', 26, '男', 170) ('小秦', 21, '男',
8   168)
9   ('小胖', 21, '女', 175) ('娜娜', 19, '女', 160) ('朵朵', 20, '女',
10  167)]
```

```
1 '''
2 计算平均年龄
3 '''
4 # 获取年龄的数组
5 ages = data['age']
6 ages.mean()
7 # 计算年龄的中位数
8 np.median(ages)
```

```
1 21.0
```

```
1 # 计算平均身高
2
3 # 计算身高中位数
4
5 data['gender']
```

```
1 array(['男', '女', '女', '男', '男', '女', '女', '男', '男', '女',
2        '女', '女'],
3        dtype='<U1')
```

```
1 # 计算女生的平均身高
2 print(data['gender'])
3 isgirl = data['gender'] == '女'
4 print(isgirl)
5 print(data['height'])
6 data['height'][isgirl]
7 '{:.2f}'.format(np.mean(data['height'][isgirl]))
```

```
1 ['男' '女' '女' '男' '男' '女' '女' '男' '男' '女' '女' '女']
2 [False  True  True False False  True  True False False  True  True
3    True]
3 [170 165 167 180 168 167 159 170 168 175 160 167]
```

```
1 '165.71'
```

读取指定的列

```

1 # 读取指定的列 usecols=(1,3) 标识只读取第2列和第4列
2 user_info = np.dtype([('age', 'i1'), ('height', 'i2')])
3 print(user_info)
4 # 使用自定义的数据类型 读取数据,
5 data =
  np.loadtxt('has_title.csv', dtype=user_info, delimiter=',', skiprows=
  1, usecols=(1,3))
6 # 注意以上参数中:a.设置类型; b.跳过第一行; c.分隔符 ; d.跳过第一行; e.
  编码
7 print(data)
8
9

```

数据中存在空值进行处理

需要借助用于 converters参数,传递一个字典,key为列索引,value为对列中值得处理

比如:

csv中学生信息中存在空的年龄信息:

姓名 年龄 性别 身高

小王 21 男 170

...

小谭 男 169

...

小陈 27 男 177

converters={列索引:处理的函数,key1:value}

文件:has_empty_data.csv

```

1 # 处理空数据,需要创建一个函数,接收列的参数,并加以处理.
2 def parse_age(age):
3     try:
4         return int(age)
5     except:
6         return 0

```



```

1 data =
  np.loadtxt('has_empty_data.csv', delimiter=',', skiprows=1, usecols=1
    , converters={1: parse_age})
2 print(data)

```

```

1 --!!!-
2 --!!!-
3 --!!!-
4 --!!!-
5 ---
6 --!!!-
7 --!!!-
8 --!!!-
9 [21. 25. 19.  0. 21. 19. 27.]

```

```

1 # 处理空数据,需要创建一个函数,接收列的参数,并加以处理.
2 def parse_age(age):
3     try:
4         return int(age)
5     except:
6         return 0

```

```

1 # 和之前一样的步骤
2 print(user_info)
3 # 使用自定义的数据类型 读取数据,
4 data =
  np.loadtxt('has_empty_data.csv', dtype=user_info, delimiter=',', skip
    rows=1, usecols=(1, 3), converters={1: parse_age})
5 print(data)

```

计算班级年龄的平均值.由于存在0的数据,因此一般做法是将中位数填充

```

1 # 填充中位数:
2 ages = data['age']
3 ages[ages==0] = np.median(ages)
4 print(ages)
5
6 # 计算平均值
7 np.round(np.mean(ages), 2)

```

✳ 作业?

考虑下列与学生（虚构）人口有关的数据文本文件

```
# Student data collected on 17 July 2014.
# Researcher: Dr Wicks, University College Newbury.

# The following data relate to N = 20 students. It
# has been totally made up and so therefore is 100%
# anonymous.
```

Subject (ID)	Sex M/F	DOB dd/mm/yy	Height m	Weight kg	BP mmHg	VO2max mL.kg ⁻¹ .min ⁻¹
JW-1	M	19/12/95	1.82	92.4	119/76	39.3
JW-2	M	11/1/96	1.77	80.9	114/73	35.5
JW-3	F	2/10/95	1.68	69.7	124/79	29.1
JW-6	M	6/7/95	1.72	75.5	110/60	45.5
# JW-7	F	28/3/96	1.66	72.4	101/68	-
JW-9	F	11/12/95	1.78	82.1	115/75	32.3
JW-10	F	7/4/96	1.60	-	-/-	30.1
JW-11	M	22/8/95	1.72	77.2	97/63	48.8

文件为:student-data.txt

```
1  '''1 .找出学生的平均身高?'''
2
3  # 1. 定义数据类型
4
5  # 2.确定文件内容特点: a.需要跳过9行; b.只需要第二列和第4列
6  data = np.loadtxt('student-data.txt',dtype=object,usecols=
7  (1,3),skiprows=9)
8  data
9
10 # 3. 取得身高信息
11
12 # 4.计算学生身高平均值 mean
13
```

```
1  array(['M/F', 'm'],
2        ['M', '1.82'],
3        ['M', '1.77'],
4        ['F', '1.68'],
```

```
5      ['M', '1.72'],
6      ['F', '1.78'],
7      ['F', '1.60'],
8      ['M', '1.72'],
9      ['M', '1.83'],
10     ['F', '1.56'],
11     ['F', '1.64'],
12     ['M', '1.63'],
13     ['M', '1.67'],
14     ['M', '1.66'],
15     ['F', '1.59'],
16     ['F', '1.70'],
17     ['M', '1.97'],
18     ['F', '1.66'],
19     ['F', '1.63'],
20     ['M', '1.69']], dtype=object)
```

```
1  '''2. 找到男学生的平均身高?'''
2
3  # 1.首先判断那些数据性别为男性 ,或者取得性别为男的行索引
4
5  # 提取出和性别对应为True的身高信息
6
7
8  # 计算平均值
9
```

```
1  # 3.计算女生身高中位数? 步骤同上.
2
```