5.2 - NumPy: Mathematical & Statistical Methods

Ha Khanh Nguyen (hknguyen)

---

1. Calculation Basic Statistics

```
import numpy as np

np.random.seed(430)
arr = np.random.randn(5, 4)
arr
```

```
## array([[ 0.4742312 , -0.85940424, -0.75509958, -0.54634703],
##        [ 1.01367802, -1.48055291,  2.12899127, -0.20291703],
##        [ 0.57428592, -0.35479235,  0.19028642,  1.10250873],
##        [ 1.24961928,  0.92632742, -0.45853894,  0.65669902],
##        [-0.25024139,  0.16887152,  0.05787939, -0.40472773]])
```

- Compute the sample mean:

```
arr.mean()
```

```
## 0.1615378501518339
```

```
np.mean(arr)
```

```
## 0.1615378501518339
```

- Compute the sum of all the elements:

```
arr.sum()
```

```
## 3.2307570030366777
```

- Functions like `mean` and `sum` take an optional axis argument that computes the statistic over the given axis, resulting in an array with one fewer dimension:

```
# compute mean across the columns
arr.mean(axis=1)
```

```
## array([-0.42165491,  0.36479984,  0.37807218,  0.59352669, -0.10705455])
```

```
# compute sum down the rows
arr.sum(axis=0)
```

```
## array([ 3.06157304, -1.59955056,  1.16351856,  0.60521596])
```

Method	Description
sum	Sum of all the elements in the array or along an axis; zero-length arrays have sum 0
mean	Arithmetic mean; zero-length arrays have NaN mean
std, var	Standard deviation and variance, respectively, with optional degrees of freedom adjustment (default denominator n)
min, max	Minimum and maximum
argmin, argmax	Indices of minimum and maximum elements, respectively
cumsum	Cumulative sum of elements starting from 0
cumprod	Cumulative product of elements starting from 1

2. Sorting

- Like Python's built-in list type, NumPy arrays can be sorted in-place with the `sort` method:

```
arr = np.random.randn(5)
arr
```

```
## array([-0.99232288,  1.17977178, -0.15407498, -1.02124231,  0.95328187])
```

```
arr.sort()
arr
```

```
## array([-1.02124231, -0.99232288, -0.15407498,  0.95328187,  1.17977178])
```

- You can sort each one-dimensional section of values in a multidimensional array inplace along an axis by passing the axis number to sort:

```
arr = np.random.randn(5, 3)
arr
```

```
## array([[ 1.64033079,  0.68565206,  0.54643213],
##        [-2.76931098, -0.30328563, -0.38740235],
##        [ 0.55320402, -1.03859445, -0.00931142],
##        [-1.24925273, -0.90224748, -0.97430454],
##        [-1.04522033, -2.35703711,  0.94462521]])
```

```
arr.sort(1)
arr
```

```
## array([[ 0.54643213,  0.68565206,  1.64033079],
##        [-2.76931098, -0.38740235, -0.30328563],
##        [-1.03859445, -0.00931142,  0.55320402],
##        [-1.24925273, -0.97430454, -0.90224748],
##        [-2.35703711, -1.04522033,  0.94462521]])
```

This lecture note is modified from Chapter 4 of Wes McKinney's Python for Data Analysis 2nd Ed (https://www.oreilly.com/library/view/python-for-data/9781491957653/).