1 Numpy basics: building an array from lists, basic statistics, converting to booleans, referencing the array, and taking slices

Most commonly we will be loading files into NumPy arrays, but here we build an array from lists and perform some basics stats on the array.

The examples below construct and use a 2 dimensional array (which may be though of as 'rows and columns'). Later we will look at higher dimensional arrays.

1.1 Building an array from lists

We use the *np.array* function to build an array from existing lists. Here each list represents a row of a data table.

```
import numpy as np

row0 = [23, 89,100]
row1 = [10, 51, 99]
row2 = [40, 78, 102]
row3 = [35, 81, 110]
row4 = [50, 75, 95]
row5 = [65, 51, 101]

data = np.array([row0, row1, row2, row3, row4, row5])
```

We now have a data array:

```
print (data)

OUT:

[[ 23  89 100]
 [ 10  51  99]
 [ 40  78 102]
 [ 35  81 110]
 [ 50  75  95]
 [ 65  51 101]]
```

1.2 Performing basic statistics on the array

We can see, for example, the mean of all the data

```
print (data.mean())

OUT:

69.72222222222223
```

Or we can use the *axis* argument to show the mean by column or mean by row:

```
print (np.mean(data,axis=0)) # average all rows in each column
print (np.mean(data,axis=1)) # average all columns in each row

OUT:

[ 37.16666667  70.83333333 101.16666667]
[70.66666667 53.33333333 73.33333333 75.33333333 73.33333333 72.33333333]
```

Some other commonly used statistics (all of which are by column, or dimension 0, below):

```
print ('Mean:\n', np.mean(data, axis=0))
print ('Median:\n', np.median(data, axis=0))
print ('Sum:\n', np.sum(data, axis=0))
print ('Maximum:\n', np.max(data,axis=0))
print ('Minimum\n:', np.min(data,axis= 0))
print ('Range:\n', np.ptp(data, axis=0))
print ('10th percentile:\n', np.percentile(data, 10, axis=0))
print ('90th percentile:\n', np.percentile(data, 90, axis=0))
print ('Population standard deviation\n' ,np.std(data, axis=0))
print ('Sample standard deviation:\n', np.std(data, axis=0, ddof=1))
print ('Population variance:\n', np.var(data, axis=0))
print ('Sample variance:\n', np.var(data, axis=0, ddof=1))

OUT:

Mean:
 [ 37.16666667  70.83333333 101.16666667]
Median:
 [ 37.5  76.5 100.5]
Sum:
 [223 425 607]
Maximum:
 [ 65  89 110]
Minimum
: [10 51 95]
Range:
 [55 38 15]
10th percentile:
 [16.5 51.  97. ]
90th percentile:
 [ 57.5  85.  106. ]
Population standard deviation
 [17.75215167 14.6562463   4.52462399]
Sample standard deviation:
 [19.44650783 16.05511341  4.95647724]
Population variance:
 [315.13888889 214.80555556  20.47222222]
Sample variance:
 [378.16666667 257.76666667  24.56666667]
```

The returned array may be referenced by index number (beginning at zero):

```
results = np.mean(data, axis=0)

print (results[0])
OUT:

37.166666666666664
```

Basic python may also be incorporated into the statistics. For example the 10th percentiles, from 0 to 100 may be calculated in a loop (remember that the standard Python range function goes up to, but does not include, the maximum value given, so we need to put a higher maximum than 100 in order to include the 100th percentile in the loop):

```
for percent in range(0,101,10):
    print(percent,'percentile:',np.percentile(data, percent, axis=0))

OUT:

0 percentile: [10. 51. 95.]
```

```
10 percentile: [16.5 51.  97. ]
20 percentile: [23. 51. 99.]
30 percentile: [29.  63.  99.5]
40 percentile: [ 35.  75. 100.]
50 percentile: [ 37.5  76.5 100.5]
60 percentile: [ 40.  78. 101.]
70 percentile: [ 45.   79.5 101.5]
80 percentile: [ 50.  81. 102.]
90 percentile: [ 57.5  85.  106. ]
100 percentile: [ 65.  89. 110.]
```

1.3 Converting values to a boolean True/False

We can use test values against some standard. For example, here we test whether a value is equal to, or greater than, the mean of the column.

```
column_mean = np.mean(data, axis=0)
greater_than_mean = data >= column_mean

print (greater_than_mean)

OUT:

[[False  True False]
 [False False False]
 [ True  True  True]
 [False  True  True]
 [ True  True False]
 [ True False False]]
```

True/False values in Python may also be used in calculations where False has an equivalent value to zero, and True has an equivalent value to 1.

```
print (np.mean(greater_than_mean, axis=0))

OUT:

[0.5        0.66666667 0.33333333]
```

1.4 Referencing arrays and taking slices

NumPy arrays are referenced similar to lists, except we have two (or more dimensions). For a two dimensional array the reference is [dimension_0, dimension_1], which is equivalent to [row, column].

REMEMBER: Like lists, indices start at index zero, and when taking a slice the slice goes up to, but does not include, the maximum value given.

```
print ('Row zero:\n', data[0,:])
print ('Column one:\n', data[:,1])
# Note in the example below a slice goes up to ,but dot include, the maximum index
print ('Rows 0-3, Column zero:\n', data[0:4,0])
print ('Row 1, Column 2\n', data[1,2])

OUT:

Row zero:
 [ 23  89 100]
Column one:
 [89 51 78 81 75 51]
```

```
Rows 0-3, Column zero:
 [23 10 40 35]
Row 1, Column 2
 99
```