The Engineering World #DataScience 12 & 13

May 31, 2018

AKKAL BAHADUR BIST
DATA SCIENTIST AT
KATHMANDU INSTITUTE OF APPLIED SCIENCES (KIAS)
Center for Conservation Biology (CCB)

1 USING NUMPY TO PERFORM ARITHMETIC OPERATION ON DATA

```
In [1]: import numpy as np
        from numpy.random import randn
```

```
In [2]: np.set_printoptions(precision = 2)
```

1.0.1 Creating Arrays

1.0.2 Creating array using a list

```
In [3]: a = np.array([1,2,3,4,5,6])
        a
```

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

```
In [4]: b = np.array([[10,20,30,40,50], [60,70,80,90,100]])
        b
```

```
Out[4]: array([[ 10,  20,  30,  40,  50],
               [ 60,  70,  80,  90, 100]])
```

1.0.3 Creating arrays vis assignment

```
In [5]: np.random.seed(25)
        c = 35*np.random.randn(6)
        c
```

```
Out[5]: array([  7.99,  35.94, -29.39, -20.69, -33.49,  -7.78])
```

```
In [6]: d = np.arange(1,35)
        d
```

```
Out[6]: array([ 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])
```

1.0.4 Performing arithmetic on array

```
In [7]: a*10

Out[7]: array([10, 20, 30, 40, 50, 60])

In [8]: c+a

Out[8]: array([  8.99,  37.94, -26.39, -16.69, -28.49,  -1.78])

In [9]: c-a

Out[9]: array([  6.99,  33.94, -32.39, -24.69, -38.49, -13.78])

In [10]: c*a

Out[10]: array([   7.99,   71.88,  -88.16,  -82.77, -167.46,  -46.69])

In [11]: c/a

Out[11]: array([ 7.99, 17.97, -9.8 , -5.17, -6.7 , -1.3 ])
```

1.0.5 Multiplying matrices and basic algebra

```
In [12]: aa = np.array([[1.,2.,3.,4.,5.], [10.,20.,30.,40.,50.], [100.,200.,300.,400.,500.]])
         aa

Out[12]: array([[  1.,   2.,   3.,   4.,   5.],
                [ 10.,  20.,  30.,  40.,  50.],
                [100., 200., 300., 400., 500.]])

In [13]: bb = np.array([[0.,1.,2.,3.,4.], [00.,11.,22.,33.,44.], [100.,200.,300.,400.,500.]])
         bb

Out[13]: array([[  0.,   1.,   2.,   3.,   4.],
                [  0.,  11.,  22.,  33.,  44.],
                [100., 200., 300., 400., 500.]])

In [14]: aa *bb

Out[14]: array([[0.00e+00, 2.00e+00, 6.00e+00, 1.20e+01, 2.00e+01],
                [0.00e+00, 2.20e+02, 6.60e+02, 1.32e+03, 2.20e+03],
                [1.00e+04, 4.00e+04, 9.00e+04, 1.60e+05, 2.50e+05]])

In [15]: a1 = np.array([[1,2,3], [4,5,6], [7,8,9]])
         a1

Out[15]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

```
In [16]: b1 = np.array([[10,20,30],[40,50,60],[70,80,90]])
         b1

Out[16]: array([[10, 20, 30],
                [40, 50, 60],
                [70, 80, 90]])

In [17]: np.dot(a1,b1)

Out[17]: array([[ 300,  360,  420],
                [ 660,  810,  960],
                [1020, 1260, 1500]])
```

2 DESCRIPTIVE STATISTICS

2.0.1 Generating summary statistics using pandas and scipy

```
In [18]: import numpy as np
         import pandas as pd
         from pandas import Series, DataFrame

         import scipy
         from scipy import stats

In [19]: address = 'mtcars.csv'
         cars = pd.read_csv(address)
         cars.columns = ['car_names', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'a
         cars.head()
```

Out[19]:

	car_names	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

2.0.2 Looking at summary statistics that describe a variable's numeric values

```
In [20]: cars.sum()

Out[20]: car_names    Mazda RX4Mazda RX4 WagDatsun 710Hornet 4 Drive...
         mpg                                                     642.9
```

```
cyl                                                    198
disp                                                7383.1
hp                                                    4694
drat                                                115.09
wt                                                 102.952
qsec                                                571.16
vs                                                      14
am                                                      13
gear                                                   118
carb                                                    90
dtype: object
```

In [21]:
```
cars.sum(axis=1)
```

Out[21]:
```
0     328.980
1     329.795
2     259.580
3     426.135
4     590.310
5     385.540
6     656.920
7     270.980
8     299.570
9     350.460
10    349.660
11    510.740
12    511.500
13    509.850
14    728.560
15    726.644
16    725.695
17    213.850
18    195.165
19    206.955
20    273.775
21    519.650
22    506.085
23    646.280
24    631.175
25    208.215
26    272.570
27    273.683
28    670.690
29    379.590
30    694.710
31    288.890
dtype: float64
```

In [22]:
```
cars.median()
```

```
Out[22]: mpg      19.200
         cyl       6.000
         disp    196.300
         hp      123.000
         drat      3.695
         wt        3.325
         qsec     17.710
         vs        0.000
         am        0.000
         gear      4.000
         carb      2.000
         dtype: float64

In [23]: cars.mean()

Out[23]: mpg      20.090625
         cyl       6.187500
         disp    230.721875
         hp      146.687500
         drat      3.596563
         wt        3.217250
         qsec     17.848750
         vs        0.437500
         am        0.406250
         gear      3.687500
         carb      2.812500
         dtype: float64

In [24]: cars.max()

Out[24]: car_names    Volvo 142E
         mpg                33.9
         cyl                   8
         disp                472
         hp                  335
         drat               4.93
         wt                5.424
         qsec               22.9
         vs                    1
         am                    1
         gear                  5
         carb                  8
         dtype: object

In [25]: mpg = cars.mpg
         mpg.idxmax()

Out[25]: 19
```

2.0.3 Looking at summary statistics that describe variable distribution

```
In [26]: cars.std()
```

```
Out[26]: mpg       6.026948
         cyl       1.785922
         disp    123.938694
         hp       68.562868
         drat      0.534679
         wt        0.978457
         qsec      1.786943
         vs        0.504016
         am        0.498991
         gear      0.737804
         carb      1.615200
         dtype: float64
```

```
In [27]: cars.var()
```

```
Out[27]: mpg        36.324103
         cyl         3.189516
         disp    15360.799829
         hp       4700.866935
         drat        0.285881
         wt          0.957379
         qsec        3.193166
         vs          0.254032
         am          0.248992
         gear        0.544355
         carb        2.608871
         dtype: float64
```

```
In [28]: gear = cars.gear
         gear.value_counts()
```

```
Out[28]: 3    15
         4    12
         5     5
         Name: gear, dtype: int64
```

```
In [29]: cars.describe()
```

```
Out[29]:              mpg        cyl        disp          hp       drat          wt  \
         count  32.000000  32.000000   32.000000   32.000000  32.000000  32.000000
         mean   20.090625   6.187500  230.721875  146.687500   3.596563   3.217250
         std     6.026948   1.785922  123.938694   68.562868   0.534679   0.978457
         min    10.400000   4.000000   71.100000   52.000000   2.760000   1.513000
         25%    15.425000   4.000000  120.825000   96.500000   3.080000   2.581250
         50%    19.200000   6.000000  196.300000  123.000000   3.695000   3.325000
```

75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000

	qsec	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.0000
mean	17.848750	0.437500	0.406250	3.687500	2.8125
std	1.786943	0.504016	0.498991	0.737804	1.6152
min	14.500000	0.000000	0.000000	3.000000	1.0000
25%	16.892500	0.000000	0.000000	3.000000	2.0000
50%	17.710000	0.000000	0.000000	4.000000	2.0000
75%	18.900000	1.000000	1.000000	4.000000	4.0000
max	22.900000	1.000000	1.000000	5.000000	8.0000