# KEY_Lesson15_Basic_Stats_I_Averages

June 4, 2020

## 1 Basic Statistics I: Averages

An **average** is the central value of a set of numbers.

The **arithmetic mean** is the sum of the elements along the axis divided by the number of elements.

[1]:
```
import numpy as np
```

[2]:
```
# Make an array of rank 1
array1 = np.array([1, 2, 3, 4, 5])
```

Let's manually calculate the average of array arr.

[3]:
```
# Manually calculate the average
average_manual = np.sum(array1)/len(array1)
print(average_manual)
```

```
3.0
```

Now, to make our life easier, let's use the built-in method mean to calculate the average.

[4]:
```
# Calculate the average using the built-in method mean
average_numpy = np.mean(array1)
print(average_numpy)
```

```
3.0
```

We can also calculate averages on arrays of rank 2.

[5]:
```
# Make an array of rank 2
array2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
```

[6]:
```
# Determine the mean of the array of rank 2
print(np.mean(array2))
```

```
6.5
```

[7]:
```
# Determine the mean of each column
averages_columnwise = np.mean(array2, axis = 0)
print(averages_columnwise)
```

```
[5.5 6.5 7.5]
```

[8]:
```
# Determine the mean of each row
averages_rowwise = np.mean(array2, axis = 1)
print(averages_rowwise)
```

```
[ 2.  5.  8. 11.]
```

We have been practicing on simulated data, so let's now world with a real-world dataset by using the iris dataset.

[9]:
```
# Import the load_iris method
from sklearn.datasets import load_iris
```

[10]:
```
# Import pandas, so that we can work with the data frame version of the iris␣
↪data
import pandas as pd
```

[11]:
```
# Load the iris data
iris = load_iris()
```

[12]:
```
# Convert the iris data to a data frame format, so that it's easier to view
# and process
iris_df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
iris_df
```

[12]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 |
| .. | … | … | … | … |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 |

[150 rows x 4 columns]

[13]:
```
# Determine the mean of each feature
averages_column = np.mean(iris_df, axis = 0)
print(averages_column)
```

```
sepal length (cm)    5.843333
sepal width (cm)     3.057333
petal length (cm)    3.758000
```

```
petal width (cm)     1.199333
dtype: float64
```

So we can determine the averages by row, but should we do this? Why or why not?

[14]:
```
# Determine the mean of each row
averages_row = np.mean(iris_df, axis = 1)
print(averages_row)
```

```
0      2.550
1      2.375
2      2.350
3      2.350
4      2.550
       …
145    4.300
146    3.925
147    4.175
148    4.325
149    3.950
Length: 150, dtype: float64
```

How should we interpret a value of averages_row? It's hard to interpret these values, since taking an average across different features does not make sense.

Even though we can calculate any statistics that we want, some statistics may not be interpretable. So be careful on your calculations!

Great work! You just learned about how to take averages in Python! You learned:

- To manually and automatically calculate averages
- To calculate averages by row and by columns.
- To calculate averages on a real dataset.
- To know when it is appropriate to calculate row-wise or column-wise averages.