Statistics

Learn how to apply statistical metrics to NumPy data.

Chapter Goals:

- Learn about basic statistical analysis in NumPy
- Write code to obtain statistics for NumPy arrays

A. Analysis

It is often useful to analyze data for its main characteristics and interesting trends. Though we will go more in-depth on data analysis in the section of this course titled Data Preprocessing with scikit-learn (https://www.educative.io/collection/page/6083138522447872/5629499534213120/5697070107197440/), there are still a few techniques in NumPy that allow us to quickly inspect data arrays.

For example, we can obtain minimum and maximum values of a NumPy array using its inherent `min` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.min.html) and `max` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.max.html) functions. This gives us an initial sense of the data's range, and can alert us to extreme outliers in the data.

The code below shows example usages of the `min` and `max` functions.

```
arr = np.array([[0, 72, 3],
                [1, 3, -60],
                [-3, -2, 4]])
print(arr.min())
print(arr.max())

print(repr(arr.min(axis=0)))
print(repr(arr.max(axis=-1)))
```

Output 0.341s

```
-60
72
array([ -3,  -2, -60])
array([72,  3,  4])
```

The `axis` keyword argument is identical to how it was used in `np.argmin` and `np.argmax` from the chapter on Indexing. In our example, we use `axis=0` to find an array of the minimum values in each column of `arr` and `axis=1` to find an array of the maximum values in each row of `arr`.

B. Statistical metrics

NumPy also provides basic statistical functions such as `np.mean` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html), `np.var` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.var.html), and `np.median` (https://docs.scipy.org/doc/numpy/reference/generated/numpy.median.html), to calculate the mean, variance, and median of the data, respectively.

The code below shows how to obtain basic statistics with NumPy. Note that `np.median` applied without `axis` takes the median of the flattened array.

```
arr = np.array([[0, 72, 3],
                [1, 3, -60],
                [-3, -2, 4]])
print(np.mean(arr))
print(np.var(arr))
print(np.median(arr))
print(repr(np.median(arr, axis=-1)))
```

Output 0.417s

```
2.0
977.3333333333334
1.0
array([ 3.,  1., -2.])
```

Each of these functions takes in the data array as a required argument and `axis` as a keyword argument. For a more comprehensive list of statistical functions (e.g. calculating percentiles, creating histograms, etc.), check out the NumPy statistics page (https://docs.scipy.org/doc/numpy/reference/routines.statistics.html).

Time to Code!

Each coding exercise in this chapter will be to complete a small function that takes in a 2-D NumPy matrix (`data`) as input. The first function to complete is `get_min_max` , which returns the overall minimum and maximum element in `data` .

Set `overall_min` equal to `data.min` applied with no arguments.

Set `overall_max` equal to `data.max` applied with no arguments.

Return a tuple of `overall_min` and `overall_max` , in that order.

```
5
6
```

Show Results Show Console

2 of 2 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓		-9	-9	Correct value for overall_min, good job!
✓		20	20	Correct value for overall_max, good job!

0.403s

Next, we'll complete `col_min`, which returns the minimums across each column of `data`.

Set `min0` equal to `data.min` with the `axis` keyword argument set to `0`.

Then return `min0`.

```
def col_min(data):
  min0 = data.min(axis=0)
  return min0


```

Show Results | Show Console

1 of 1 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓		[-9 -1 -5 -3 0]	[-9 -1 -5 -3 0]	Correct return value, good job!

0.340s

The final function to complete is `basic_stats`, which returns the mean, median, and variance of `data`.

Set `mean` equal to `np.mean` applied to `data`.

Set `median` equal to `np.median` applied to `data`.

Set `var` equal to `np.var` applied to `data`.

Return a tuple containing `mean`, `median`, and `var`, in that order.

```
def basic_stats(data):
  mean = np.mean(data)
  median = np.median(data)
  var = np.var(data)
  return mean, median, var
```

Show Results | Show Console

3 of 3 Tests Passed

Result	Input	Expected Output	Actual Output	Reason
✓		3.15	3.15	Correct value for mean, good job!
✓		1.0	1.0	Correct value for median, good job!
✓		65.02750000000002	65.02750000000002	Correct value for var, good job!

0.426s

Back
Filtering

Next
Aggregation

Mark as Completed