Python102

Python for Data Science Bootcamp

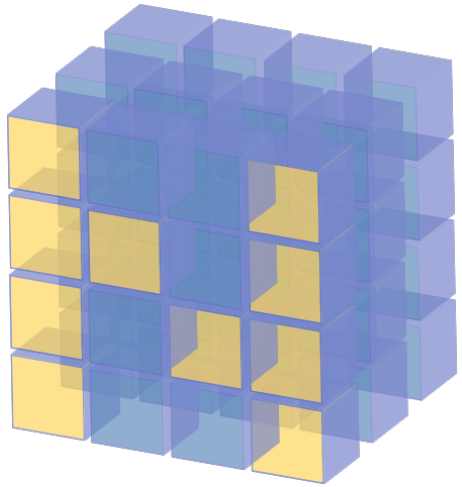
(4.1) Python for Data Analysis

NumPy

AIAT Academy

Python for Data Analysis Outline

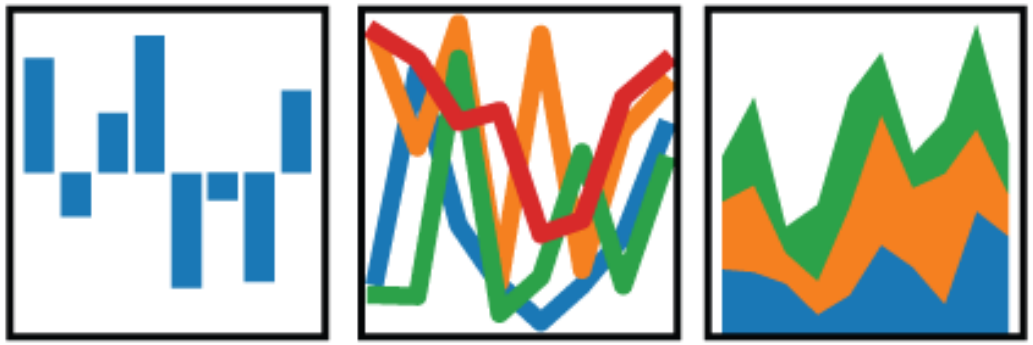
- **NumPy**
- Pandas



NumPy

pandas

$$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$$

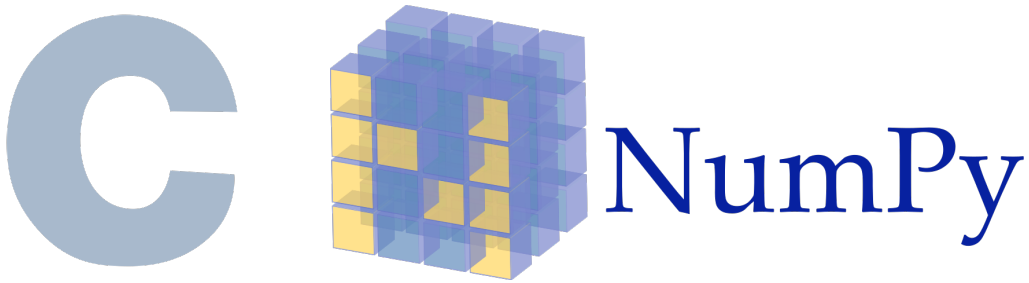


AiAT ACADEMY

NumPy

Introduction to NumPy

- Fundamental package for scientific computing with Python.
  - Linear Algebra, Fourier transform, etc.
- NumPy is incredibly fast since it has bindings to C libraries.
- Optimized library for matrix and vector computation.
- N-dimensional array object
- Open source



NumPy Installation

- It is highly recommend you install Python using the Anaconda distribution to make sure all dependencies all sync up with the use of a conda install.
- To install NumPy, just going to your terminal or command prompt and typing

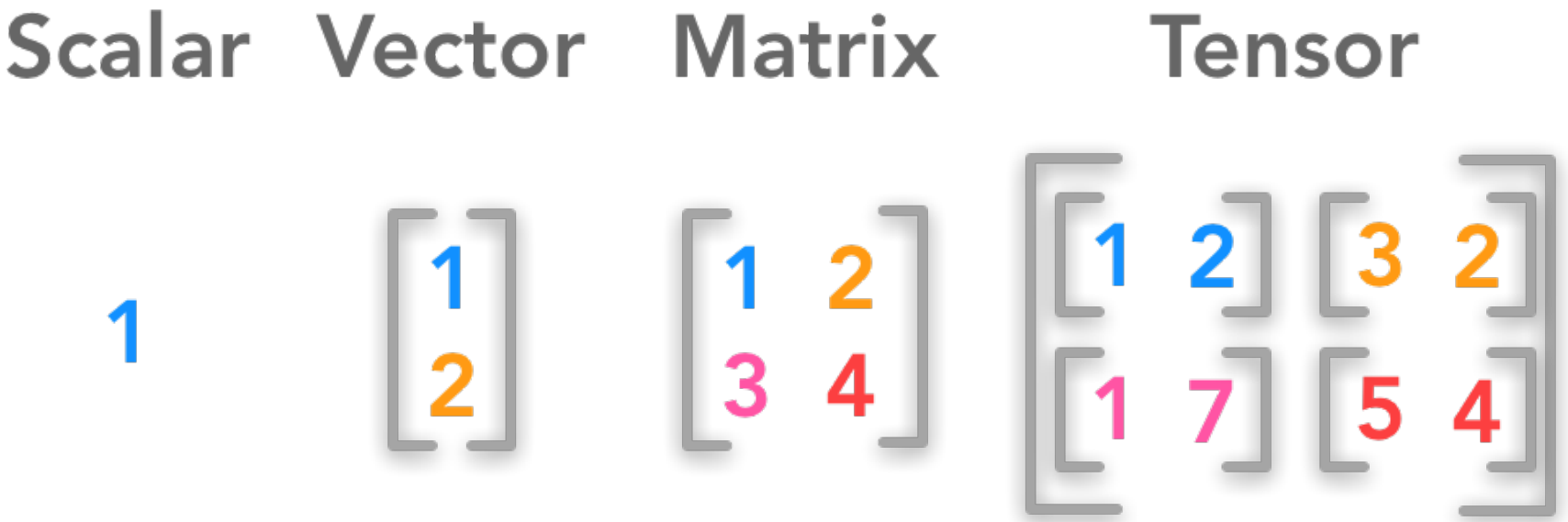
```
conda install numpy
```

or

```
pip install numpy
```

NumPy in Python102

- NumPy arrays are the main way we will use NumPy in this course.
- NumPy arrays come in 2 flavors: Vectors and Matrices.
- Vectors are 1-d arrays and 2-d arrays



AiAT ACADEMY

NumPy Arrays

AiAT ACADEMY

Main data type: `ndarray` (`np.ndarray`)

This is the data type that you will use to represent vector/matrix computation

Note: Constructor function is `np.array()`

NumPy Arrays (np.array)

```
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 6]])

# >> array([ [1, 2, 3],
#            [4, 5, 6] ] )
```

```
np.array([Numerical list(s)])
```

NumPy Arrays (np.array)

AiAT ACADEMY

```
import numpy as np

A_float = np.array([1, 2, 3], float)

# >> array([ [1., 2., 3.],
#            [4., 5., 6.] ] )
```

NumPy Arrays (np.arange)

AiAT ACADEMY

```
import numpy as np

np.arange(3)          # >> array([0, 1, 2])

np.arange(3.0)        # >> array([0., 1., 2.])

np.arange(3, 7)       # >> array([3, 4, 5, 6])

np.arange(3, 7, 2)    # >> array([3, 5])
```

```
np.arange(start, stop, step)
```

NumPy Arrays (np.zeros)

AiAT ACADEMY

```
import numpy as np

A = np.zeros((2, 3))
# >> array( [ [0., 0., 0.],
#             [0., 0., 0.] ] )

A.shape
# >> (2, 3)
```

```
np.zeros(shape)
```

NumPy Arrays (np.ones)

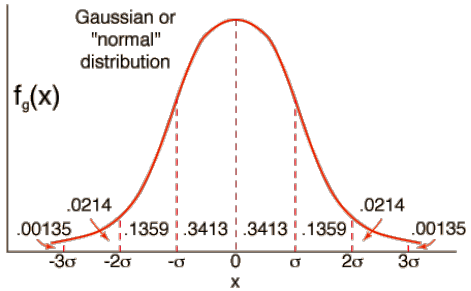
AiAT ACADEMY

```
import numpy as np

A = np.ones((2, 3))
# >> array( [ [1., 1., 1.],
#             [1., 1., 1.] ] )

A.shape
# >> (2, 3)
```

```
np.ones(shape)
```

NumPy Arrays (np.random)

```
np.random.random((2, 3))
# >> array([ [ 0.78084261, 0.64328818, 0.55380341],
#            [ 0.24611092, 0.37011213, 0.83313416] ])
```




```
np.random.normal((loc=1.0, scale=2.0, size=(2, 2)))
# >> array([ [ 2.87799514, 0.6284259 ],
#            [ 3.10683164, 2.05324587] ])
```

NumPy Arrays (Array Attributions)

AiAT ACADEMY

```
a = np.arange(10).reshape((2, 5))
```

```
a.ndim      # 2 (dimensions)
a.shape     # (2, 5) (shape of array)
a.size      # 10 (number of elements)
a.T         # Transpose
a.dtype     # Data type
```

AiAT ACADEMY

NumPy Indexing and Selection

NumPy Indexing and Selection

AiAT ACADEMY

```
a = np.arange(4)     # >> array( [0, 1, 2, 3] )

a[1]     # >> 1

a[1:3]   # >> array( [1, 2] )

a[:3]    # >> array( [0, 1, 2] )

a[1:]    # >> array( [1, 2, 3] )

a[1:3] = 10 # >> a = array( [0, 10, 10, 3] )
```

NumPy Indexing and Selection
(Nested Arrays | 2D+ Arrays)

```
a_2d = np.array([ [1, 2, 3], [4, 5, 6] ])

# >> array( [ [1, 2, 3],
#             [4, 5, 6] ] )

a_2d[1]     # >> array([4, 5, 6])

a_2d[1][1]      # >> 5

a_2d[1,1]  # >> 5 (same as a_2d[1][1])
```

NumPy Indexing and Selection
(Nested Arrays | 2D+ Arrays)

```
a_2d = np.array([ [1,2,3], [4,5,6], [7,8,9] ])

# >> array( [ [1, 2, 3],
#             [4, 5, 6],
#             [7, 8, 9] ] )

a_2d[:2]

# >> array( [ [1, 2, 3],
#             [4, 5, 6] ] )
```

NumPy Indexing and Selection
(Nested Arrays | 2D+ Arrays)

```
a_2d = np.array([ [1,2,3], [4,5,6], [7,8,9] ])

# >> array( [ [1, 2, 3],
#             [4, 5, 6],
#             [7, 8, 9] ] )

a_2d[:2, 1:]

# >> array( [ [2, 3],
#             [5, 6] ] )
```

NumPy Indexing and Selection (Statement)

AiAT ACADEMY

```
a = np.arange(1,5)
# >> array([1, 2, 3, 4])

a > 2
# >> array([False, False, True, True ], dtype=bool)

a[a > 2]     # >> array([3, 4])

a[a < 2]     # >> array([1])
```

AiAT ACADEMY

NumPy Operation

NumPy Arrays (Operations)

AiAT ACADEMY

```
a = np.arange(4)
# >> array( [0, 1, 2, 3] )
b = np.array([2, 3, 2, 4])

a * b          # array( [ 0, 3, 4, 12] )
b - a          # array( [ 2, 2, 0, 1] )

c = [2, 3, 4, 5]
a * c          # array( [ 0, 3, 8, 15] )
```

NumPy Arrays (Operations)

AiAT ACADEMY

```
a = np.arange(4)
# >> array( [0, 1, 2, 3] )
b = np.array([2, 3, 2, 4])


a - 10       # array( [ -10, -9, -8, -7] )
np.exp(a)    # array([ 1., 2.718, 7.389, 20.085]])
```

For more operations:

https://docs.scipy.org/doc/numpy/reference/ufuncs.html

NumPy Arrays (Vector Operations)

AiAT ACADEMY

```
# NumPy automatically converts lists
u = [1, 2, 3]                v = [1, 1, 1]
np.inner(u, v)               # 6
np.outer(u, v)
# array([[1, 1, 1],
#        [2, 2, 2],
#        [3, 3, 3]])
np.dot(u, v)                 # 6
```