Data Wrangling with Python

Python & Statistics Bootcamp

NaLette M. Brodnax

The Institute for Quantitative Social Science
Harvard University

June 4, 2018

Set up version control

1. Create a new repository on Github (https://github.com) with a README file
2. Clone the repository on your computer
3. Navigate to the directory on your computer and make a change to `README.md`
4. Check the status of your repository
   ```
   git status
   ```
5. Add the modified file to the staging area
   ```
   git add README.md
   ```
6. Commit the changes with a message
   ```
   git commit -m 'add repository description'
   ```
7. Push your changes to Github (log in required)
   ```
   git push origin master
   ```

Set up Jupyter notebook

1. Launch the `conda` command line interface
2. Navigate to the directory where you cloned the repository from Github
3. Launch Jupyter from the command line: `jupyter notebook`, or from the Anaconda Navigator graphical interface
4. Navigate to the browser where your notebook is running
5. Create a new Python 3 notebook called `DataWrangling`

Working with NumPy

Slicing and indexing

Modifying arrays

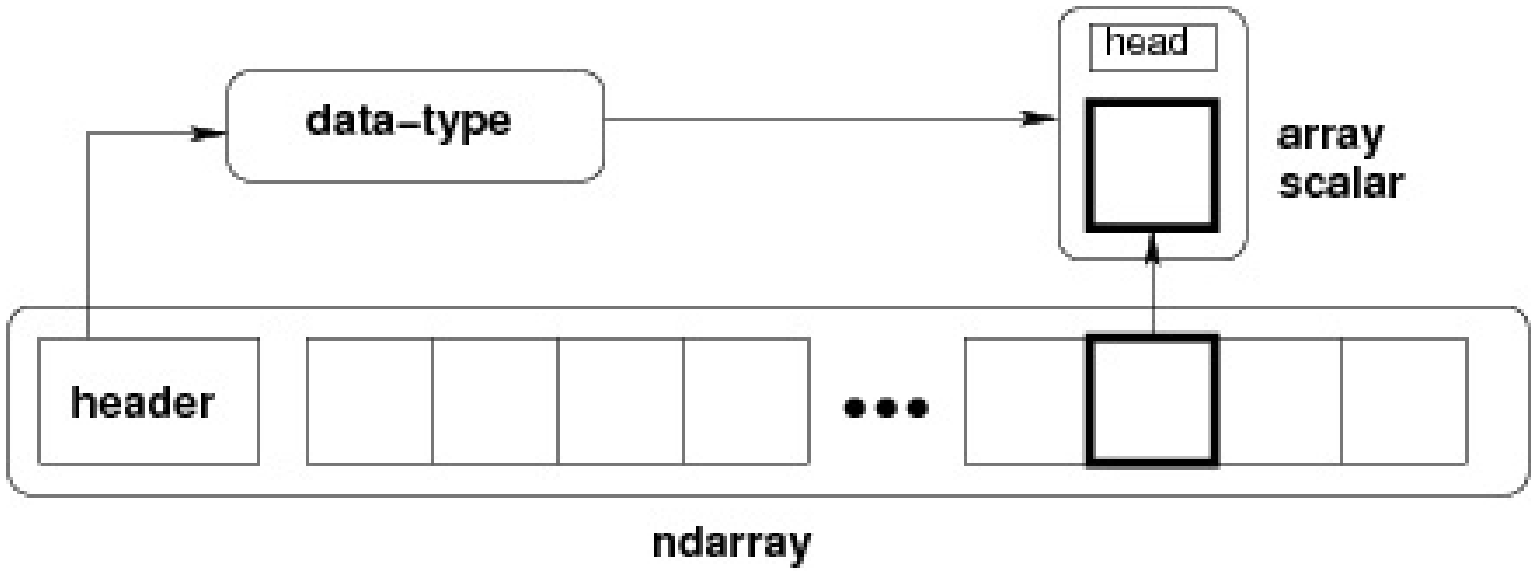
Mathematical operations

The numpy package

NumPy, short for Numerical Python, provides multidimensional objects and routines for processing them

The key object is an n-dimensional array, or **ndarray**

- Contains elements of the same data type (**dtype**)
- Uses a zero-based index



Defining arrays

```
import numpy as np

hello = 'Hello, world.'
print(hello)
```

```
# one-dimensional array
arr1 = np.array([1, 2, 3])
print(arr1)
```

Defining arrays

```
import numpy as np

hello = 'Hello, world.'
print(hello)
```

```
# one-dimensional array
arr1 = np.array([1, 2, 3])
print(arr1)
```

Activity: Create a two-dimensional array called `arr2` with two elements in each sub-array.

Array attributes

```
# specify the number of dimensions
arr3 = np.array([1, 2, 3, 4, 5, 6], ndmin=3)
print(arr3)
```

```
# specify the data type
arr_float = np.array(arr3, dtype=float)
print(arr_float)
```

Array attributes

```
# specify the number of dimensions
arr3 = np.array([1, 2, 3, 4, 5, 6], ndmin=3)
print(arr3)
```

```
# specify the data type
arr_float = np.array(arr3, dtype=float)
print(arr_float)
```

Activity: Create a one-dimensional array called `arr_bool` with values `[0,1,0,1]` and data type `bool`.

Changing an array attribute

Use `.shape` to access or change the dimensions of an array

```
# get the array dimensions
print(arr3)
print(arr3.shape)

# resize an array using the .shape attribute
arr4 = np.array(arr2)
print(arr4)
print(arr4.shape, '\n')

arr4.shape = (4,1)
print(arr4)
print(arr4.shape)
```

Changing an array attribute

Use `.shape` to access or change the dimensions of an array

```
# get the array dimensions
print(arr3)
print(arr3.shape)

# resize an array using the .shape attribute
arr4 = np.array(arr2)
print(arr4)
print(arr4.shape, '\n')

arr4.shape = (4,1)
print(arr4)
print(arr4.shape)
```

Activity: Use the `.reshape()` method to create a 4x1 array from `arr2`.

Activity: useful functions

1. Review the documentation for `np.arange()`, `np.linspace()`, and `np.asarray()`.
2. Create an array called `x` using `np.arange()` with a start value of 1 and a stop value of 20.
3. Create an array called `y` using `np.linspace()` with a start value of 1 and a stop value of 20. Does `y` have the same length as `x`? Why or why not?
4. Create an array called `xnum_arr` from a 4-element list called `xnum`.

Working with NumPy

Slicing and indexing

Modifying arrays

Mathematical operations

Slicing a 1D array

Create a **slice** using colons → `[start:stop:step]`

```
a = np.arange(5, 30, 2)
print(a)
print(a[:5])
print(a[2:5])
print(a[7:])
```

Slicing a 1D array

Create a **slice** using colons → `[start:stop:step]`

```
a = np.arange(5, 30, 2)
print(a)
print(a[:5])
print(a[2:5])
print(a[7:])
```

Activity: What is the result of the following?

```
print(a[::-1])
```

Slicing a 2D array

```
b = np.array([[1,2,3,4,5],
              [2,4,6,8,10],
              [3,6,9,12,15],
              [5,10,15,20,25],
              [6,12,18,24,30]])
print(b, '\n')
print(b[:1], '\n')
print(b[1:], '\n')
print(b[::2])
print(b[:1, :2])
```

Working with NumPy

Slicing and indexing

Modifying arrays

Mathematical operations

Modifying an array element

Arrays have a fixed type, so if you assign a value with a different type, you may end up with unexpected behavior

```
k = np.array([[1,2,3,4],
              [2,4,6,8],
              [3,6,9,12]])
ksub = k[:3, :2]
print(k)
print(ksub)

# modify a value
ksub[1, 0] = 100
print(k)
print(ksub)
```

Slicing an array returns a *view* rather than a *copy*; use `.copy()` to create a new array.

Combining arrays

```
m = np.array([9,18,27,36])
n = np.array([10,20,30,40])
o = np.array([7,14,21,28])

print(np.concatenate([m,n]))
print(np.concatenate([m,n,o]))
```

Combining arrays

```
m = np.array([9,18,27,36])
n = np.array([10,20,30,40])
o = np.array([7,14,21,28])

print(np.concatenate([m,n]))
print(np.concatenate([m,n,o]))
```

Activity: What is the result of the following?

```
print(np.vstack([m,n,o]))
print(np.hstack([m,n,o]))
```

Working with NumPy

Slicing and indexing

Modifying arrays

Mathematical operations

Arithmetic is faster with arrays

```
r = np.arange(8)
print("r =", r)
print("r + 10", r + 10)
print("r + 10", r / 2)

s = np.arange(100)

def div_by_2(array):
    v = np.empty(len(array))
    for i in range(len(array)):
        v[i] = i / 2
    return v
```

Summary statistics

```
X = np.random.random(1000)
print(np.sum(X))
x_range = (X.min(), X.max())
print(x_range)

Y = np.random.random((3,4))
print(Y)
print(Y.sum(axis=0)) # columns
print(Y.sum(axis=1)) # axis is the dimension to
    collapse
```

Summary statistics

```
X = np.random.random(1000)
print(np.sum(X))
x_range = (X.min(), X.max())
print(x_range)

Y = np.random.random((3,4))
print(Y)
print(Y.sum(axis=0)) # columns
print(Y.sum(axis=1)) # axis is the dimension to
    collapse
```

Activity: Create a random sample of 100 integers ranging from 1 to 10 (HINT: `np.random.randint()`. Using array functions from `numpy`, find the mean, variance, standard deviation, and median of the sample.

Questions?