# DATA 133 - Introduction to Data Science I

Instructor: Renzhi Cao
Computer Science Department
Pacific Lutheran University

## Announcements

- In-class exercise for day 18 will be due today. Continue to work on Project 2.
- Check hint on course website for data visualization
- Today we are going to learn Linear Algebra and statistics in Python

## Reference book

- Data Science from Scratch - First Principles with Python. O'Reilly Media, 2015.

- Reading (Data Science from Scratch):
  - Read chapter 4: Linear Algebra
  - Read chapter 5: Statistics
  - Read chapter 6: Probability

Pictures from: https://www.google.com/search?

## Introduction

- Linear Algebra

*Is there anything more useless or less useful than Algebra?*
—Billy Connolly

## Linear Algebra

- Branch of Mathematics that deals with Vector Spaces.
- Vector Space? What is a Vector?

- Formal: “a quantity having direction as well as magnitude, especially as determining the position of one point in space relative to another.”
- Informal: Point in a finite-dimensional space. They can be added together and multiplied by scalars (numbers)
- Example: A vector in a 3-d space: Age, Height, Weight
- A vector in a 4-d space: Exam1, Exam2, Exam3, Exam4

- What to represent vector in R? How about Python?

## Linear Algebra

- List = [10, 20, 30]

```
def vector_add(v, w):
    """adds corresponding elements"""
    return [v_i + w_i  for v_i, w_i in zip(v, w)]

def scalar_multiply(c, v):
    """c is a number, v is a vector"""
    return [c * v_i for v_i in v]

def scalar_multiply(c, v):
    """c is a number, v is a vector"""
    return [c * v_i for v_i in v]

def dot(v, w):
    """v_1 * w_1 + ... + v_n * w_n"""
    return sum(v_i * w_i  for v_i, w_i in zip(v, w))
```

Not very practical to use lists!

- Cannot perform operations as vectors!

## Linear Algebra

```
import numpy as np

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

b = np.array([5,2,6],float)      OR      b = np.array([5,2,6])

print a +b

print a * 5
```

## Linear Algebra

Python does not have 2-d arrays, but we could use vectors to represent them.

```
friendship = np.array([[0,0,1],[1,0,1],[1,0,0]])
friendship2 = np.array([[0,0,1],[1,0,1],[1,0,0]])
Test =friendship + friendship2
```

## Numpy details

http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf

Also on the course website as a pdf file

## Practice

- Explore Numpy document with your partner.
- Read the data.txt and load the first column as list 1, and the second column as list2
- Use Numpy to calculate the mean, min, max of all data for each list. Write function to do that.
- Use Numpy to do a vector add, subtract, and multiply of this two lists.

## Break

## Statistics

This is a GIGANTIC topic.

In this class, we will just cover the surface.

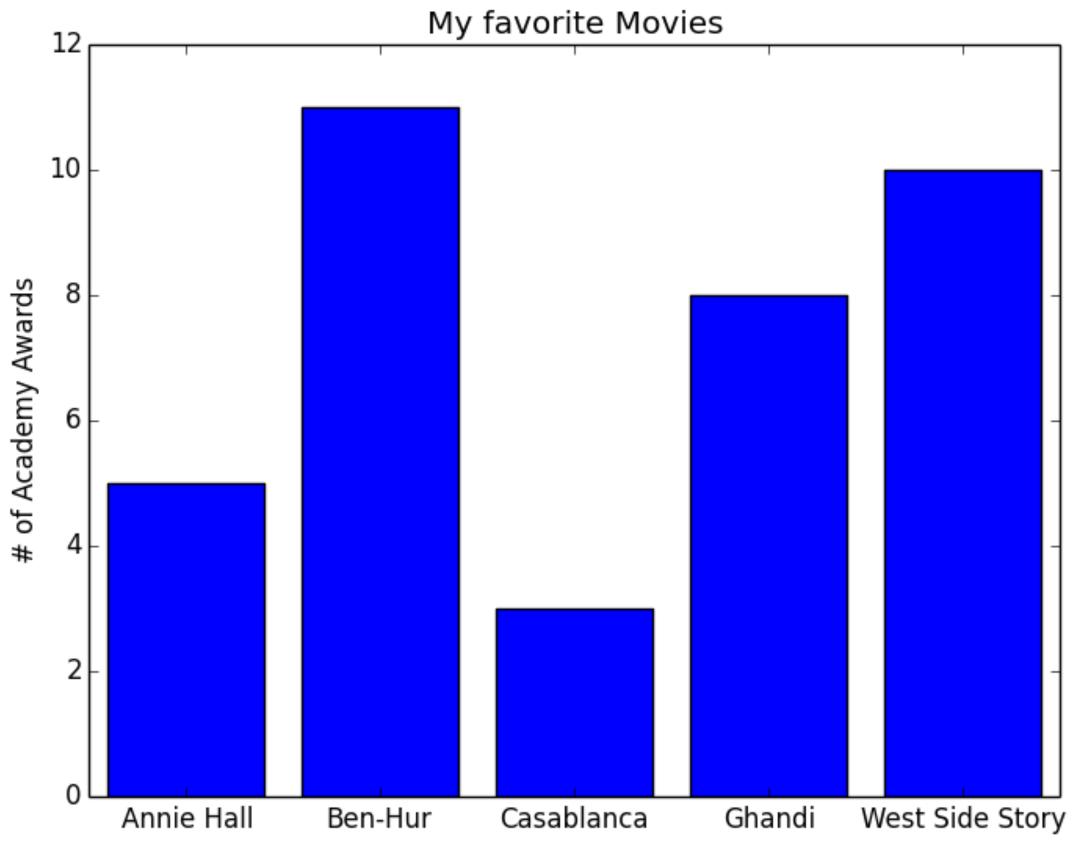
Covering basic concepts that we will use in the future.

"Facts are stubborn, but statistics are more pliable"

Mark Twain

## Statistics

- What do you want to know from this picture?



## Measurements of central tendency and Dispersion

- Mean
- Median
- Mode
- Min and Max value
- Percentiles

- Range
- Variance
- Standard Deviation

## Statistics

- movies = [3,5,2,4,7]
- How to write a function to calculate the mean?

```
def mymean(v):
    total = 0
    for i in range(len(v)):
        total = total + v[i]
    result = total/len(v)
    return result

mymean(movies)
```

Simple way, use numpy, numpy.mean(movies)

## Statistics

- movies = [3,5,2,4,7]
- How about median?

```
def median(v):
     """finds the 'middle-most' value of v"""
     n = len(v)
    sorted_v = sorted(v)
    midpoint = n // 2
    if n%2==1:
         # if odd, return the middle value return sorted_v[midpoint]
    else:
        # if even, return the average of the middle values
        lo = midpoint - 1
        hi = midpoint
        return (sorted_v[lo] + sorted_v[hi]) / 2

median(num_friends) # 6.0
```

Simple way, use numpy, numpy.median(movies)

## Statistics

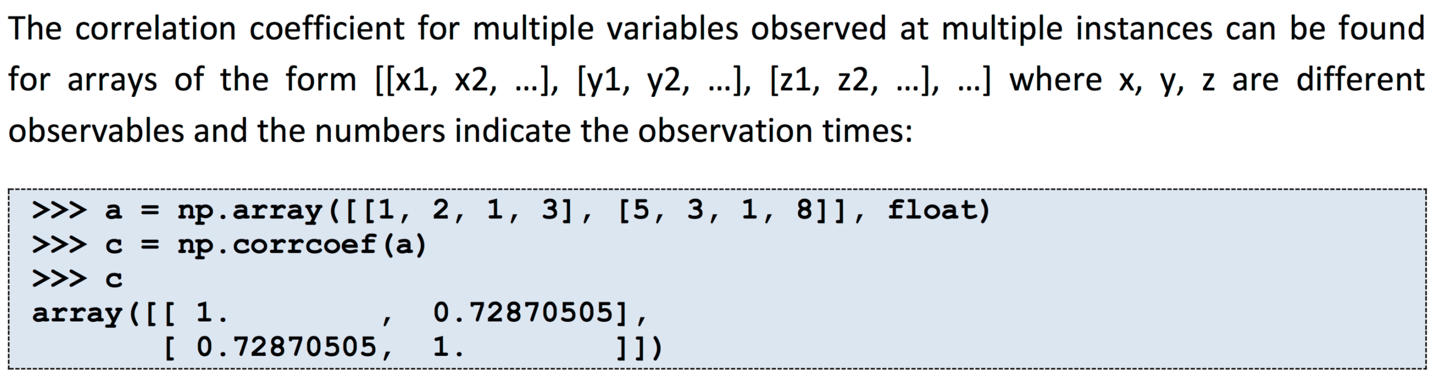
Example:

Statistics is a module only supported in 3.0
In 2.7 we can use numpy to calculate those values or create our own functions.

```
a = np.array([[0,2], [3, -1], [3, 5]], float)
a.mean(axis=0)       # will get [2,2]
a.mean(axis=1)       # will get [1,1,4]
l = [1,3,6,7,8]
np.median(l)         # get the median of l
```

## Correlation

- In numpy, use numpy.corrcoef.



The correlation coefficient for multiple variables observed at multiple instances can be found for arrays of the form [[x1, x2, …], [y1, y2, …], [z1, z2, …], …] where x, y, z are different observables and the numbers indicate the observation times:

```
>>> a = np.array([[1, 2, 1, 3], [5, 3, 1, 8]], float)
>>> c = np.corrcoef(a)
>>> c
array([[ 1.        ,  0.72870505],
       [ 0.72870505,  1.        ]])
```

- A correlation of -1 means perfect anti-correlation
- A correlation of 1 means perfect positive correlation

- Indicates a relationship between the two parameters, lists, etc.
- Correlation does not imply causation!!!! Will be saying this many many times

## Simpson’s paradox

Confounding variables (**confounding variable** is an unmeasured third variable that influences both the supposed cause and the supposed effect.) ?

Number of friends for a group of scientists

| Coast | # of Members | Avg.# of friends |
|---|---|---|
| West | 101 | 8.2 |
| East | 103 | 6.5 |

## Simpson’s paradox

| Coast | Degree | # of members | Avg. # of friends |
|---|---|---|---|
| West | Ph.D. | 35 | 3.1 |
| East | Ph.D. | 70 | 3.2 |
| West | No Ph.D. | 66 | 10.9 |
| East | No Ph.D. | 33 | 13.4 |

Assumes all other values are equal
Always look at all confounding values!

## Practice

Apply the things you learned in today’s class to your project 2!

Any question or demo needed for project 2?

## Homework of R

1. Read book chapter 4 - 5
2. Continue to read chapter 6 if you have time
3. No homework for today. Continue to work on Project 2.