Data Science I

Introduction to probability and statistics

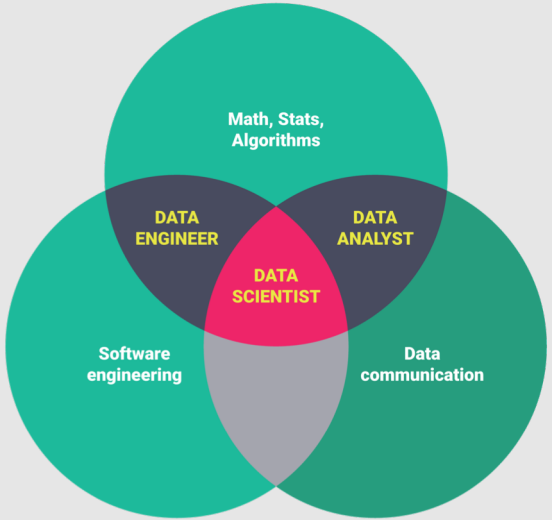
What is data science?

Data science is a subject about making decisions and predictions.

- Explaining what is going on by processing history of the data
- Using algorithms to identify the occurrence of a particular event in future

It is a blend of various disciplines.

- Statistics
- Machine learning
- Economics
- Finance
- ...



Segment overview

Probability and statistics (1 week)

Machine learning (2 weeks)

- Learn by doing, with OOP
- New and useful algorithms
  - Monte Carlo simulation
  - k-means clustering
  - K-nearest neighbor classification
  - Linear / perceptron classifier

Agenda

- Warming up questions
- Probability
- Monte Carlo simulation
- Normal distribution and law of large numbers

Odd Question #1

- Throw a fair coin once. Head or Tail? Which result is more likely to happen?

- Throw a fair dice once. Which number will face up? Which result is more likely to happen?

Odd Question #2

- Draw a card from a deck (52)
- How likely you get a number greater than 10 (i.e. J, Q, K)?

Odd Question #3

A pair of fair dices (with 6 faces)

- Throw of 7: if adds up to 7
- Throw of 6: if adds up to 6

Question:

- Which is more likely?

7: [1,6], [2, 5], [3, 4], [4, 3], [5, 2], [6, 1] 6/36

6: [1, 5], [2, 4], [3, 3], [4, 2], [5, 1] 5/36

Odd Question #3

A pair of fair dices (with 6 faces)

- Throw of 7: if adds up to 7
- Throw of 6: if adds up to 6

Question:

- Which is more likely?

Odd Question #4

6/49 (government-run lottery)

- 6 lucky numbers out of 1 to 49, without replacement
- Prizes divided among those who choose the lucky numbers

- Two sequences of lucky numbers:
  - A: 1, 2, 3, 4, 5, 6
  - B: 28, 21, 33, 17, 4, 8

- Question: which one do you prefer?

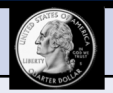
Probability

Probability

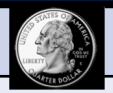
- An **experiment** is a procedure that yields one of a given set of possible outcomes.
- The sample space of the experiment is the set of possible outcomes.
- An event is a **subset** of the sample space.

Let S be a finite nonempty sample space of equally likely outcomes, and E is an event (i.e., a subset of S), then the *probability of* E *is* $p(E) = \frac{|E|}{|S|}$. *Obviously, we have* $0 \leq p(E) \leq 1$.

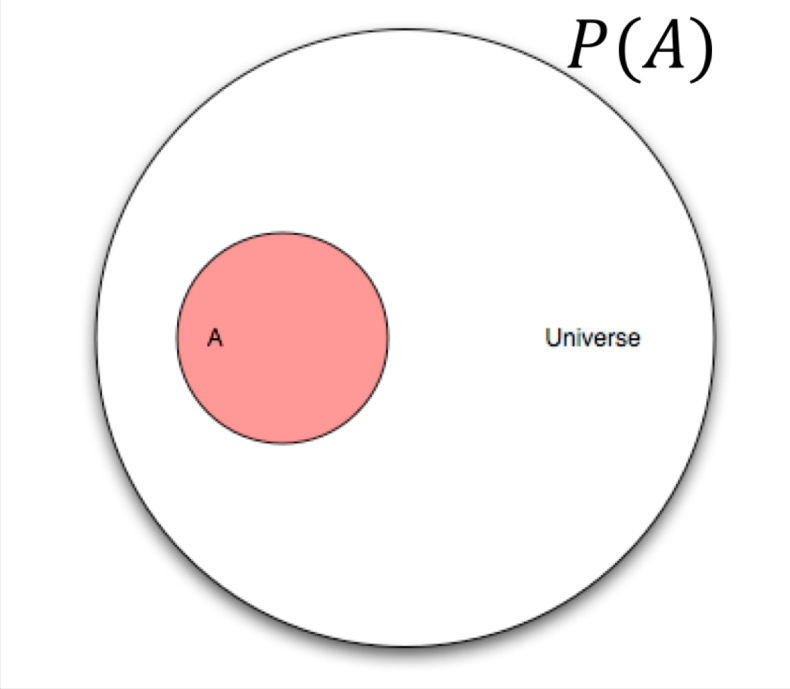
Probability

Experiment	Event X	Probability
Throw a coin 	Head	
	Tail	
Throw a dice 	1	
	2	
	3	
	4	
	5	
	6	

Probability

Experiment	Event X	Probability
Throw a coin 	Head	0.5
	Tail	0.5
Throw a dice 	1	1/6
	2	1/6
	3	1/6
	4	1/6
	5	1/6
	6	1/6

Probability “universe” : {individual event}



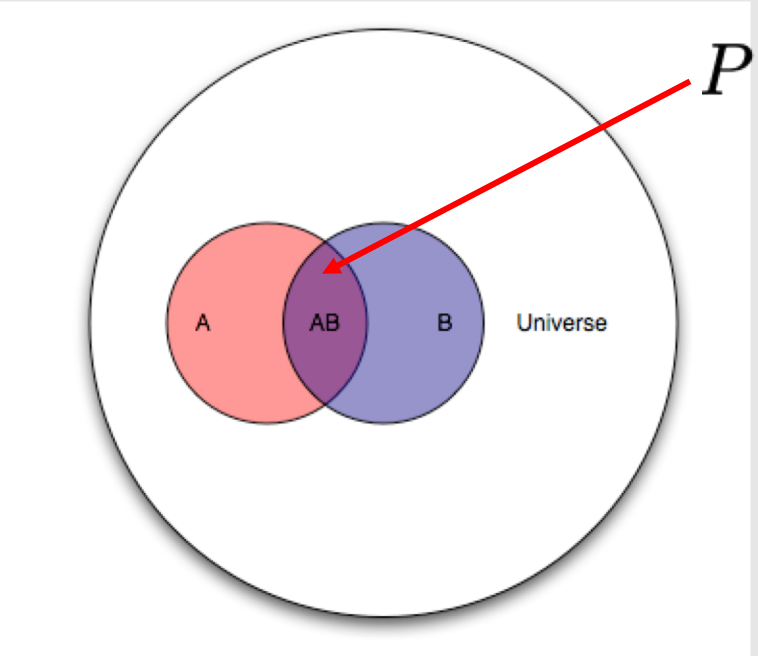
When throw a dice once and check # of dots facing up:

- **universe** : 1, 2, 3, 4, 5, 6
- **event A** : greater than 4

“Universe” is the sample space; event A can be a set of several individual event.

Joint probability

- Joint events: the intersection of two events.



$$P(A \cap B) = P(AB)$$

Question:
If P(A) = 0.3 and P(B) = 0.2,
Can we say P(AB) < 0.2?

No. If B⊂A, P(AB) = P(B).

Joint probability [wiki page of joint distribution]

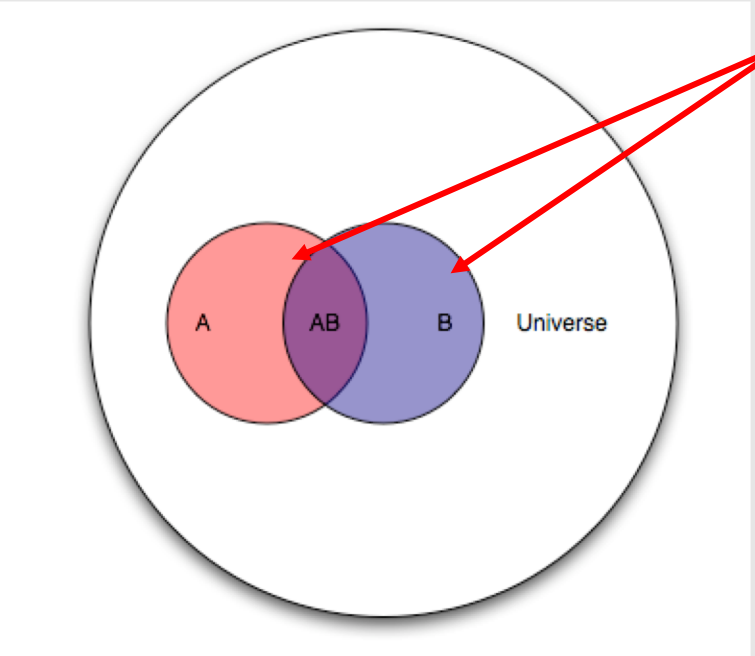
- Event A: outcome is an even number
- Event B: outcome is a prime number
- What's the probability that both A and B are true?

	1	2	3	4	5	6
A: even		1		1		1
B: prime		1	1		1	

$$\mathrm{P}(A = 1, B = 1) = P\{2\} = \frac{1}{6}.$$

Unions of events

- The union of two events



$$P(A \cup B)$$

$$P(A \cup B) = P(A) + P(B) - P(AB)$$

Which one is a joint events/union of events?

The probability that tomorrow...

1. The gym will be closed **or** the cafeteria will be closed
2. both cafeteria **and** gym will closed
3. The gym will be closed, given today it is closed

Conditional probability wiki page of joint distribution

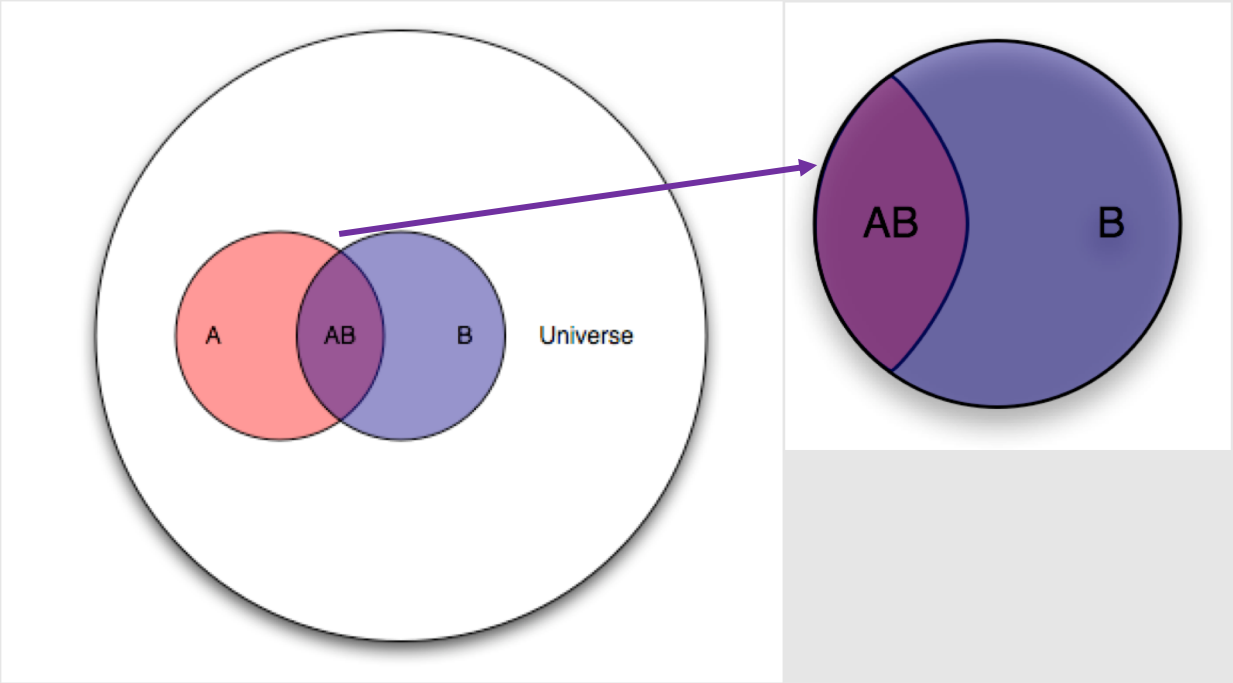
The conditional probability of A given B is the probability of A when B has occurred, denoted by $P(A|B)$. It is defined as

$$P(A|B) = \frac{P(AB)}{P(B)} = \frac{\text{number of occurrence of both A and B}}{\text{number of occurrence of B}}$$

Examples: The probability of

- “x < 3, given that x is a positive odd number < 10 ”
- “I’ll get up at 7 am tomorrow, given I’m never up at 1 pm”

Conditional probability



Question:

- $P(A) = 0.3$
- $P(B) = 0.2$

Can we say

- $P(B|A) < 0.2$

No, because P(AB) ≤ 0.2.

$$P(A|B) = \frac{P(AB)}{P(B)} \iff P(AB) = P(A|B)P(B)$$

Conditional probability

	Have pets	Do not have pets	Total
Male	0.08	0.42	0.5
Female	0.12	0.38	0.5
Total	0.2	0.8	

- For a randomly selected person, given that he/she owns a pet, what is the probability that this person is male?

$$P(M|P) = \frac{P(M \cap OP)}{P(OP)} = \frac{0.08}{0.2} = 0.4$$

Summary of probability

See [wiki page of probability]

Event	Probability
A	$P(A) \in [0, 1]$
not A	$P(A^c) = 1 - P(A)$
A or B	$P(A \cup B) = P(A) + P(B) - P(A \cap B)$ $P(A \cup B) = P(A) + P(B)$ if A and B are mutually exclusive
A and B	$P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A)$ $P(A \cap B) = P(A)P(B)$ if A and B are independent
A given B	$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)}$

Coding exercise 1: implement a dice class

```
10 class Dice:
11     def __init__(self, sides=2):
12         self.n_sides = sides
13         self.bounds = [x/sides for x in range(0, sides)]
14         self.bounds.append(1.0)
15         self.point = None
16         self.lands = 0
17
18     def set_bounds(self, r):
19         assert len(r) == len(self.bounds)
20         self.bounds = r
21
22     def get_bounds(self):
23         return self.bounds
24
25     def roll(self):
26 # replace the following line with you code
27 # it should set self.point as a random var in [0, 1]
28 # return which side the dice lands on
29         return dice_helper.roll(self)
```

- N-sided
- Maintain a list of N-partitioned range
- Roll to land in one of the partition

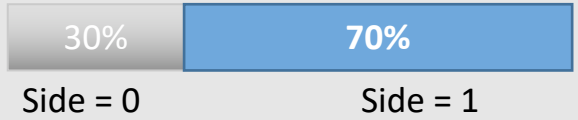
self.bounds = [0, ½, 1]

- Make it biased with set_bounds(self, range)

Coding exercise 1: implement a dice class

Make an arbitrarily biased dice

```
31 if __name__ == "__main__":
32     d = Dice()
33     ''' make a biased dice '''
34     d.set_bounds([0.0, 0.3, 1.0])
35     ones = 0
36     num_rolls = 10
37     for i in range(num_rolls):
38         ones += d.roll()
39     print("the dice is:", ones/float(num_rolls))
```




```
In [34]: d2 = Dice()

In [35]: d2.set_bounds([0.0, 0.7, 1.0])

In [36]: r = [d2.roll() for i in range(50)]

In [37]: sum(r)/50
Out[37]: 0.22
```

Monte Carlo simulation

Monte Carlo simulation

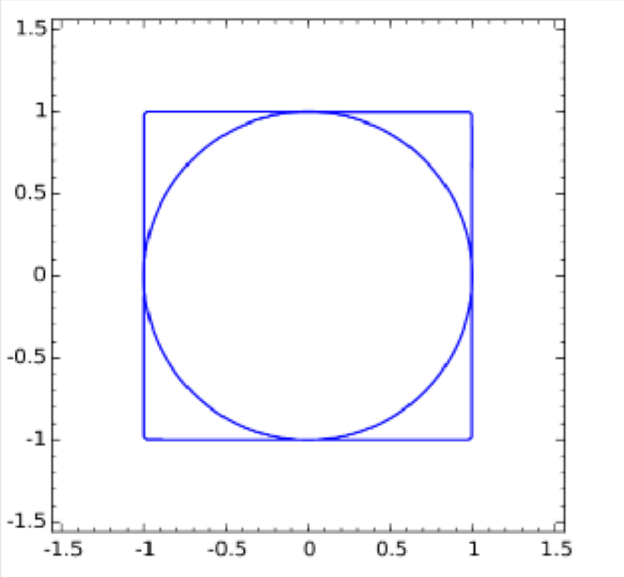
- A technique used to approximate the probability of an event by running the same simulation multiple times and averaging the results.

Measuring π

What you have:

- A round dish centered at 0,0 with radius 1
- A square with corners at [-1, -1], [-1, 1], [1, 1] and [1, -1]

- $\frac{S_{circle}}{S_{square}}$ tells something about π
- Can you figure it out?

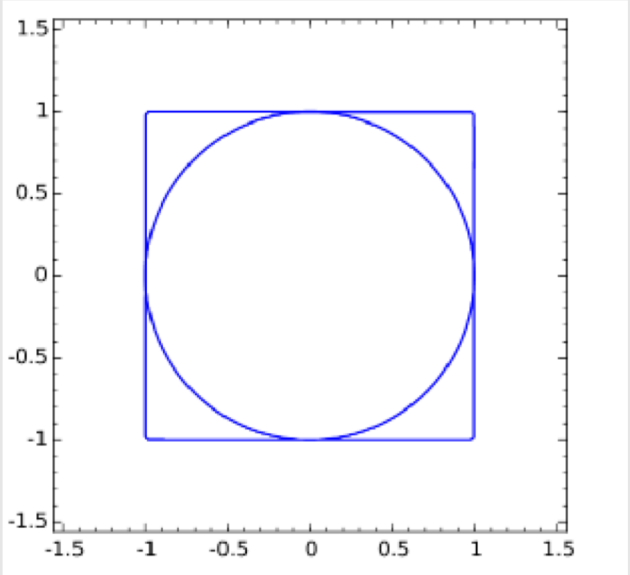


Measuring π

$$S_{square} = (2r)^2 = 4r^2$$

$$S_{circle} = \pi r^2$$

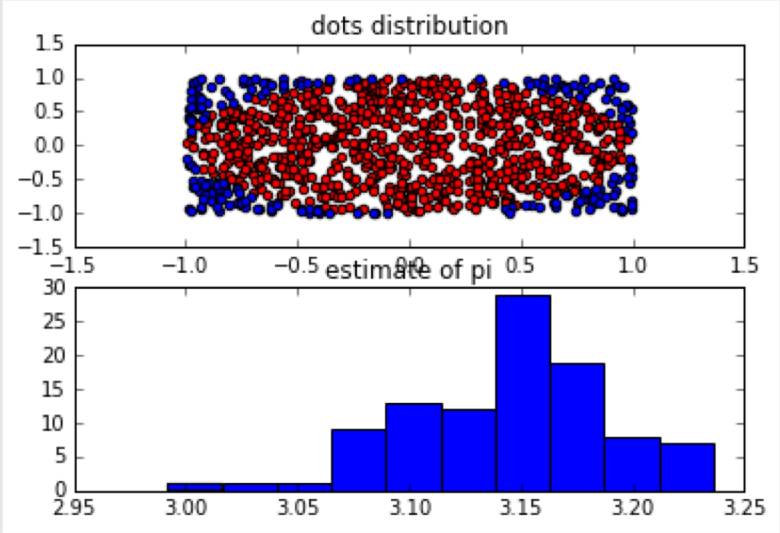
$$\pi = 4 \times (S_{circle} / S_{square})$$



Estimate $\frac{S_{circle}}{S_{square}}$ as number of times random dots land within the circle

Coding exercise 2: estimate π

```
# compute the Eucliear distance of two vectors
def comp_dist(a, b):
    assert len(a) == len(b)
    d = 0
    for i in range(len(a)):
        d += (a[i] - b[i]) ** 2
    return math.sqrt(d)

def estimate_pi(num_points):
# replace the following line with your code
# use math.uniform(-1, 1) twice to get cordinate of a random dot
# record how many times dots are within unit circle
# should return:
# - an estimate of pi
# - the x-axis cordinates list for those within the unit circle
# - the y-axis cordinates list for those within the unit circle
# - the x-axis cordinates list for those outside the unit circle
# - the y-axis cordinates list for those outside the unit circle
    return montecalo_helper.estimate_pi(num_points)

results = []
# total number of random dots per trial
num_points_per_estimate = 1000
# perform multiple trails
num_trials = 100
for i in range(num_trials):
    est_pi, in_x, in_y, out_x, out_y = \
        estimate_pi(num_points_per_estimate)
    results.append(est_pi)

print("estimated pi: ", sum(results)/num_trials)
```

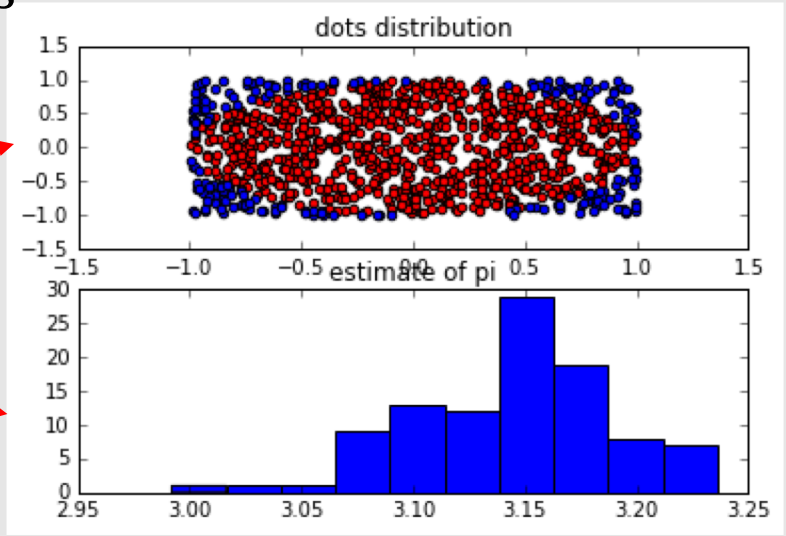


Coding exercise 2: estimate π

Two useful visualization techniques

- Scatter plot and histogram

```
45 # visualize the dots in the last trial
46 plt.subplot(2, 1, 1)
47 plt.title('dots distribution')
48 plt.scatter(in_x, in_y, c='r')
49 plt.scatter(out_x, out_y, c='b')
50
51 # visualize the estimations of all trials
52 plt.subplot(2, 1, 2)
53 plt.title('estimate of pi')
54 plt.hist(results)
```



Normal distribution and law of large numbers

How can you tell if a coin is *fair*?

A fair coin

- A fair coin: P(head) = P(tail) = 0.5
- A single flip is only 1 (for head) or 0 (for tail)

Questions: 100 flips will result in how many heads?

  - 34 or 50 or 55?
  - No exact answer; each try usually has different result.

- Random variable: a variable whose values depend on outcomes of a random phenomenon.
  - counts of head is a random variable
  - counts/100 is also a random variable

Expected value/expectation/population mean

The expectation of a random variable represents the **average** of a large number of observations of the random variable.

Law of large numbers

The average of the results obtained from a large number of trails should be close to the expected value.

- We throw the coins for n times; this process is called block of trials.

We will verify the law of large numbers by conducting a block of trials of coin throw..

Code 3: measure a coin

One block trial of coin throw

```
11 class Block_trial:
12     def __init__(self, sides=2, block_size=10):
13         self.dice = dice_student.Dice(sides)
14         self.block_size = block_size
15         self.outcomes = {}
16         for i in range(sides):
17             self.outcomes[i] = 0
18
19     def run(self):
20 # replace the following line with your code
21 # roll the dice block_size number of times, recode in
22 # the outcome dictionary
23         coin_helper.run(self)
24
25     def get_run_stats(self):
26 # report the statistics in a list
27 # the i-th entry of the list is frequency/percentage
28 # the coin lands on the i-th side
29         return coin_helper.get_run_stats(self)
```

outcomes= {0: 0 , 1:0}

outcomes= {0: 43 , 1:57}

[0.43, 0.57]

```
31 if __name__ == "__main__":
32     a_trial = Block_trial()
33     a_trial.run()
34     print(a_trial.get_run_stats())
```

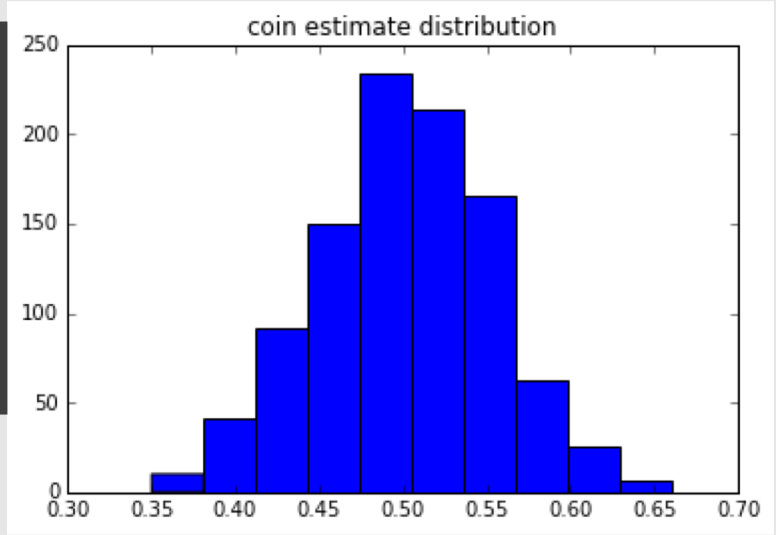
```
In [45]: t = Block_trial(2, 100)

In [46]: t.run()

In [47]: t.get_run_stats()
Out[47]: [0.43, 0.57]
```

Code 3: measure a coin

$$\frac{number\ of\ side\ 0}{total\ number\ of\ trials} = \frac{X_1+\cdots+X_n}{n} = \bar{X}$$

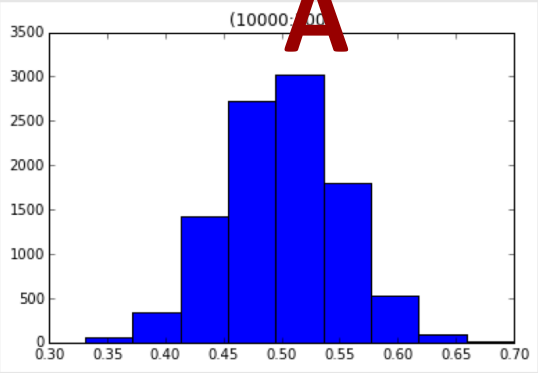
- Make many “block trials”
- What’s the effect of changing block size and number of block trials?

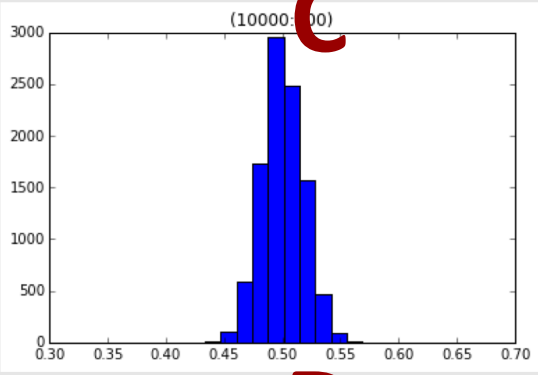
```
36    block_size = 100
37    total_blocks = 1000
38    result = []
39    for i in range(total_blocks):
40        one_trial = Block_trial(block_size=block_size)
41        one_trial.run()
42        result.append(one_trial.get_run_stats()[0])
43
44    plt.hist(result)
45    plt.title('coin estimate distribution')
46    plt.show()
```

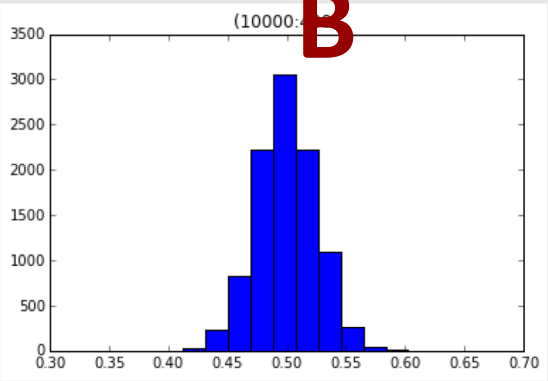


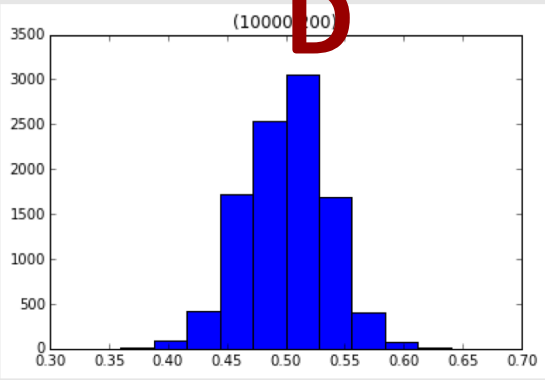
You may change these values to try larger block size and more block_trials (e.g., 400, 5000), to see how the distribution changes.

The effect of block size per trial

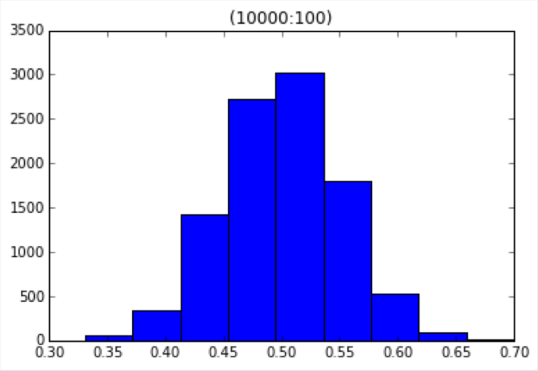


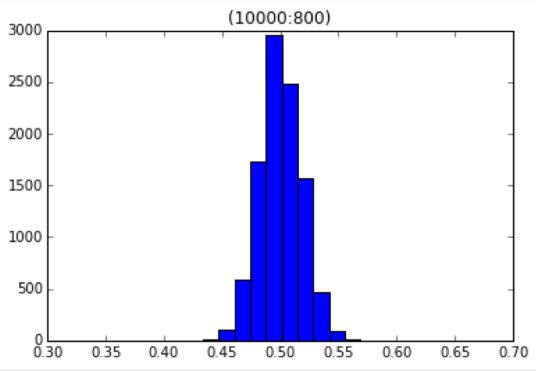


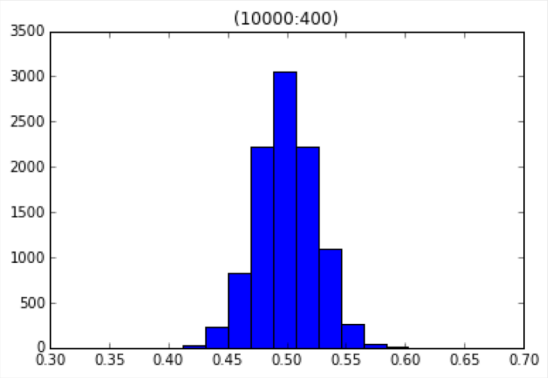


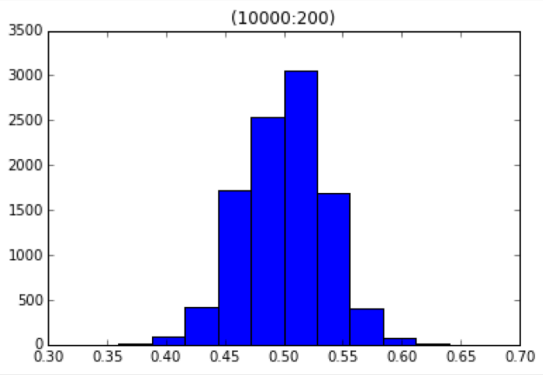


The effect of block size per trial





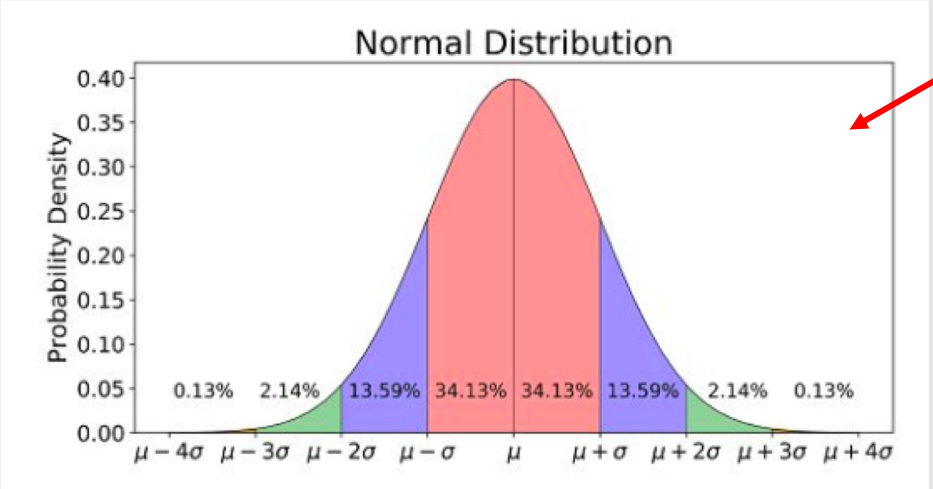




The coin estimation makes a distribution

Normal distribution: $N(\mu, \sigma^2)$

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



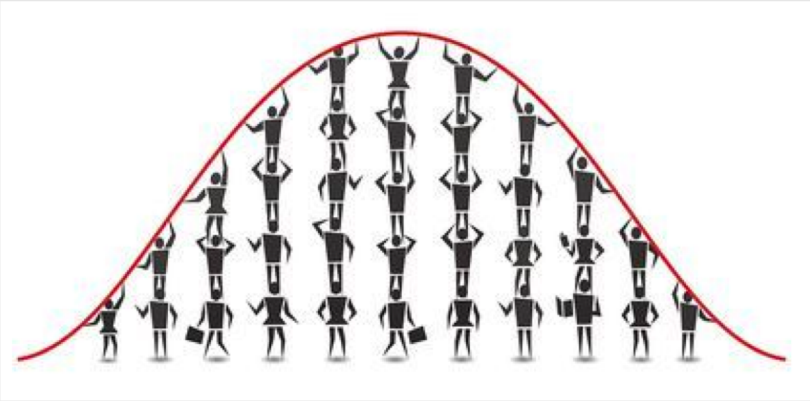
Density of normal distribution

Areas within	Percentage
1σ	68%
2σ	95%
3σ	99%

- μ: the mean of the data points
- σ: the standard deviation; represents the degree of difference between the points and μ.

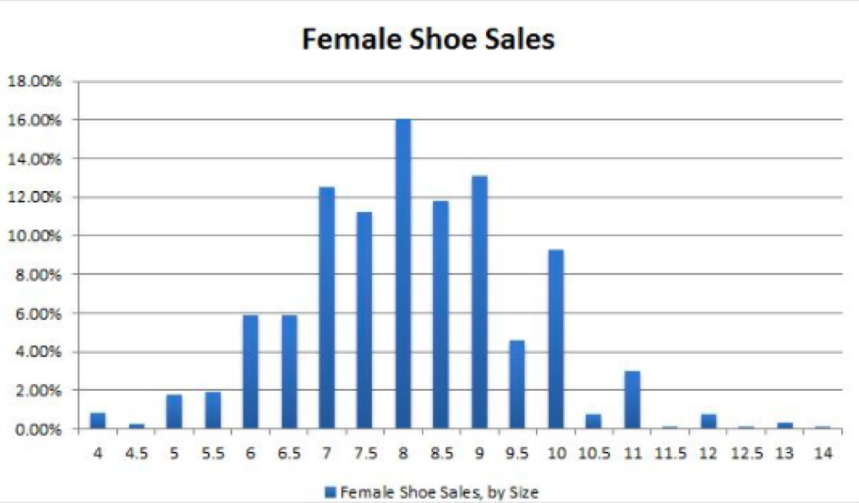
Examples of normal distribution

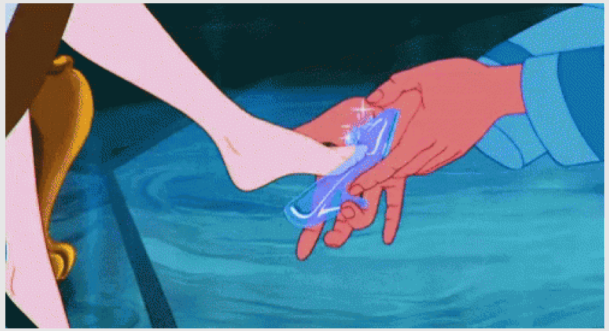
Height of the population



Examples of normal distribution

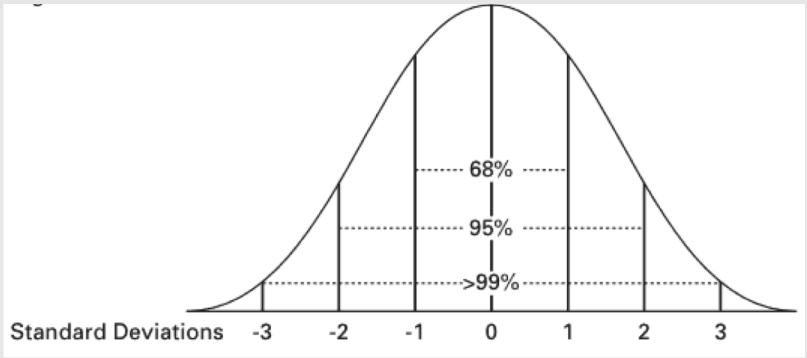
Shoe size





Understand the spread: standard deviation

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

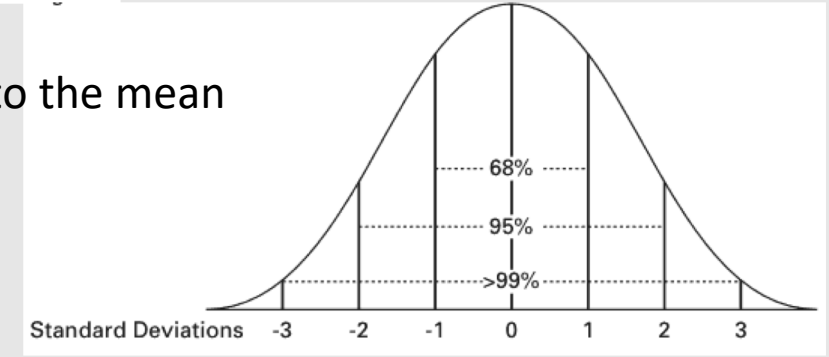


Understand the spread: standard deviation

Bring back to the same unit

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Average distance to the mean



The law of large numbers

When we perform the same experiment a large number of times

- The average of the results obtained from a large number of trials should be close to the expected value (population mean) .
- and will tend to become closer as more trials are performed.

Random sample $X_1, X_2, \ldots X_n$ are drawn from population with mean $E(X_i) = \mu$ and finite variance $Var(X_i) = \sigma^2$, then

$$\bar{X}_n = \frac{1}{n}(X_1 + \cdots X_n)$$

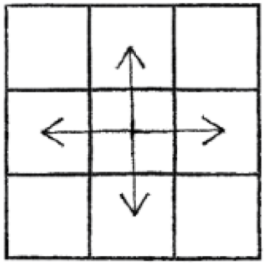
$$E(\bar{X}_n) = \mu$$

$$Var(\bar{X}_n) = \sigma^2/n$$

Summary -- Probability And Statistics

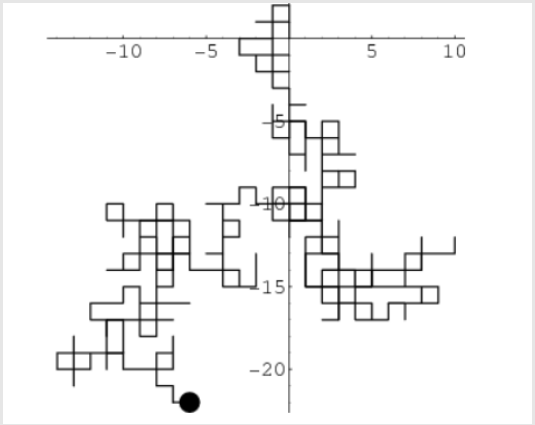
- Probabilities
- Monte Carlo Simulation
- Distribution and Law of large numbers

Random walk



4 possible steps

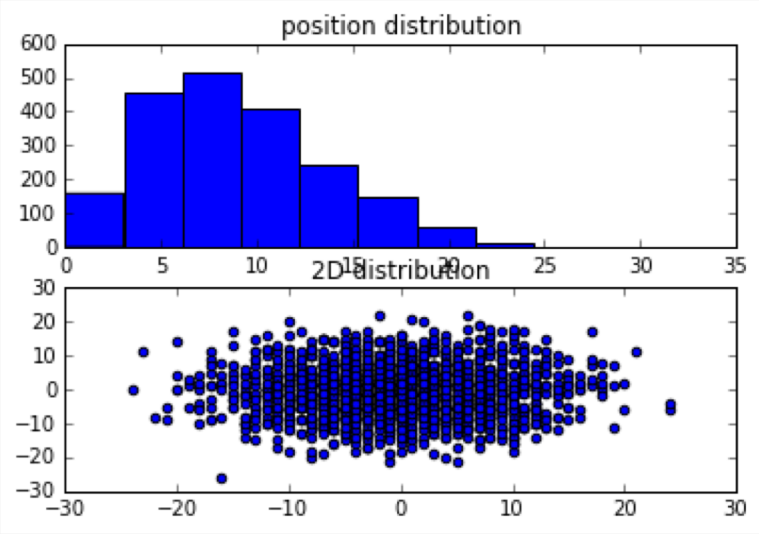
- Each step, one can go any of the four directions.
- The mean distance between starting point and ending point is $n^{0.5}$.



```python
class Walker:
    def __init__(self, directions=4, num_steps=10):
        self.dice = dice_student.Dice(directions)
        self.num_steps = num_steps
        self.x_pos = 0
        self.y_pos = 0

    def set_ranges(self, r):
        self.dice.set_ranges(r)

    def run(self):
# replace the following line with you code
# roll the dice, according to the outcome:
# - go left one step (x -= 1)
# - go right one step (x += 1)
# - go north one step (y += 1)
# - go south one step (y -= 1)
        random_walk_helper.run(self)

    def get_position(self):
        return self.x_pos, self.y_pos

    def get_dist(self):

        def comp_dist(a, b):
            assert len(a) == len(b)
            d = 0
            for i in range(len(a)):
                d += (a[i] - b[i]) ** 2
            return math.sqrt(d)

        return comp_dist([0, 0], [self.x_pos, self.y_pos])
```

Coding exercise 4: random walk

- Simulate with a 4-sided dice
- Mean distance ~ $\sqrt{n}$



Readings

- MIT Ch. 12 (Probability+Stats) 12.1, 12.2, 12.3.1, 12.3.2
- MIT Ch. 13 (Random Walks) 13.1

  (Note: Chapter indices may be different in versions of the MIT book)
- An Introduction of probability and inductive logic (Ian Hacking)