Introduction to Computational Thinking and Data Science

Recitation Hour #04: **Stochastic Thinking & Random Walk**

Activity

- Suppose we use a simulation to simulate a random walk of a class of drunk, returning a collection of actual distances from the origin for a set of trials.

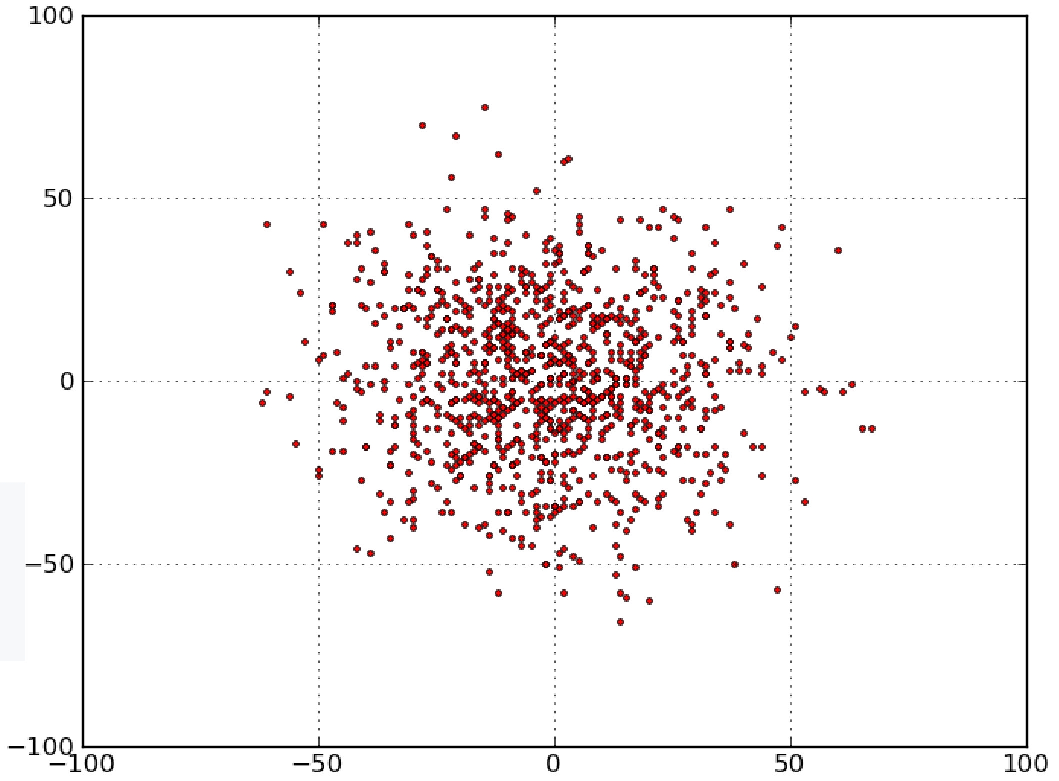
- Each graph below was generated by using one of the above five classes of a drunk (UsualDrunk, ColdDrunk, EDrunk, PhotoDrunk, or DDrunk).

- For each graph, indicate which Drunk class is mostly likely to have resulted in that distribution of distances. Click on each image to see a larger view.

UsualDrunk

The **UsualDrunk** tends to move randomly in any cardinal direction with equal probability, leading to a more evenly distributed random walk compared to some other drunk classes that might exhibit directional biases or varied step sizes.

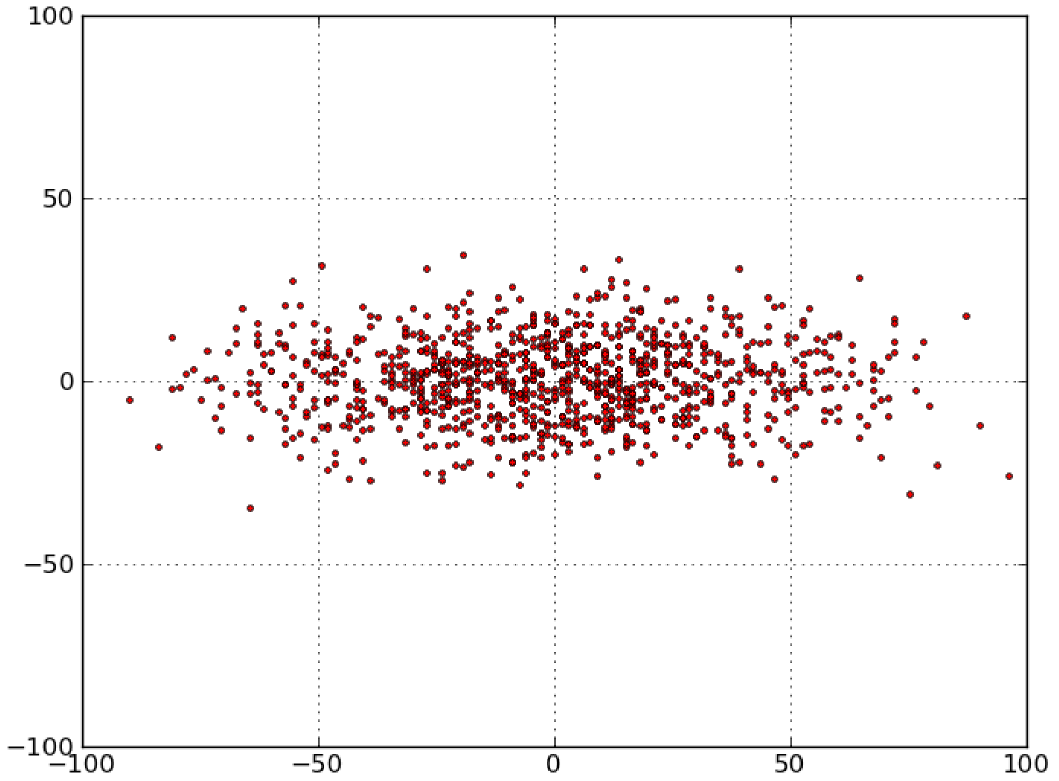
```
class UsualDrunk(Drunk):
    def takeStep(self):
        return random.choice([(0.0,1.0), (0.0,-1.0), (1.0, 0.0), (-1.0, 0.0)])
```



PhotoDrunk

So, the characteristic movement of the PhotoDrunk is due to the defined step choices that emphasize larger movements horizontally, resulting in a pattern where the drunk takes occasional larger steps in specific directions.

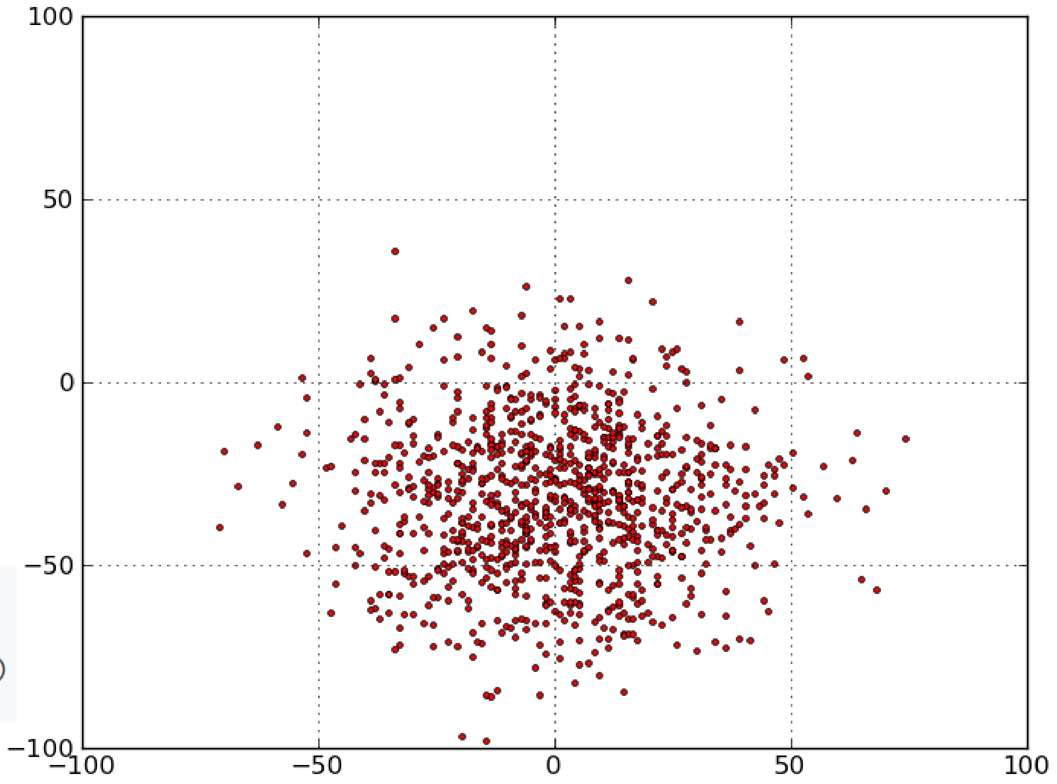
```
class PhotoDrunk(Drunk):
    def takeStep(self):
        return random.choice([(0.0, 0.5), (0.0, -0.5), (1.5, 0.0), (-1.5, 0.0)])
```



ColdDrunk

The ColdDrunk takes **longer steps** in the x-direction than in the positive y-direction, but equal step sizes in both directions along the x-axis.There is a slight bias downward in the y-direction, since the negative y-step (-1.03) is longer than the positive y-step (0.9).

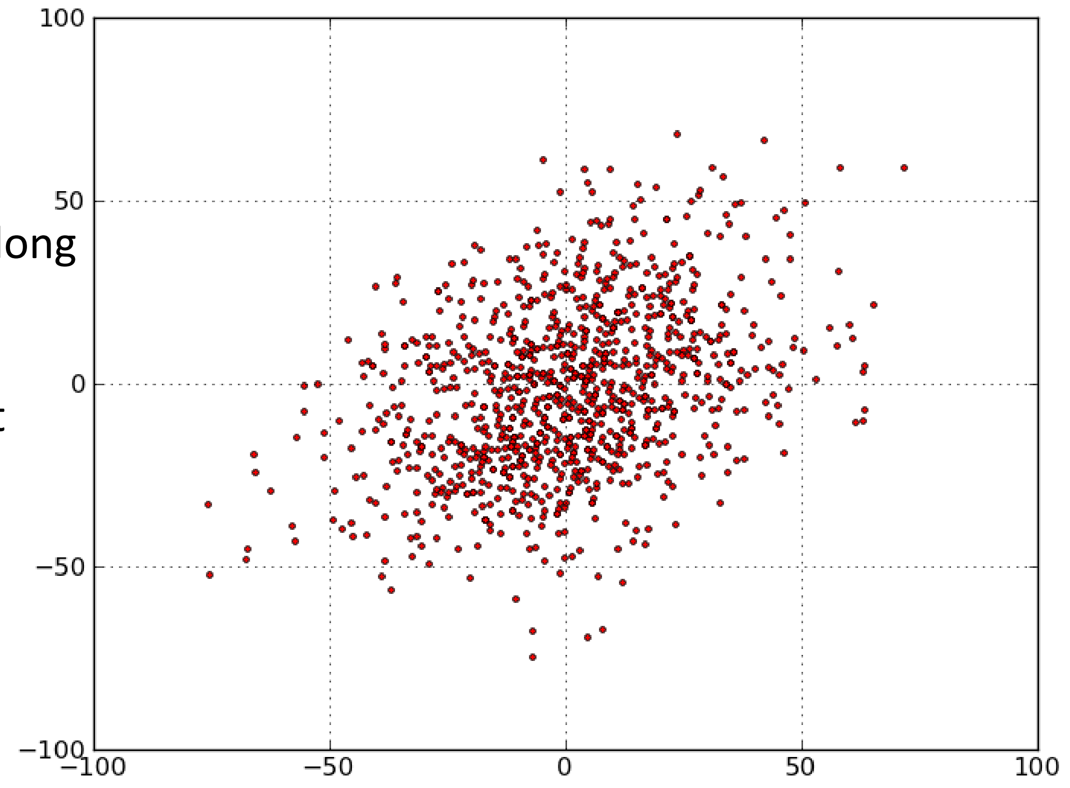
```
class ColdDrunk(Drunk):
    def takeStep(self):
        return random.choice([(0.0,0.9), (0.0,-1.03), (1.03, 0.0), (-1.03, 0.0)])
```



DDrunk

The DDrunk tends to take steps that consistently move it along these fixed diagonals at specific distances from the origin, leading to a distinct pattern in its random walk behavior compared to other drunk types that might exhibit different directional biases or step sizes.

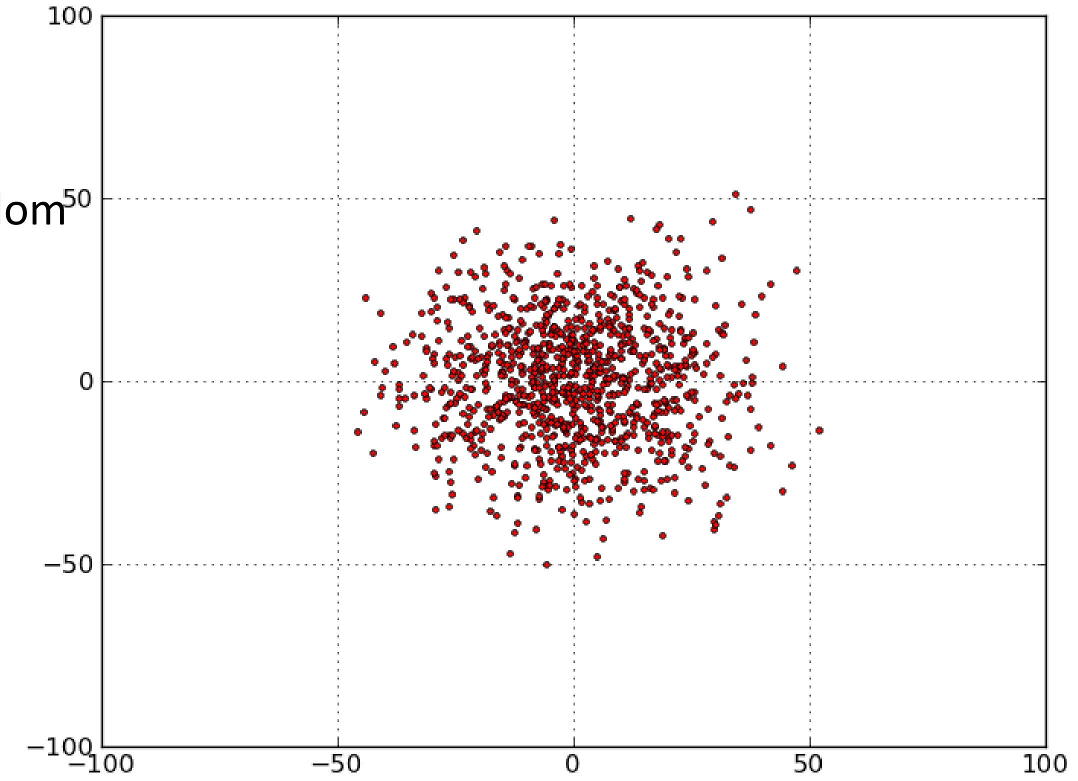
```
class DDrunk(Drunk):
    def takeStep(self):
        return random.choice([(0.85, 0.85), (-0.85, -0.85),
                              (-0.56, 0.56), (0.56, -0.56)])
```



EDrunk

This results in steps that have random angles and lengths, making the EDrunk move in random directions with random step sizes. These steps are uniformly distributed around a circle, with varying distances from the origin due to the random lengths selected for each step.
As a result, the EDrunk tends to move uniformly in random directions and distances, creating a circular pattern with an even distribution of step lengths around the origin

```
class EDrunk(Drunk):
    def takeStep(self):
        ang = 2 * math.pi * random.random()
        length = 0.5 + 0.5 * random.random()
        return (length * math.sin(ang), length * math.cos(ang))
```



Questions