Random Walks and More Plotting, Segment 3

Previous Segments

- Presented a related collection of data abstractions
- Talked about structuring and testing simulations
- Printed the results of some simulations

Iterating Over Styles

```
class styleIterator(object):
    def __init__(self, styles):
        self.index = 0
        self.styles = styles

    def nextStyle(self):
        result = self.styles[self.index]
        if self.index == len(self.styles) - 1:
            self.index = 0
        else:
            self.index += 1
        return result
```

simDrunk

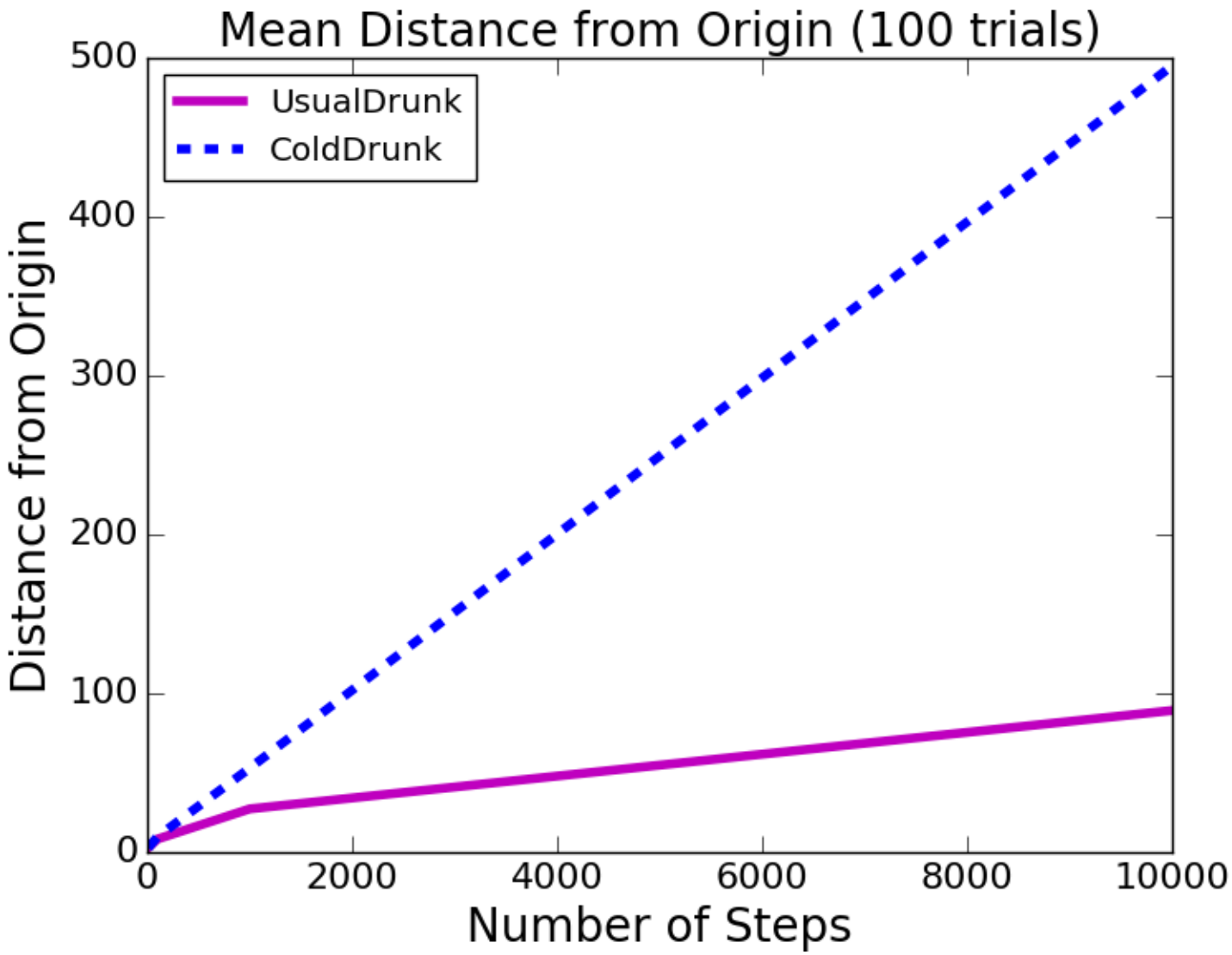
```
def simDrunk(numTrials, dClass, walkLengths):
    meanDistances = []
    for numSteps in walkLengths:
        print('Starting simulation of',
              numSteps, 'steps')
        trials = simWalks(numSteps, numTrials, dClass)
        mean = sum(trials)/len(trials)
        meanDistances.append(mean)
    return meanDistances
```

simAll (new version)

```
def simAll(drunkKinds, walkLengths, numTrials):
    styleChoice = styleIterator(('m-', 'b--', 'g-.'))
    for dClass in drunkKinds:
        curStyle = styleChoice.nextStyle()
        print('Starting simulation of', dClass.__name__)
        means = simDrunk(numTrials, dClass, walkLengths)
        pylab.plot(walkLengths, means, curStyle,
                   label = dClass.__name__)
    pylab.title('Mean Distance from Origin ('
                + str(numTrials) + ' trials)')
    pylab.xlabel('Number of Steps')
    pylab.ylabel('Distance from Origin')
    pylab.legend(loc = 'best')

numSteps = (10,100,1000,10000)
simAll((UsualDrunk, ColdDrunk), numSteps, 100)
```

Distance Trends



Getting Ends of Multiple Walks

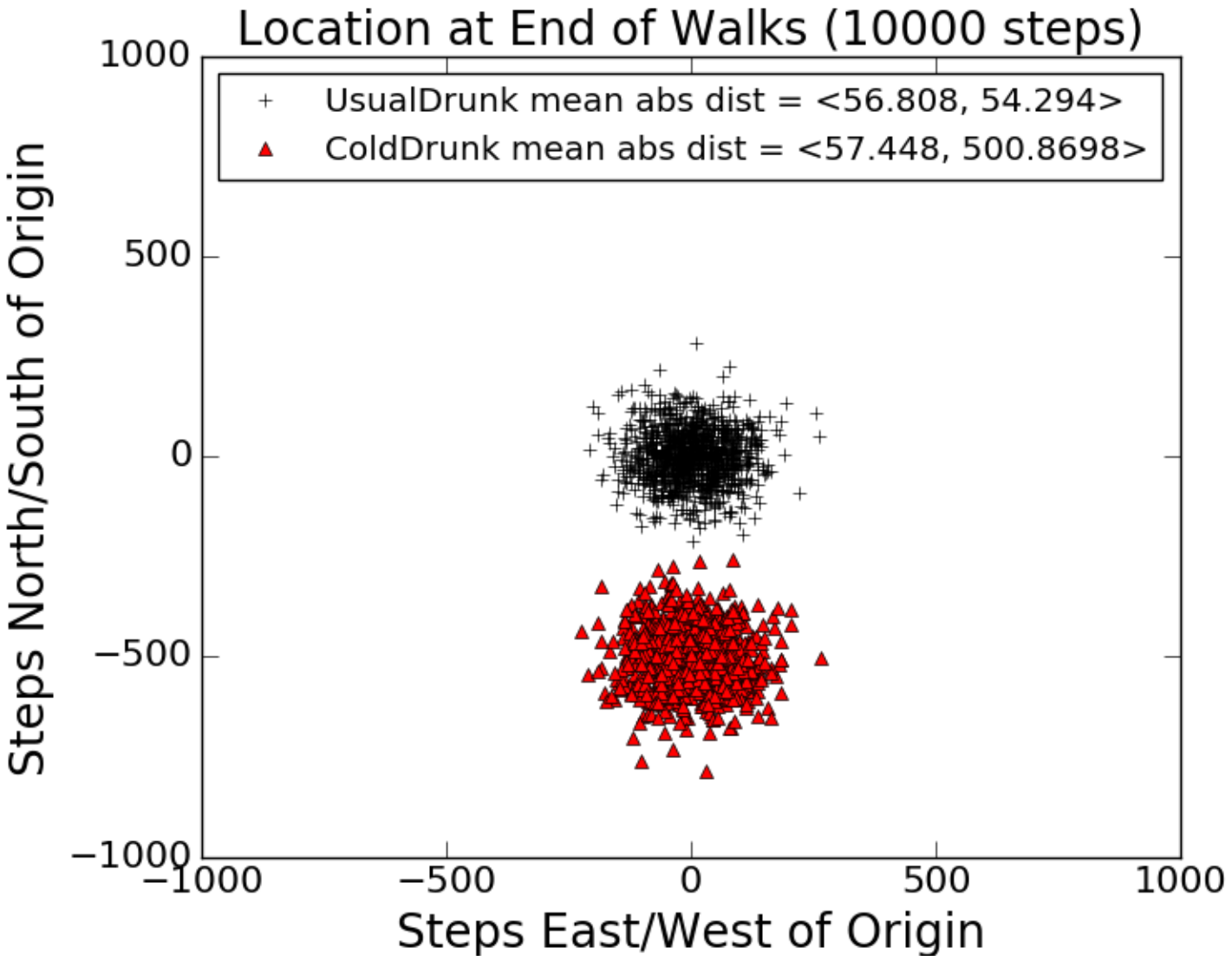
```
def getFinalLocs(numSteps, numTrials, dClass):
    locs = []
    d = dClass()
    for t in range(numTrials):
        f = Field()
        f.addDrunk(d, Location(0, 0))
        for s in range(numSteps):
            f.moveDrunk(d)
        locs.append(f.getLoc(d))
    return locs
```

Plotting Ending Locations

```
def plotLocs(drunkKinds, numSteps, numTrials):
    styleChoice = styleIterator(('k+', 'r^', 'mo'))
    for dClass in drunkKinds:
        locs = getFinalLocs(numSteps, numTrials, dClass)
        xVals, yVals = [], []
        for loc in locs:
            xVals.append(loc.getX())
            yVals.append(loc.getY())
        xVals = pylab.array(xVals)
        yVals = pylab.array(yVals)
        meanX = sum(abs(xVals))/len(xVals)
        meanY = sum(abs(yVals))/len(yVals)
```

- .
- .
- .

Ending Locations



Fields with Wormholes

A Subclass of Field, part 1

```
class OddField(Field):
    def __init__(self, numHoles = 1000,
                 xRange = 100, yRange = 100):
        Field.__init__(self)
        self.wormholes = {}
        for w in range(numHoles):
            x = random.randint(-xRange, xRange)
            y = random.randint(-yRange, yRange)
            newX = random.randint(-xRange, xRange)
            newY = random.randint(-yRange, yRange)
            newLoc = Location(newX, newY)
            self.wormholes[(x, y)] = newLoc
```

A Subclass of Field, part 2

```
def moveDrunk(self, drunk):
    Field.moveDrunk(self, drunk)
    x = self.drunks[drunk].getX()
    y = self.drunks[drunk].getY()
    if (x, y) in self.wormholes:
        self.drunks[drunk] = self.wormholes[(x, y)]
```

Tracing a Walk (part 1)

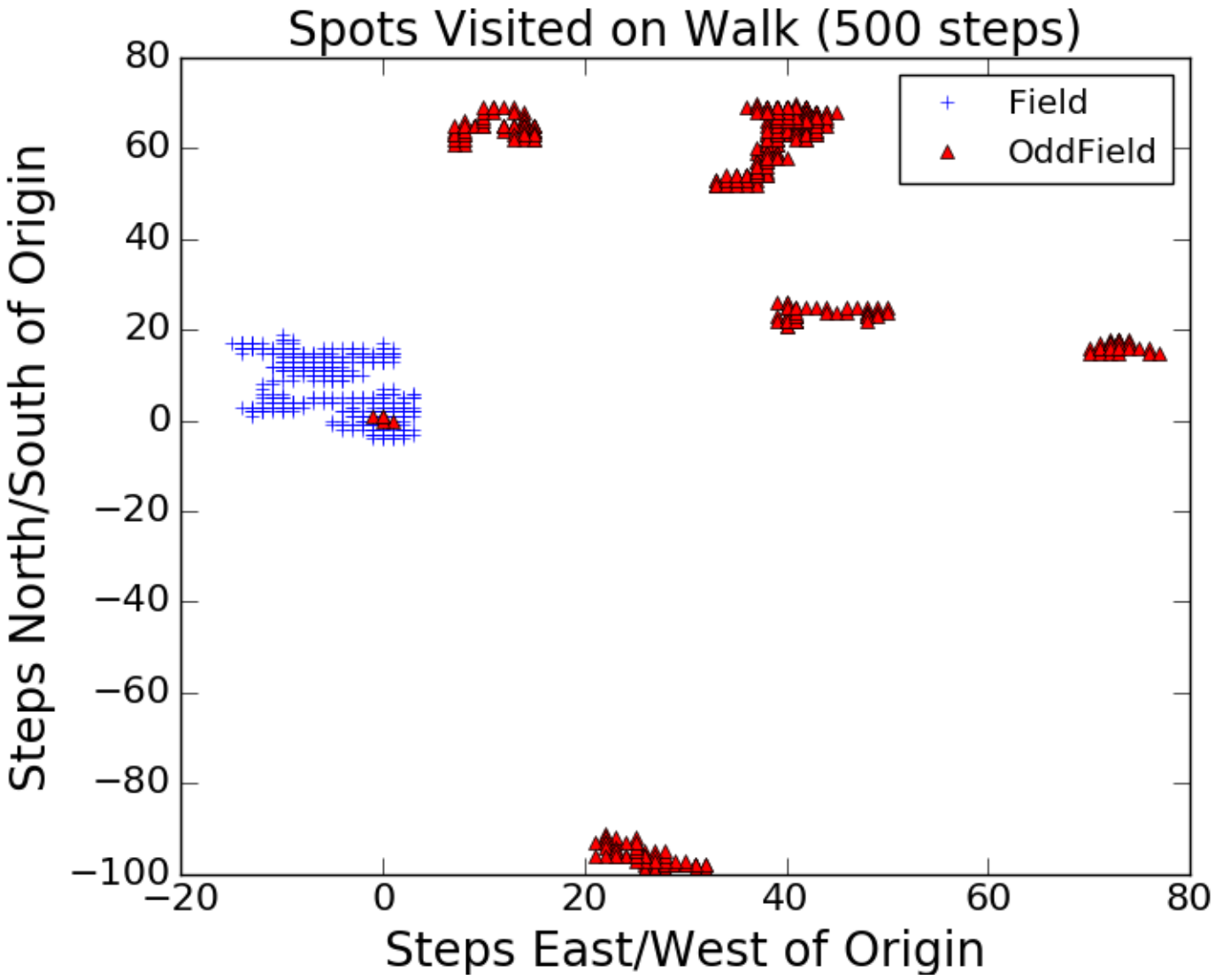
```
def traceWalk(fieldKinds, numSteps):
    styleChoice = styleIterator(('b+', 'r^', 'ko'))
    for fClass in fieldKinds:
        d = UsualDrunk()
        f = fClass()
        f.addDrunk(d, Location(0, 0))
        locs = []
        for s in range(numSteps):
            f.moveDrunk(d)
            locs.append(f.getLoc(d))
        xVals, yVals = [], []
        for loc in locs:
            xVals.append(loc.getX())
            yVals.append(loc.getY())
        curStyle = styleChoice.nextStyle()
        pylab.plot(xVals, yVals, curStyle,
                   label = fClass.__name__)
```

Tracing a Walk (part 2)

```
    pylab.title('Spots Visited on Walk ('
                + str(numSteps) + ' steps)')
    pylab.xlabel('Steps East/West of Origin')
    pylab.ylabel('Steps North/South of Origin')
    pylab.legend(loc = 'best')

traceWalk((Field, OddField), 500)
```

Strange Walks



Summary

- Point is not the simulations themselves, but how we built them
- Three classes corresponding to domain-specific types
  - Location
  - Field
  - Drunk
- Functions corresponding to
  - One trial
  - Multiple trials
  - Result reporting

Summary, cont.

- Created two subclasses of Drunk
- Simulation had an argument of type class, so we could easily investigate both classes of Drunk
- Made series of incremental changes to simulation so that we could investigate different questions
  - Get simple version working first
  - Elaborate a step at a time
- Introduced a weird subclass of Field
  - Easy to add to simulation
  - Would have been hard to model analytically

Coming Up Next

- We spent this lecture looking at a simulation and using it to draw some conclusions
- Time to get serious about stochastic simulations
  - Probabilistic thinking
  - Understanding how much confidence we should have in the result of a simulation