Monte Carlo Simulation and the CLT

Ended Last Lecture with a Cliffhanger

Empirical works for normal distributions

But the outcomes of spins of a roulette wheel are not normally distributed

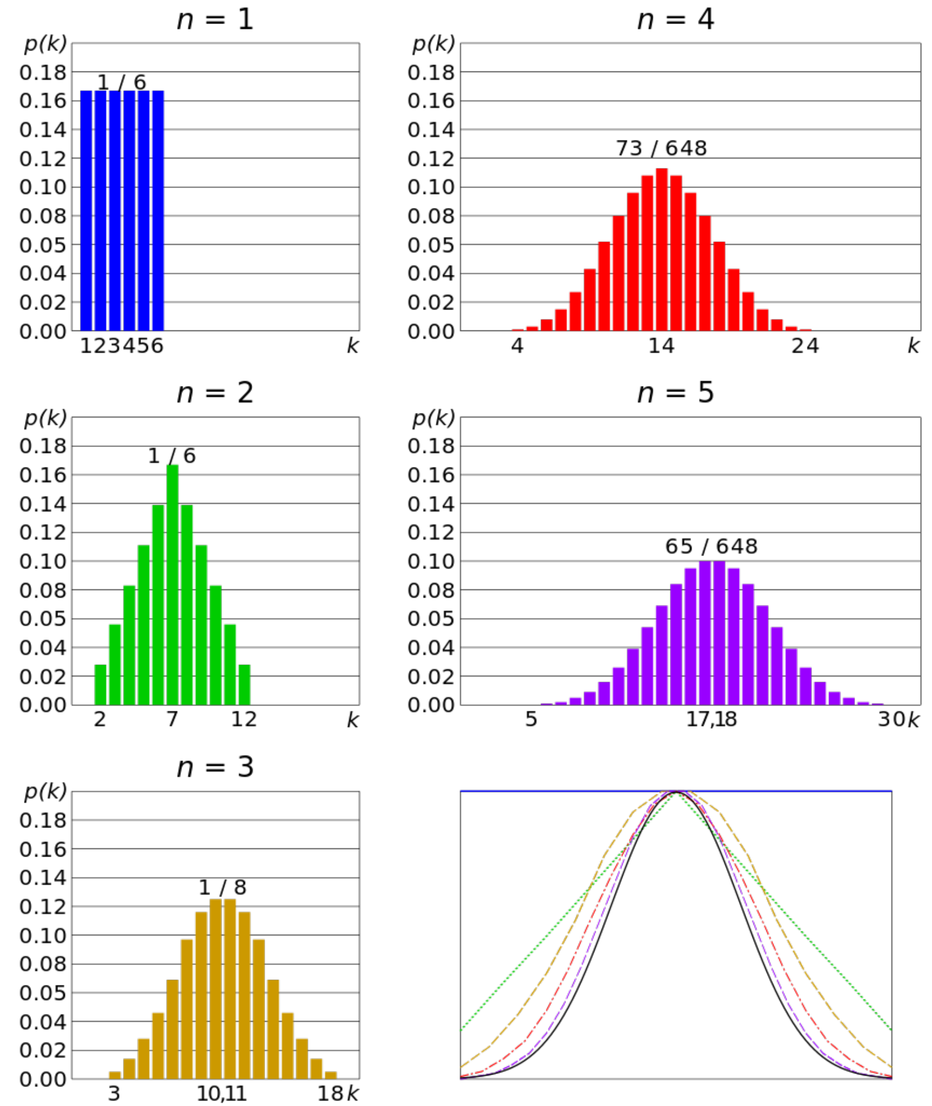
They are uniformly distributed since ach outcome is equally probable

So, why does empirical work?

Photo by Juraj Patekar

Why Did the Empirical Rule Work?

- Because we are reasoning not about a single spin, but about the mean of a set of spins
- And the central limit theorem applies



Two most important theorem in all probabilities:
> the law of large numbers
> central limit theorem

The Central Limit Theorem (CLT)

- Given a sufficiently large sample:

  1) The means of the samples in a set of samples (the sample means) will be approximately normally distributed,

  2) This normal distribution will have a mean close to the mean of the population, and

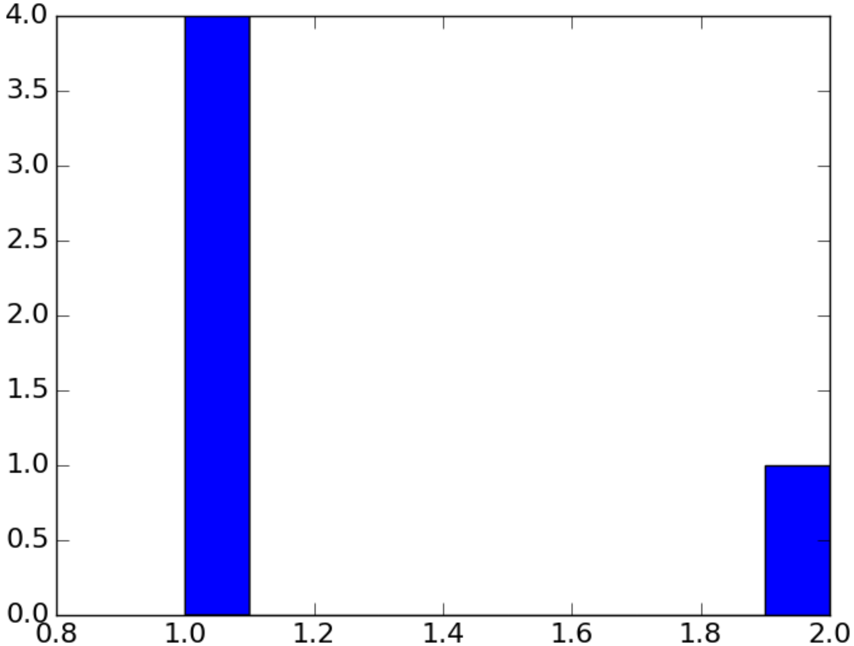
  3) The variance of the sample means will be close to the variance of the population divided by the sample size.

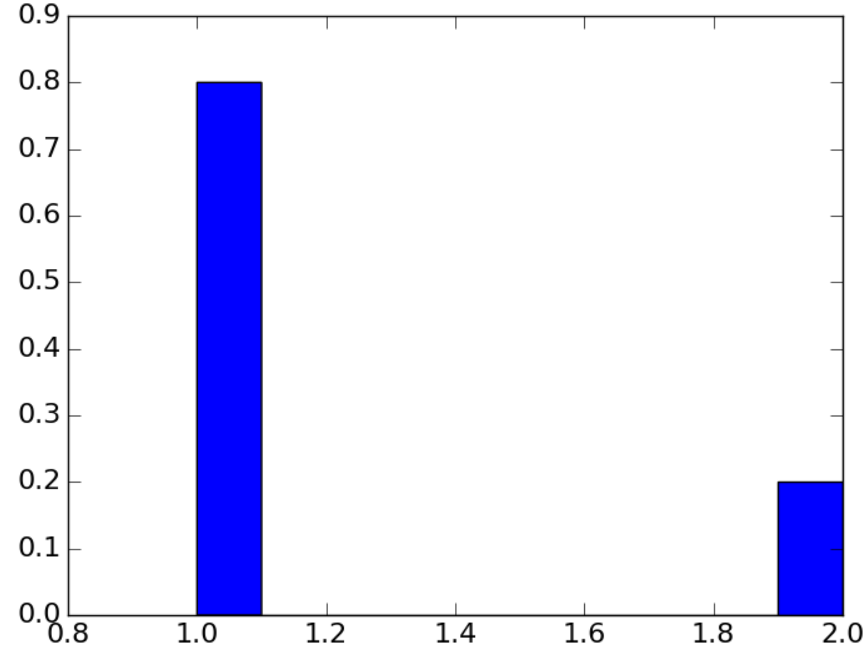
Checking CLT

```
def plotMeans(numDice, numRolls, numBins, legend, color, style):
    means = []
    for i in range(numRolls//numDice):
        vals = 0
        for j in range(numDice):
            vals += 5*random.random()
        means.append(vals/float(numDice))
    pylab.hist(means, numBins, color = color, label = legend,
               weights = pylab.array(len(means)*[1])/len(means),
               hatch = style)
    return getMeanAndStd(means)

mean, std = plotMeans(1, 1000000, 19, '1 die', 'b', '*')
print('Mean of rolling 1 die =', str(mean) + ',', 'Std =', std)
mean, std = plotMeans(50, 1000000, 19, 'Mean of 50 dice', 'r', '//')
print('Mean of rolling 50 dice =', str(mean) + ',', 'Std =', std)
pylab.title('Rolling Continuous Dice')
pylab.xlabel('Value')
pylab.ylabel('Probability')
pylab.legend()
```

Weighting the Bins





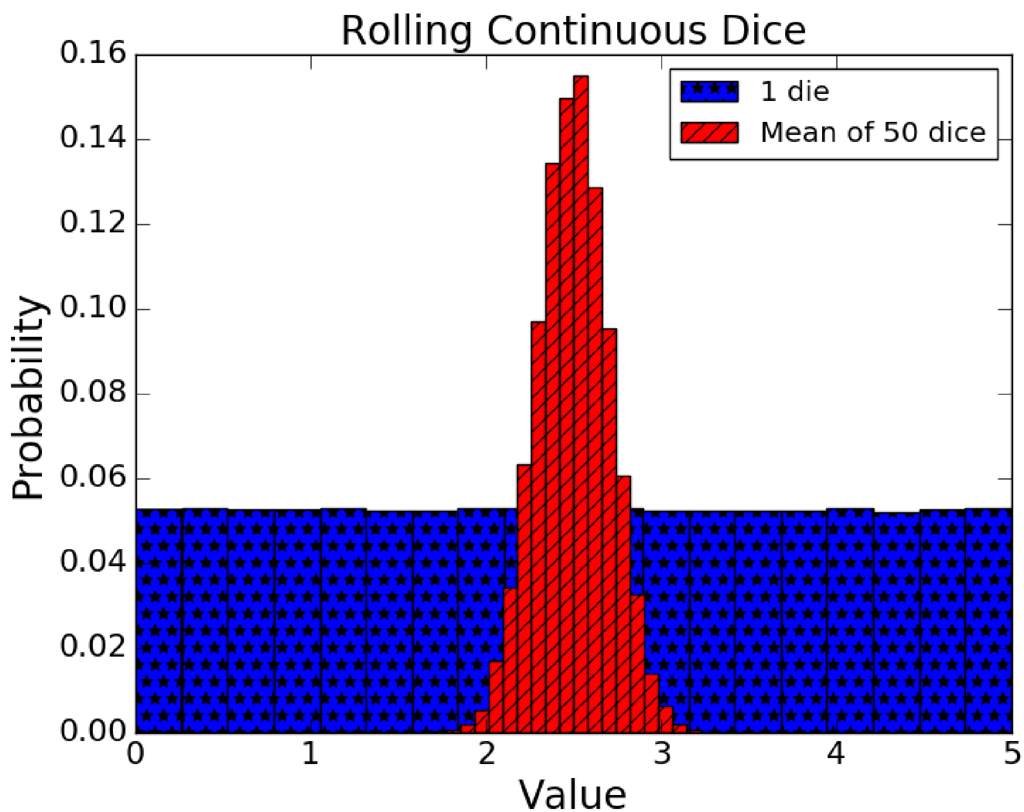
Checking CLT

```
def plotMeans(numDice, numRolls, numBins, legend, color, style):
    means = []
    for i in range(numRolls//numDice):
        vals = 0
        for j in range(numDice):
            vals += 5*random.random()
        means.append(vals/float(numDice))
    pylab.hist(means, numBins, color = color, label = legend,
               weights = pylab.array(len(means)*[1])/len(means),
               hatch = style)
    return getMeanAndStd(means)

mean, std = plotMeans(1, 1000000, 19, '1 die', 'b', '*')
print('Mean of rolling 1 die =', str(mean) + ',', 'Std =', std)
mean, std = plotMeans(50, 1000000, 19, 'Mean of 50 dice', 'r', '//')
print('Mean of rolling 50 dice =', str(mean) + ',', 'Std =', std)
pylab.title('Rolling Continuous Dice')
pylab.xlabel('Value')
pylab.ylabel('Probability')
pylab.legend()
```

Output

Mean of rolling 1 die = 2.49759575528, Std = 1.4439045633
Mean of rolling 50 dice = 2.49985051798, Std = 0.204887274645



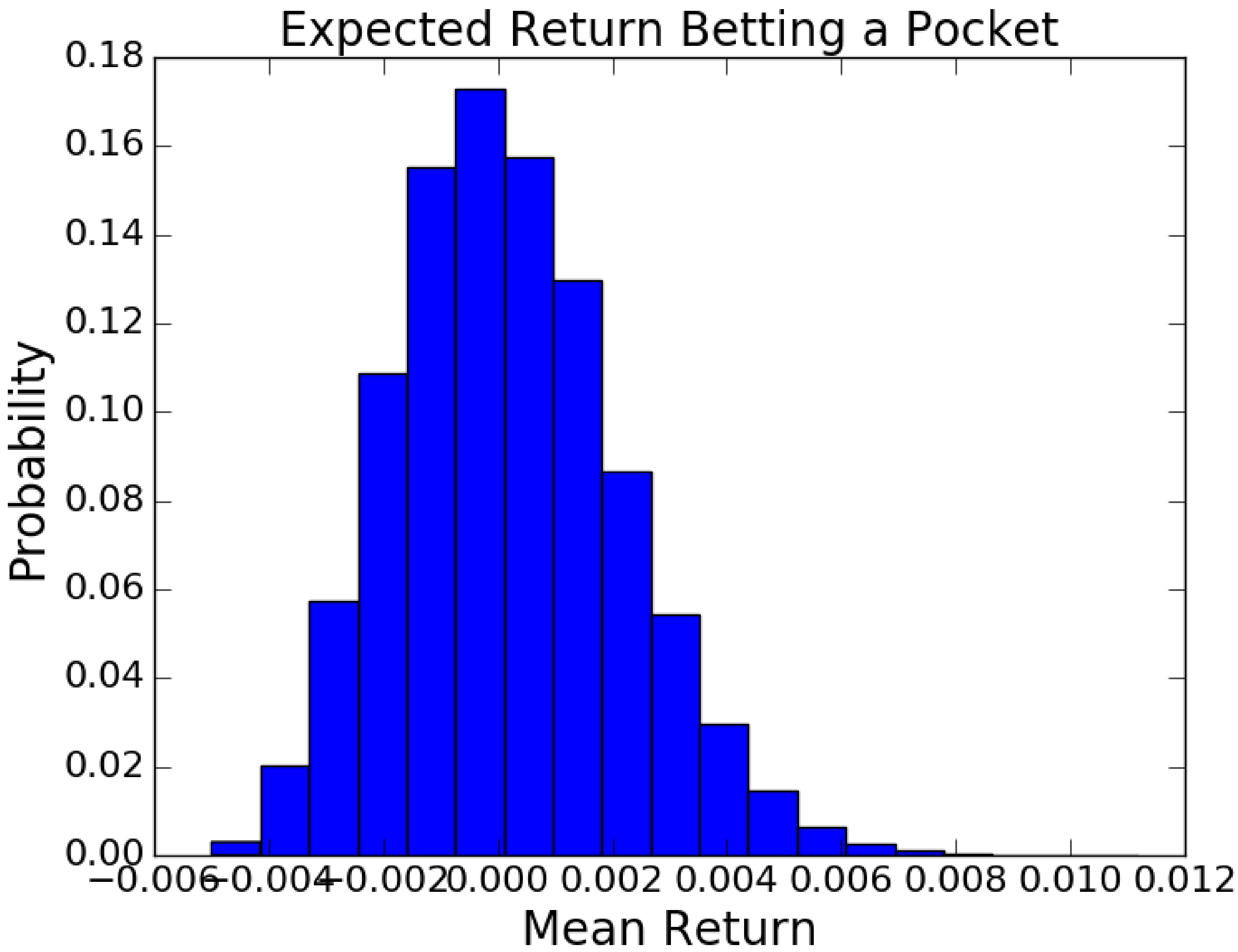
Try It for Roulette

```
numTrials = 50000
numSpins = 200
game = FairRoulette()

means = []
for i in range(numTrials):
    means.append(findPocketReturn(game, 1,
   numSpins)[0]/numSpins)

pylab.hist(means, bins = 19,
           weights = pylab.array(len(means)*[1])/len(means))
pylab.xlabel('Mean Return')
pylab.ylabel('Probability')
pylab.title('Expected Return Betting a Pocket')
```

Means Close to Normally Distributed!



Moral

- It doesn't matter what the shape of the distribution of values happens to be
- If we are trying to estimate the mean of a population using sufficiently large samples
- The CLT allows us to use the empirical rule when computing confidence intervals