Sampling and Monte Carlo Simulations

Lecturer: John Guttag

Nondeterminism

Causal nondeterminism:

Not every event is caused by previous events

Predictive nondeterminism:

Lack of knowledge about the world makes it impossible make accurate predications about future states

Stochastic Processes

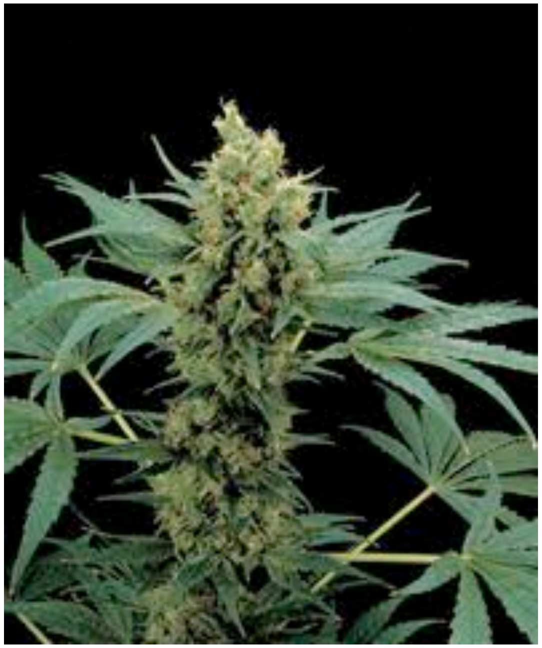
An ongoing process where the next state might depend on both the previous states and some random element.

Rolling a Die

```
def rollDie():
    """returns an int between 1 and 6"""

def rollDie():
    """returns a random int between 1 and 6"""
```

Hash Tables

Lecturer: John Guttag



```
def strToInt(s):
    number = ''
    for c in s:
        number = number + str(ord(c))
    index = int(number)
    return index
```

```
def hashStr(s, tableSize = 101):
    number = ''
    for c in s:
        number = number + str(ord(c))
    index = int(number)%tableSize
    return index
```

14
EcoBoost
6
38

The Law of Large Numbers

Lecturer: John Guttag

Law of Large Numbers

In repeated **independent tests** with the same **actual probability** *p* of a particular outcome in each test,

the chance that the **fraction of times** that outcome occurs differs from *p* converges to zero as the number of trials goes to infinity.

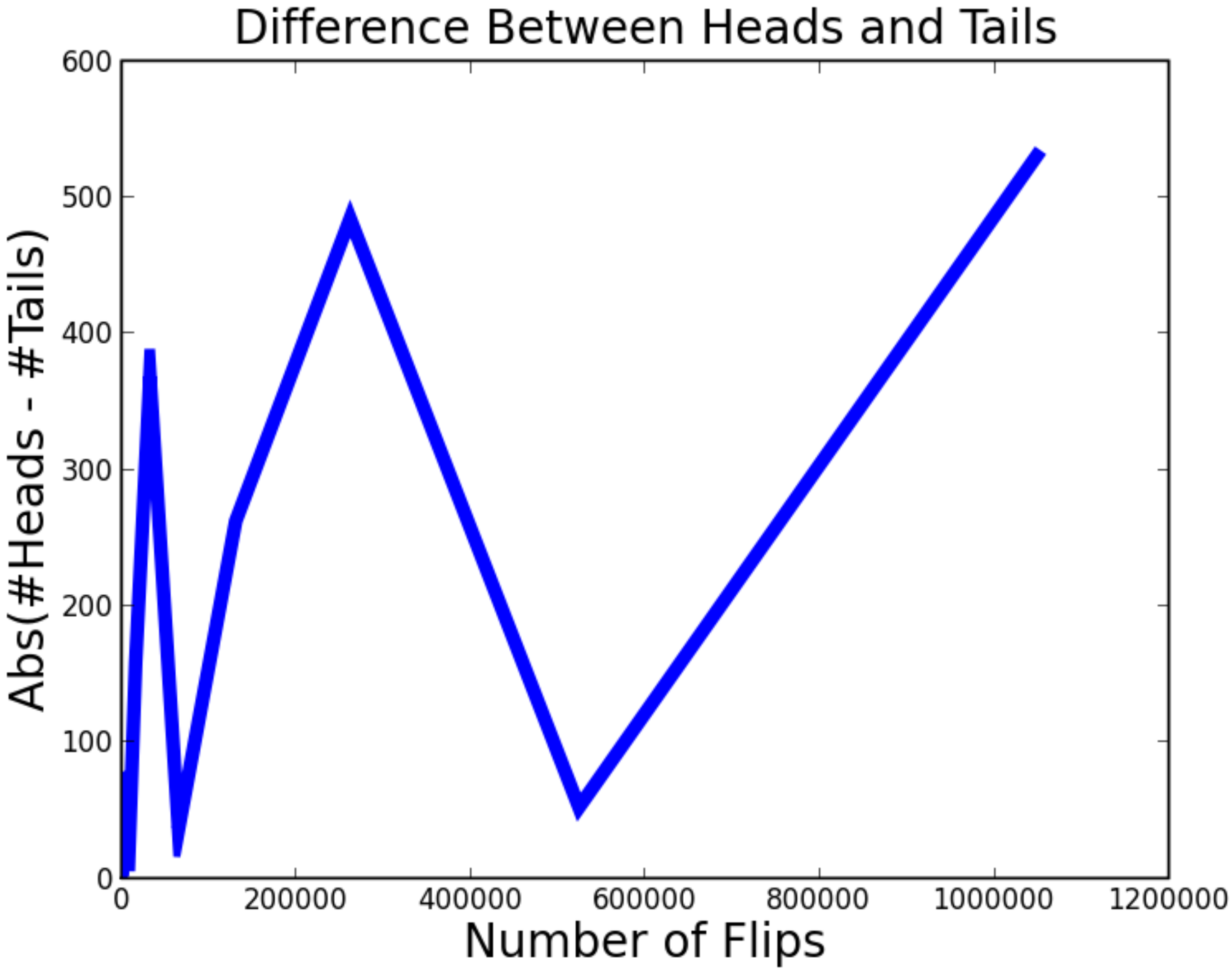
Gambler's Fallacy

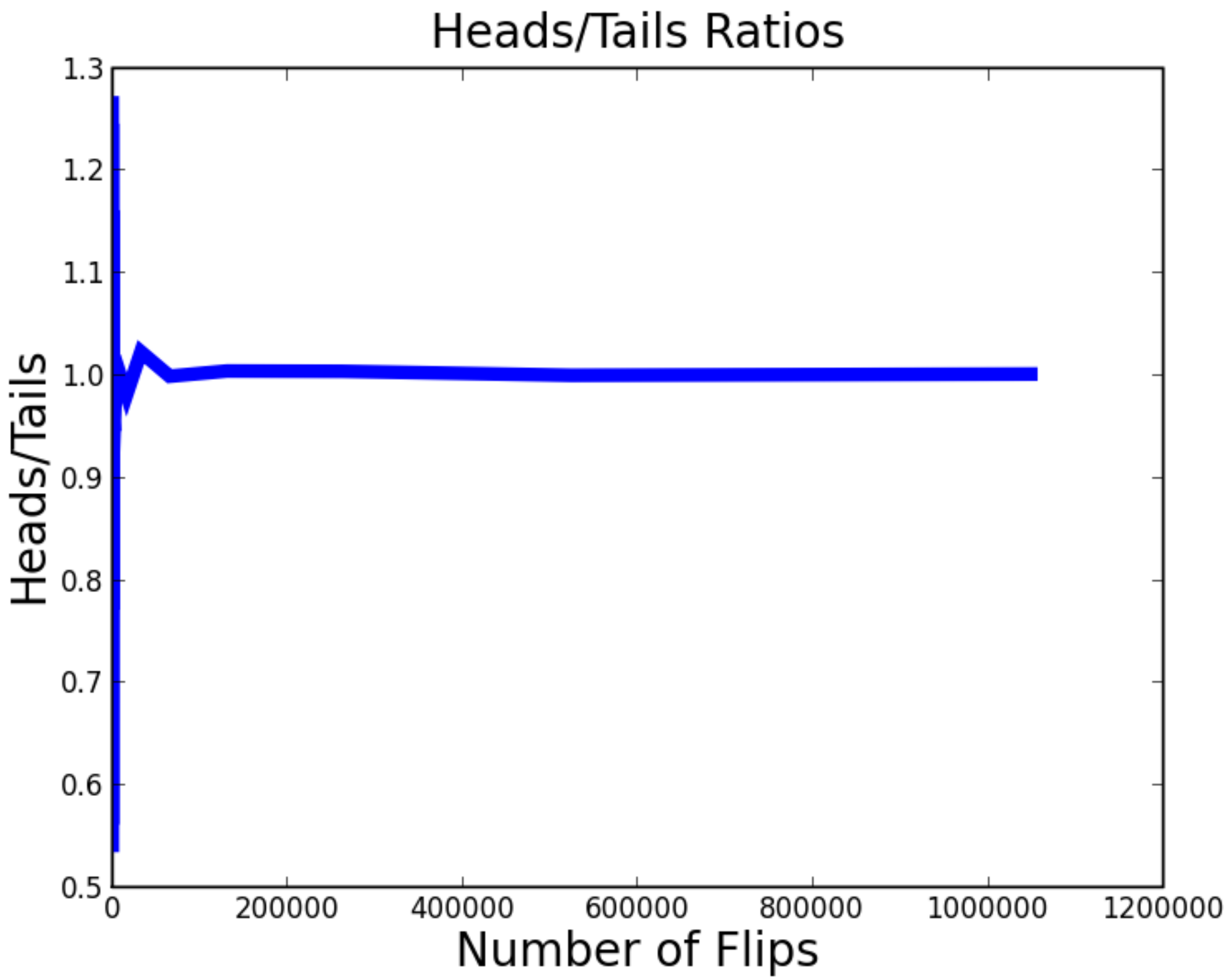
If deviations from expected behavior occur, these deviations are likely to be evened out by opposite deviations in the future.

```
def flipPlot(minExp, maxExp):
    """Assumes minExp and maxExp positive
               integers; minExp < maxExp
       Plots results of 2**minExp to
              2**maxExp coin flips"""
    ratios = []
    diffs = []
    xAxis = []
    for exp in range(minExp, maxExp + 1):
        xAxis.append(2**exp)
. . .
```

```
 for numFlips in xAxis:
        numHeads = 0
        for n in range(numFlips):
            if random.random() < 0.5:
                numHeads += 1
        numTails = numFlips - numHeads
        ratios.append(numHeads/float(numTails))
        diffs.append(abs(numHeads - numTails))

. . .
```

```
pylab.title('Difference Between Heads and Tails')
   pylab.xlabel('Number of Flips')
   pylab.ylabel('Abs(#Heads - #Tails)')
   pylab.plot(xAxis, diffs)
   pylab.figure()
   pylab.title('Heads/Tails Ratios')
   pylab.xlabel('Number of Flips')
   pylab.ylabel('Heads/Tails')
   pylab.plot(xAxis, ratios)
```


Difference Between Heads and Tails
Abs(#Heads - #Tails)
600
500
400
300
200
100
0
0
200000
400000
600000
800000
1000000
1200000
Number of Flips


Heads/Tails Ratios
Heads/Tails
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0
200000
400000
600000
800000
1000000
1200000
Number of Flips

How Much Is Enough?

Lecturer: John Guttag

How Much is Enough?

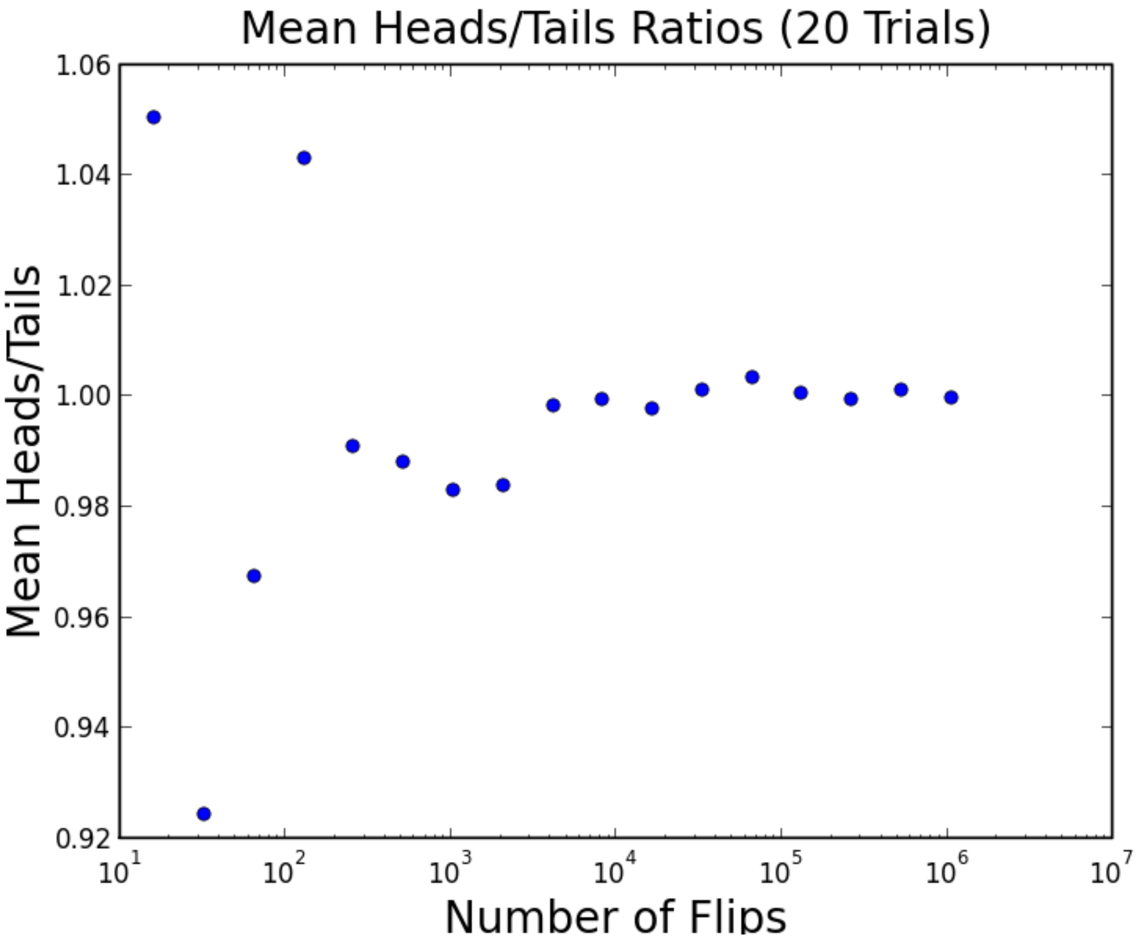
How many samples do we need to look at in order to have a justified confidence that something that is true about the population of samples is also true about the population from which the samples were drawn?

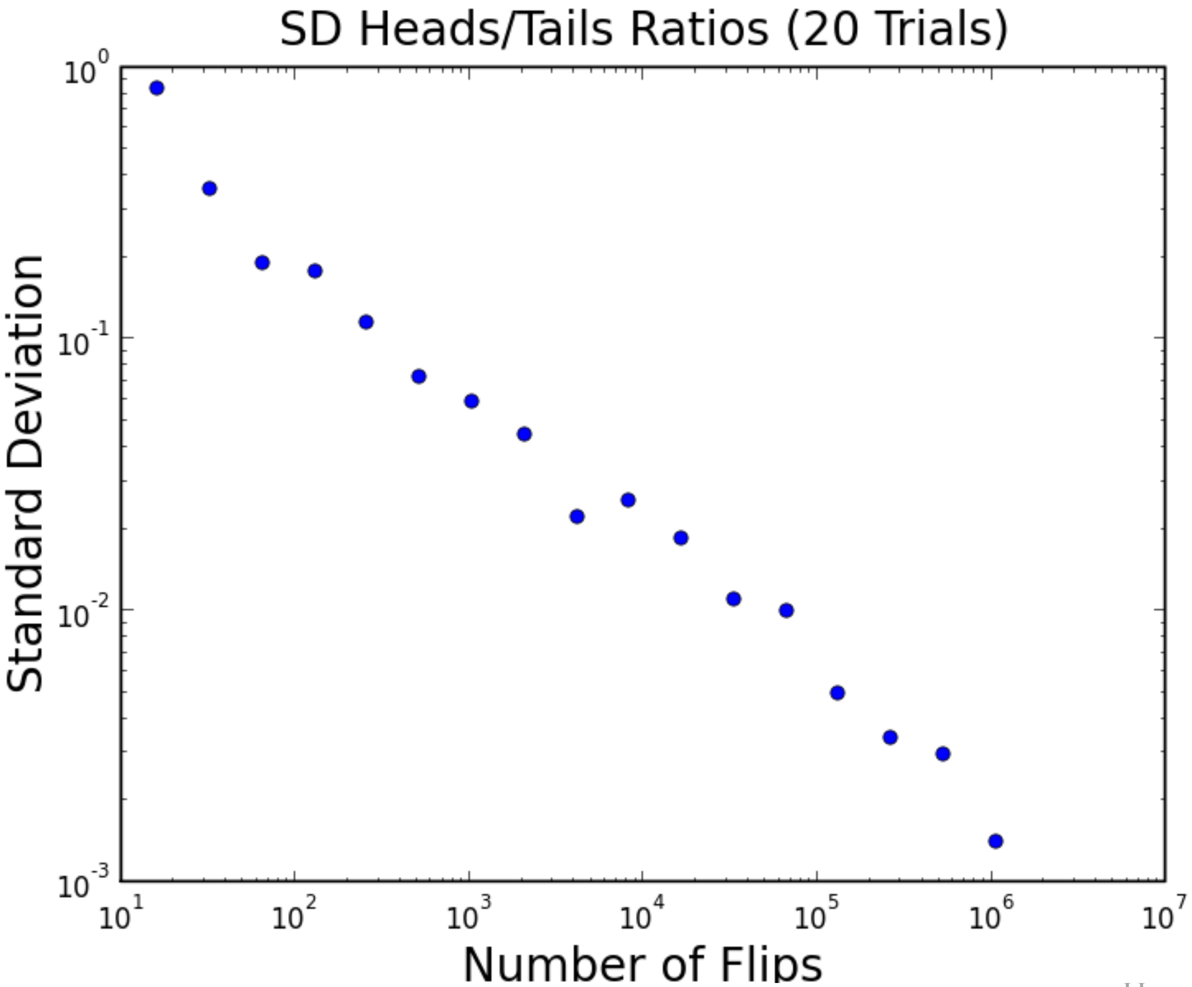
Depends upon the variance in the underlying distribution

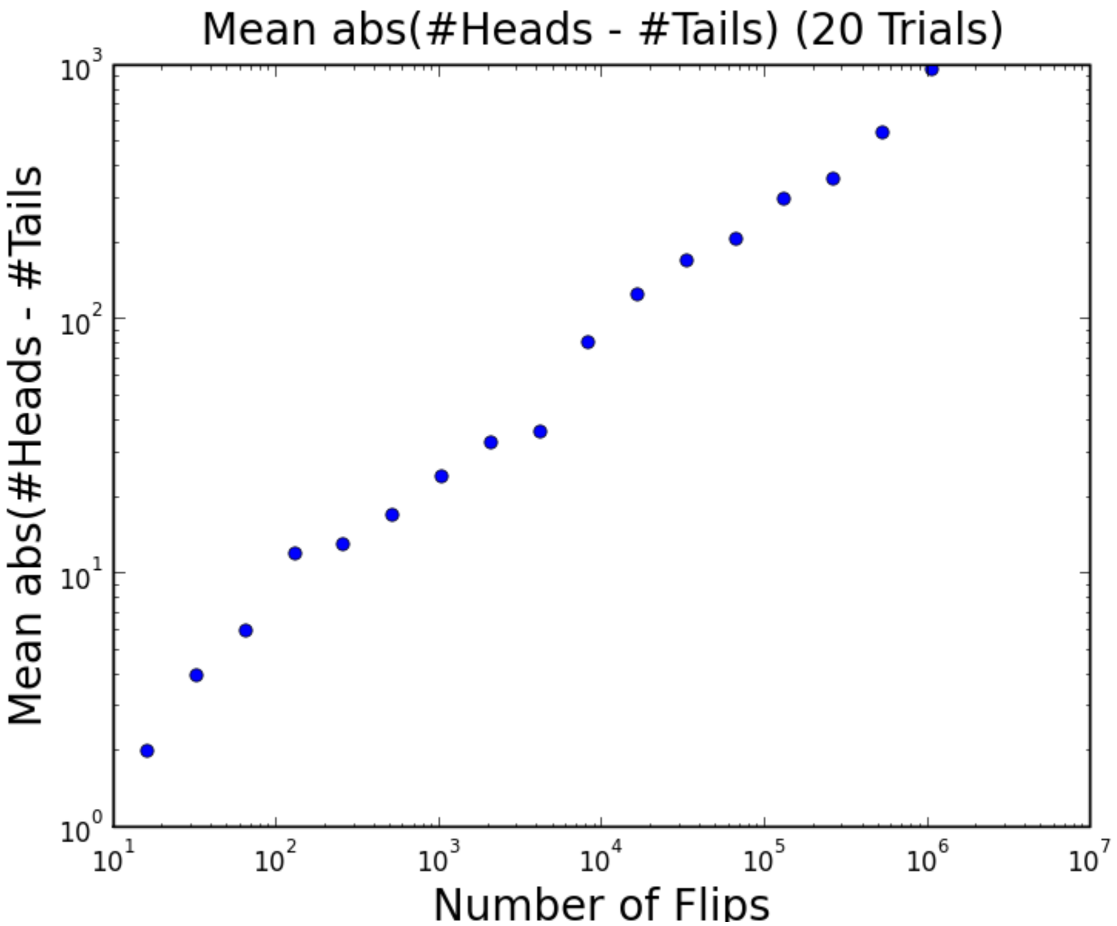
Variance

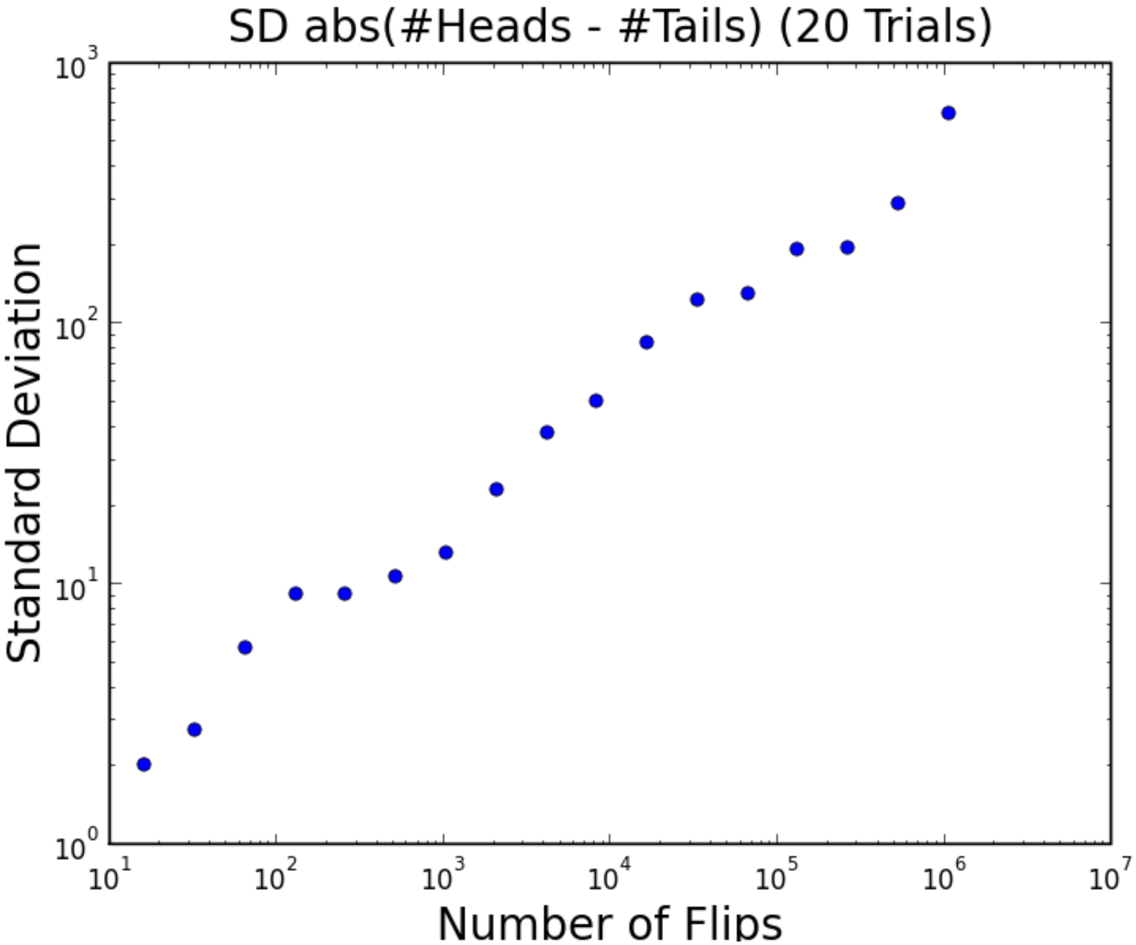
We measure the amount of variance in the outcomes of multiple trials.

```
def stdDev(X):
    mean = sum(X)/float(len(X))
    tot = 0.0
    for x in X:
        tot += (x - mean)**2
    return (tot/len(X))**0.5
```


Mean Heads/Tails Ratios (20 Trials)
Mean Heads/Tails
1.06
1.04
1.02
1.00
0.98
0.96
0.94
0.92
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Number of Flips

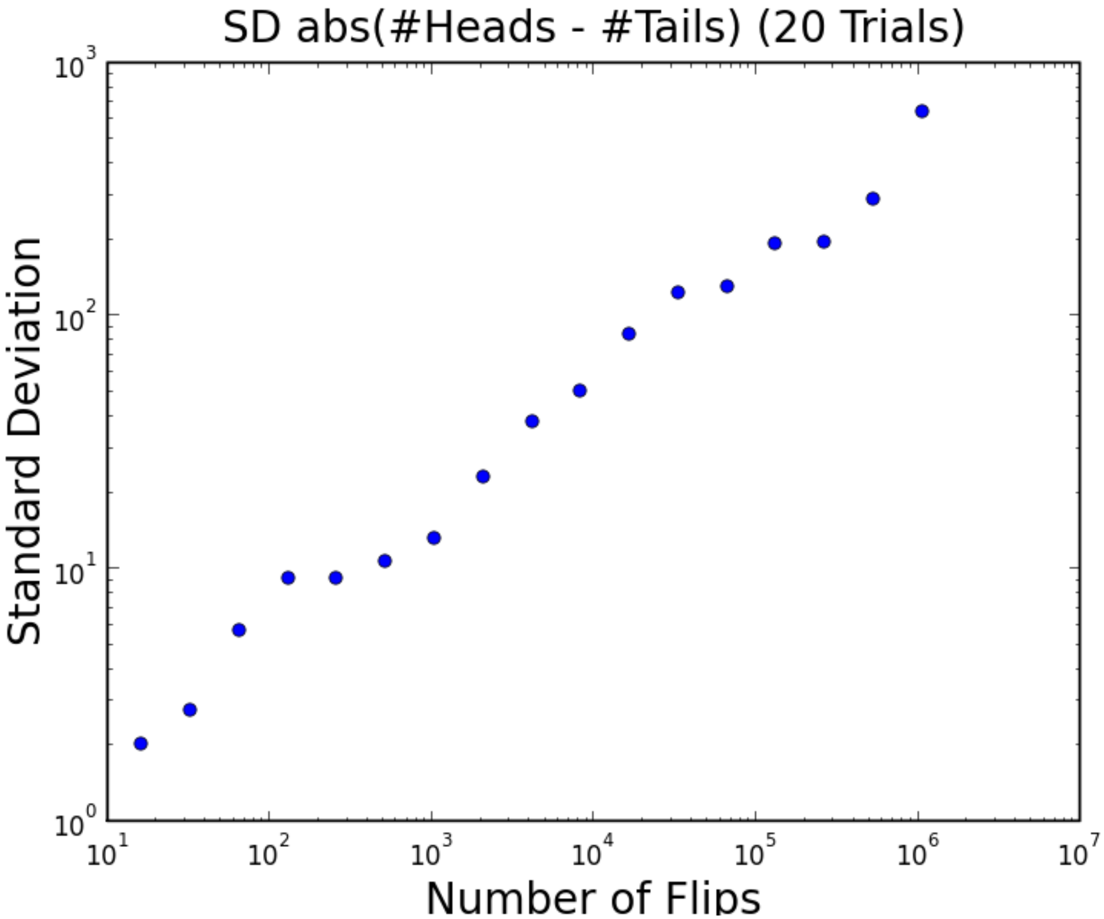

SD Heads/Tails Ratios (20 Trials)
Standard Deviation
Number of Flips
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}


Mean abs(#Heads - #Tails) (20 Trials)
Mean abs(#Heads - #Tails
Number of Flips
10^0
10^1
10^2
10^3
10^1
10^2
10^3
10^4
10^5
10^6
10^7


SD abs(#Heads - #Tails) (20 Trials)
Standard Deviation
Number of Flips
10^0
10^1
10^2
10^3
10^1
10^2
10^3
10^4
10^5
10^6
10^7

Standard Deviations and Histograms

Lecturer: John Guttag

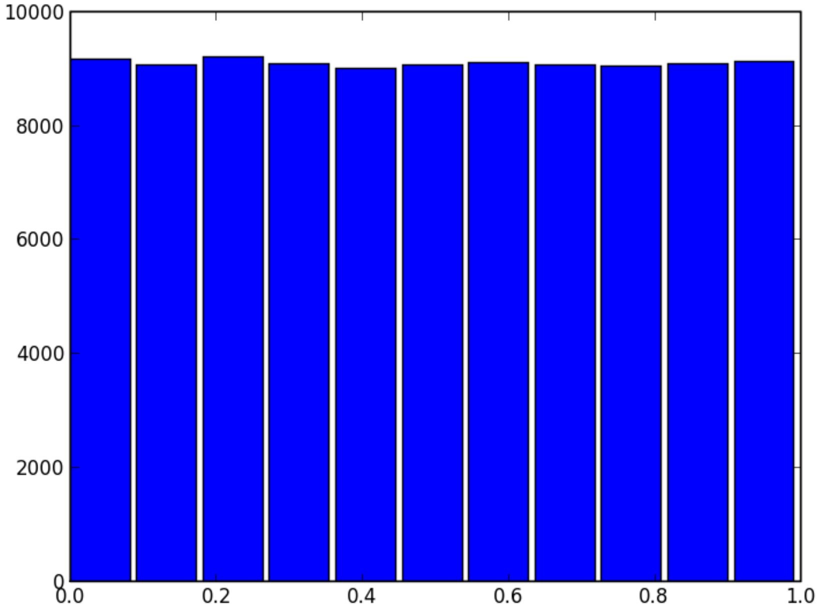

SD abs(#Heads - #Tails) (20 Trials)
Standard Deviation
Number of Flips
10^0
10^1
10^2
10^3
10^1
10^2
10^3
10^4
10^5
10^6
10^7

```
def CV(X):
    mean = sum(X)/float(len(X))
    try:
        return stdDev(X)/mean
    except ZeroDivisionError:
        return float('NaN')
```

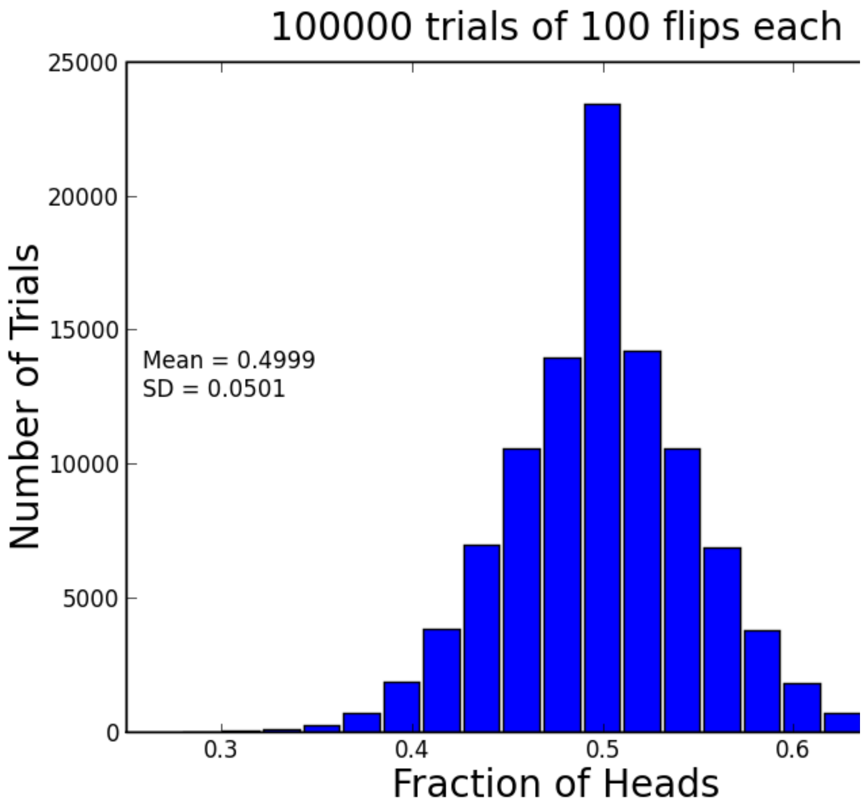
```
def flip(numFlips):

def flipSim(numFlipsPerTrial, numTrials):

def labelPlot(nf, nt, mean, sd):

def makePlots(nf1, nf2, nt):
  """nt = number of trials per experiment
     nf1 = number of flips 1st experiment
     nf2 = number of flips 2nd experiment"""
```

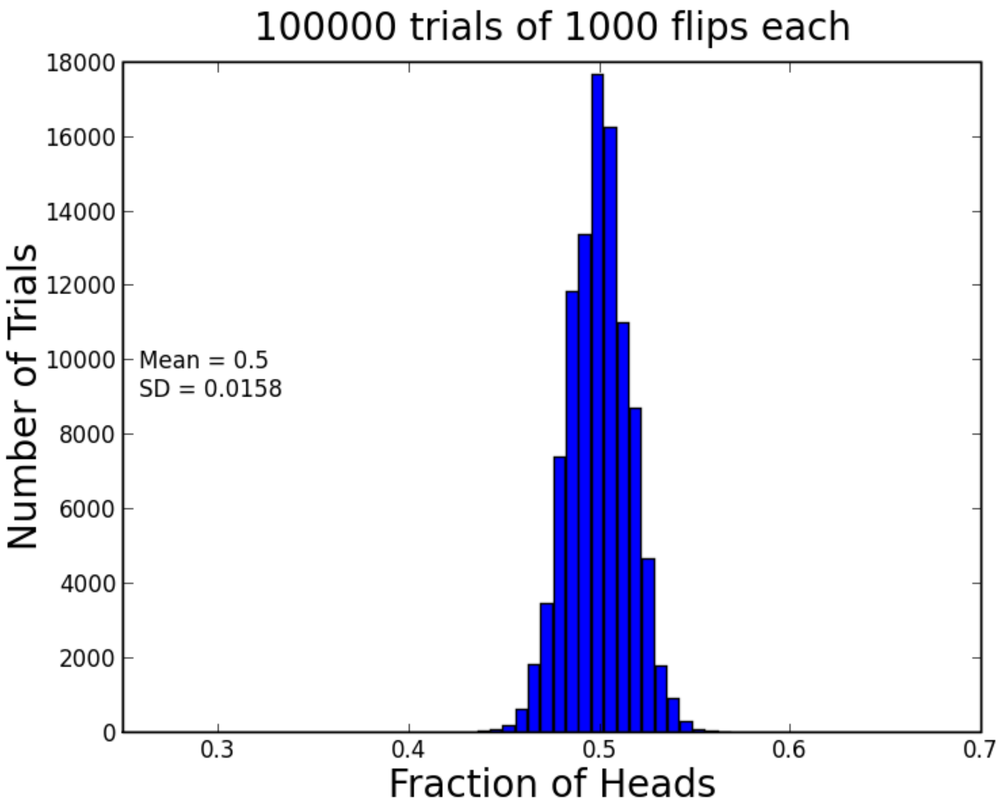
```
def makePlots(numFlips1, numFlips2, numTrials):
    val1, mean1, sd1 = flipSim(numFlips1, numTrials)
    pylab.hist(val1, bins = 20)
    xmin,xmax = pylab.xlim()
    ymin,ymax = pylab.ylim()
    labelPlot(numFlips1, numTrials, mean1, sd1)
    pylab.figure()
    val2, mean2, sd2 = flipSim(numFlips2, numTrials)
    pylab.hist(val2, bins = 20)
    pylab.xlim(xmin, xmax)
    ymin, ymax = pylab.ylim()
    labelPlot(numFlips2, numTrials, mean2, sd2)
```

```
vals = []
for i in range(100000):
    num = random.random()
    vals.append(num)
pylab.hist(vals, bins = 11)
```



```
vals = []
for i in range(100000):
    num = random.random()
    vals.append(num)
pylab.hist(vals, bins = 11)
xmin, xmax = pylab.xlim()
ymin, ymax = pylab.ylim()
print 'x-range =', xmin, '-', xmax
print 'y-range =', ymin, '-', ymax
pylab.figure
pylab.hist(vals, bins = 11)
#pylab.xlim(-1.0, 2.0)
```


100000 trials of 100 flips each
Number of Trials
25000
20000
15000
10000
5000
0
Mean = 0.4999
SD = 0.0501
0.3
0.4
0.5
0.6
Fraction of Heads


100000 trials of 1000 flips each
Number of Trials
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Mean = 0.5
SD = 0.0158
0.3
0.4
0.5
0.6
0.7
Fraction of Heads