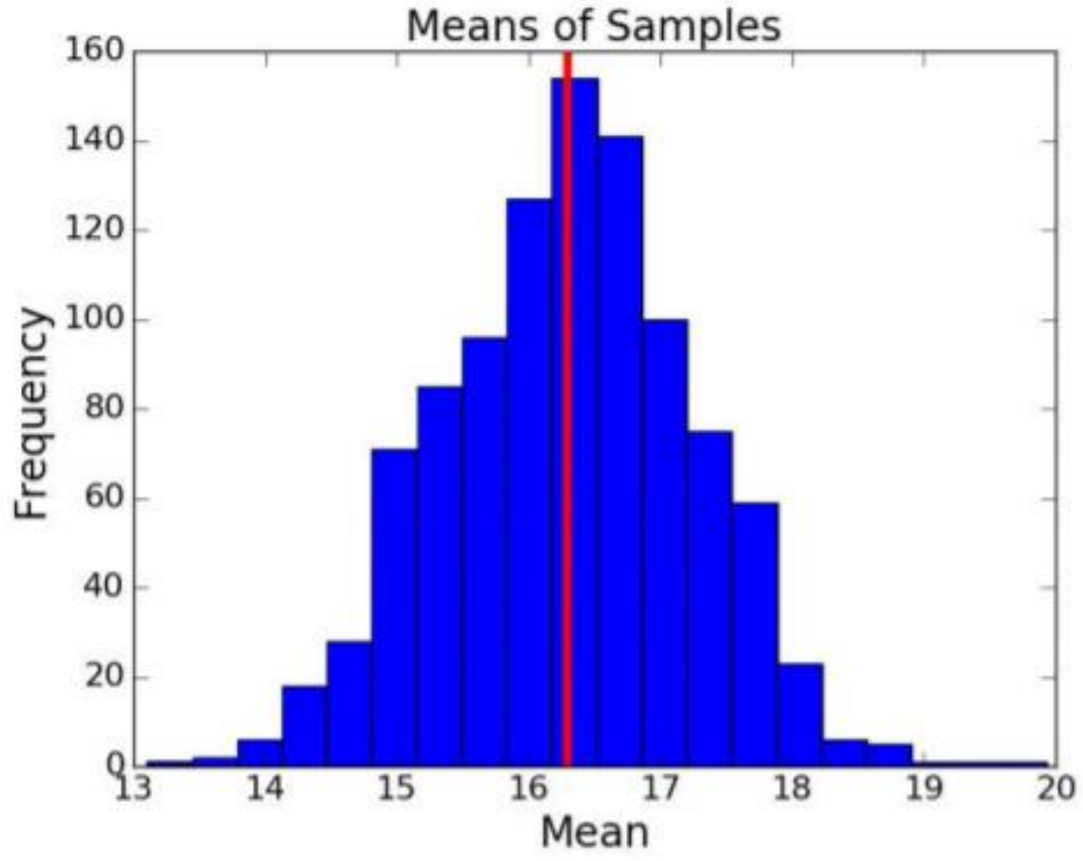
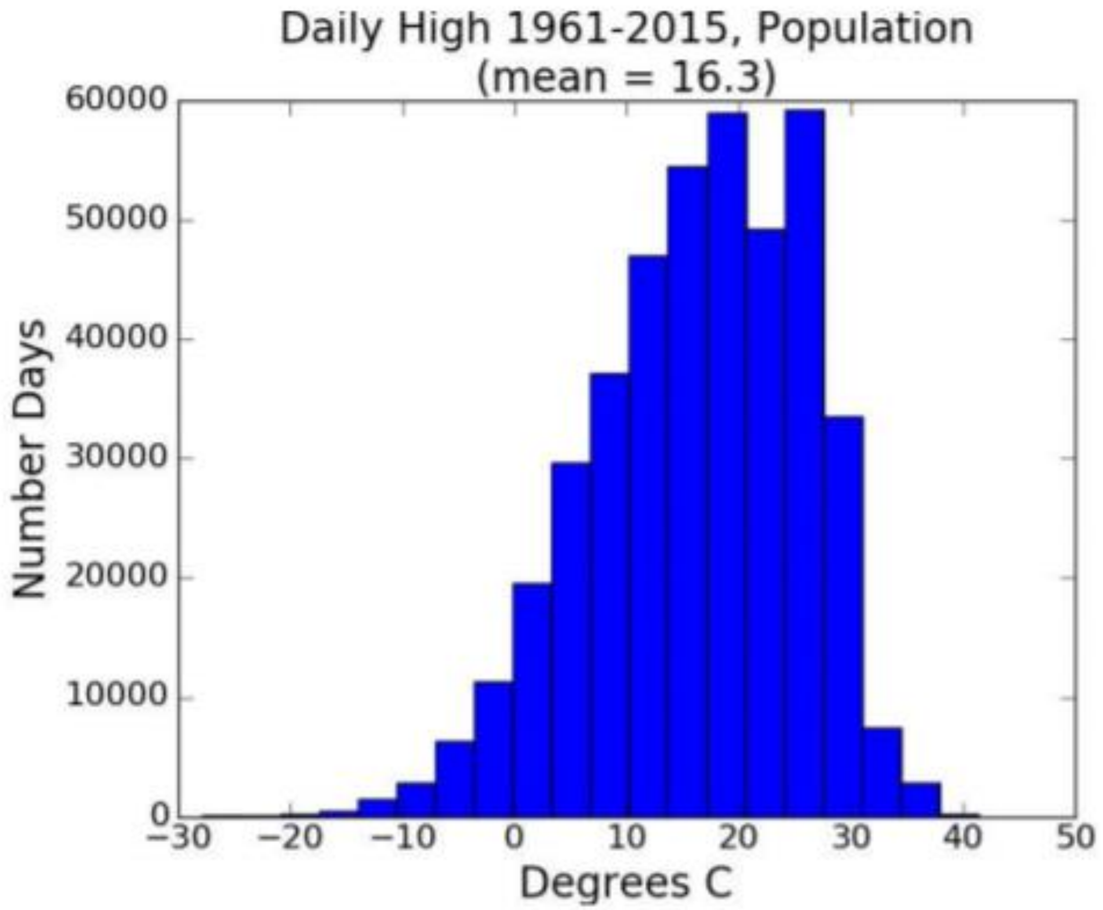
Lesson 9b

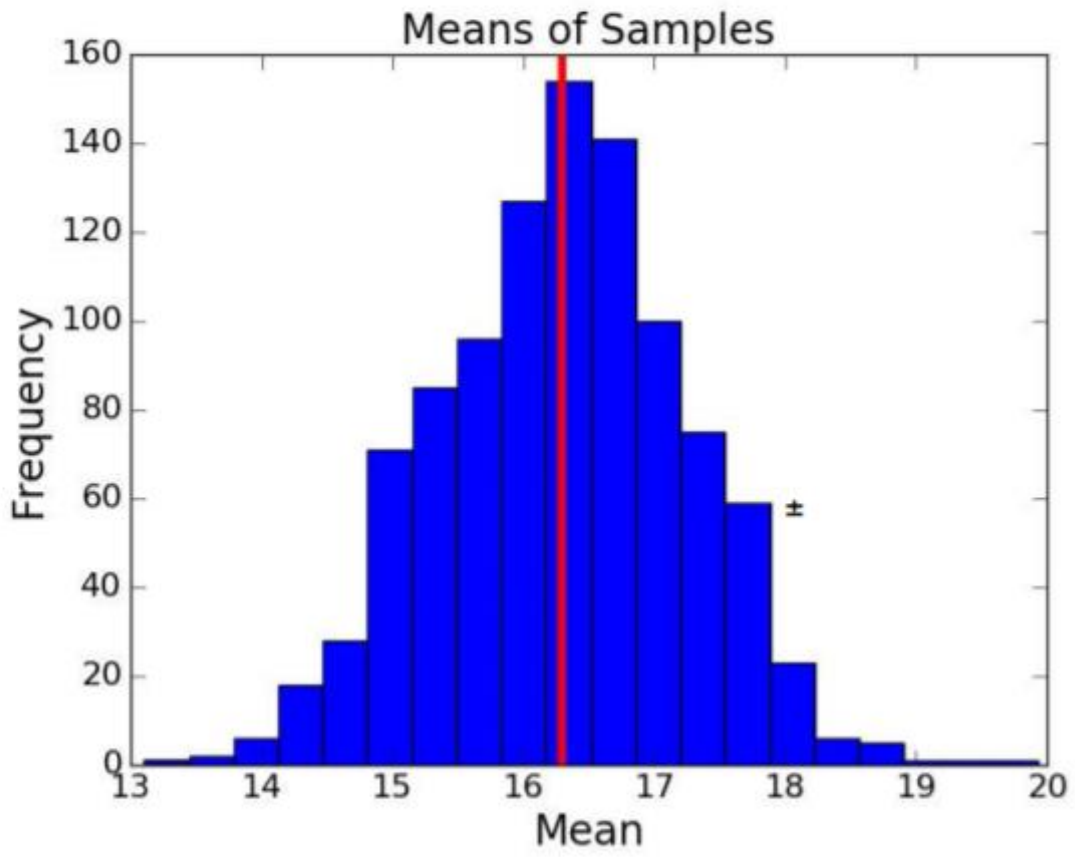
continuation of CLT

Try It 1000 Times





Try It 1000 Times



Mean of sample Means = 16.3
Standard deviation of sample means = 0.94

What's the 95% confidence interval?

16.28 +- 1.96*0.94
14.5 - 18.1

Includes population mean, but pretty wide

Suppose we want a tighter bound?

Getting a Tighter Bound

- Will drawing more samples help?
  - Let’s try increasing from 1000 to 2000
    - Standard deviation goes from 0.943 to 0.946
- How about larger samples?
  - Let’s try increasing sample size from 100 to 200
  - Standard deviation goes from 0.943 to 0.662

Larger Samples Seem to Be Better

- Going from a sample size of 50 to 600 reduced the confidence interval from about 1.2C to about 0.34C.
- But we are now looking at 600*100 = 600k examples
  - What has sampling bought us?
  - Absolutely Nothing!
    - Entire population contained ~422k samples

What Can We Conclude from 1 Sample?

- More than you might think
- Thanks to the Central Limit Theorem

Recall Central Limit Theorem

- 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 the population, and
  - 3) The variance of the sample means will be close to the variance of the population divided by the sample size.
- Time to use the 3rd feature
- Compute *standard error of the mean* (SEM or SE)

Standard Error of the Mean

$$SE = \frac{\sigma}{\sqrt{n}}$$

```
def sem(popSD, sampleSize):
    return popSD/sampleSize**0.5
```

Testing the SEM

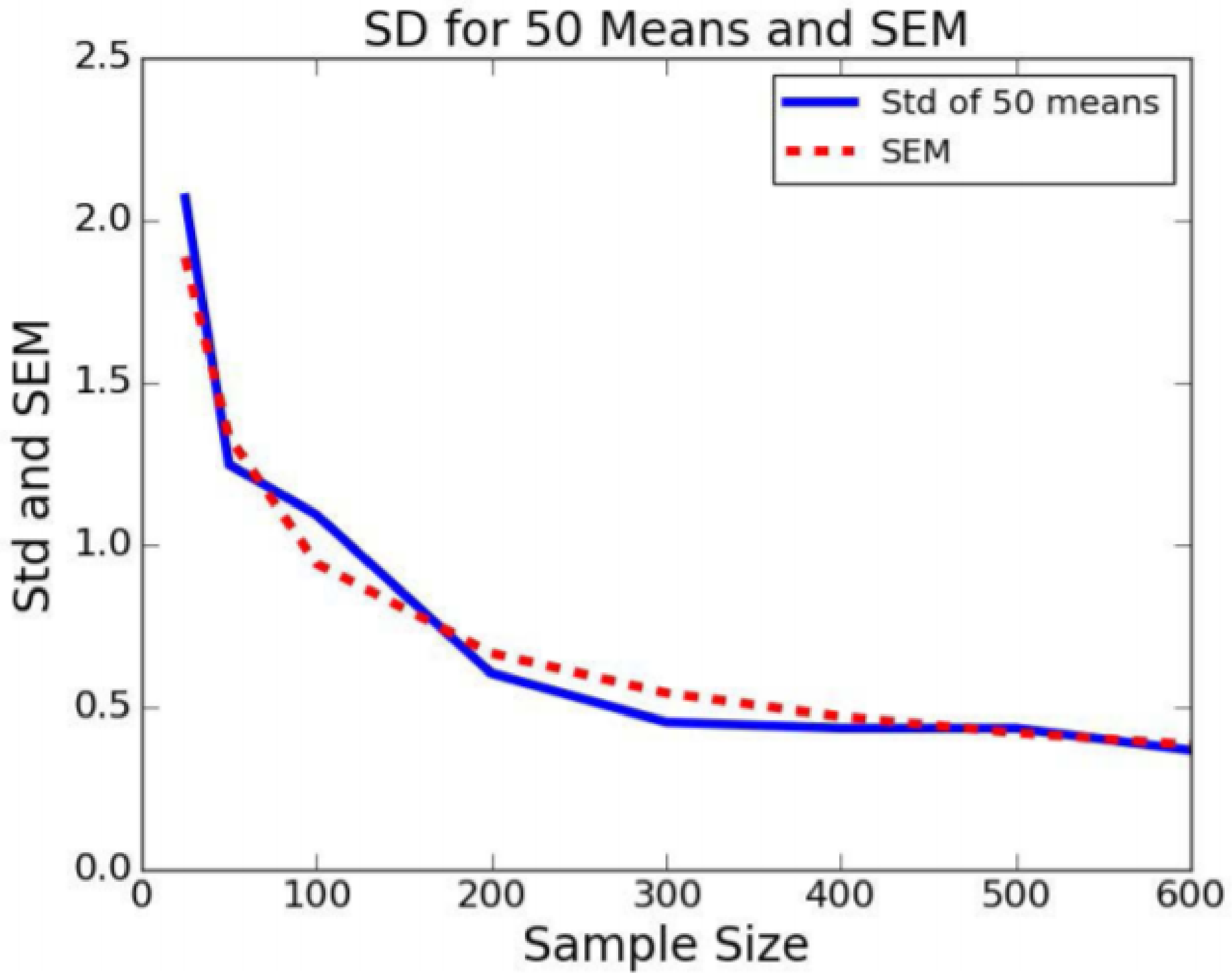
```
sampleSizes = (25, 50, 100, 200, 300, 400, 500, 600)
numTrials = 50
population = getHighs()
popSD = numpy.std(population)
sems = []
sampleSDs = []
for size in sampleSizes:
    sems.append(sem(popSD, size))
    means = []
    for t in range(numTrials):
        sample = random.sample(population, size)
        means.append(sum(sample)/len(sample))
    sampleSDs.append(numpy.std(means))
pylab.plot(sampleSizes, sampleSDs,
           label = 'Std of ' + str(numTrials) + ' means')
pylab.plot(sampleSizes, sems, 'r--', label = 'SEM')
pylab.xlabel('Sample Size')
pylab.ylabel('Std and SEM')
pylab.title('SD for ' + str(numTrials) + ' Means and SEM')
pylab.legend()
```

Standard Error of the Mean

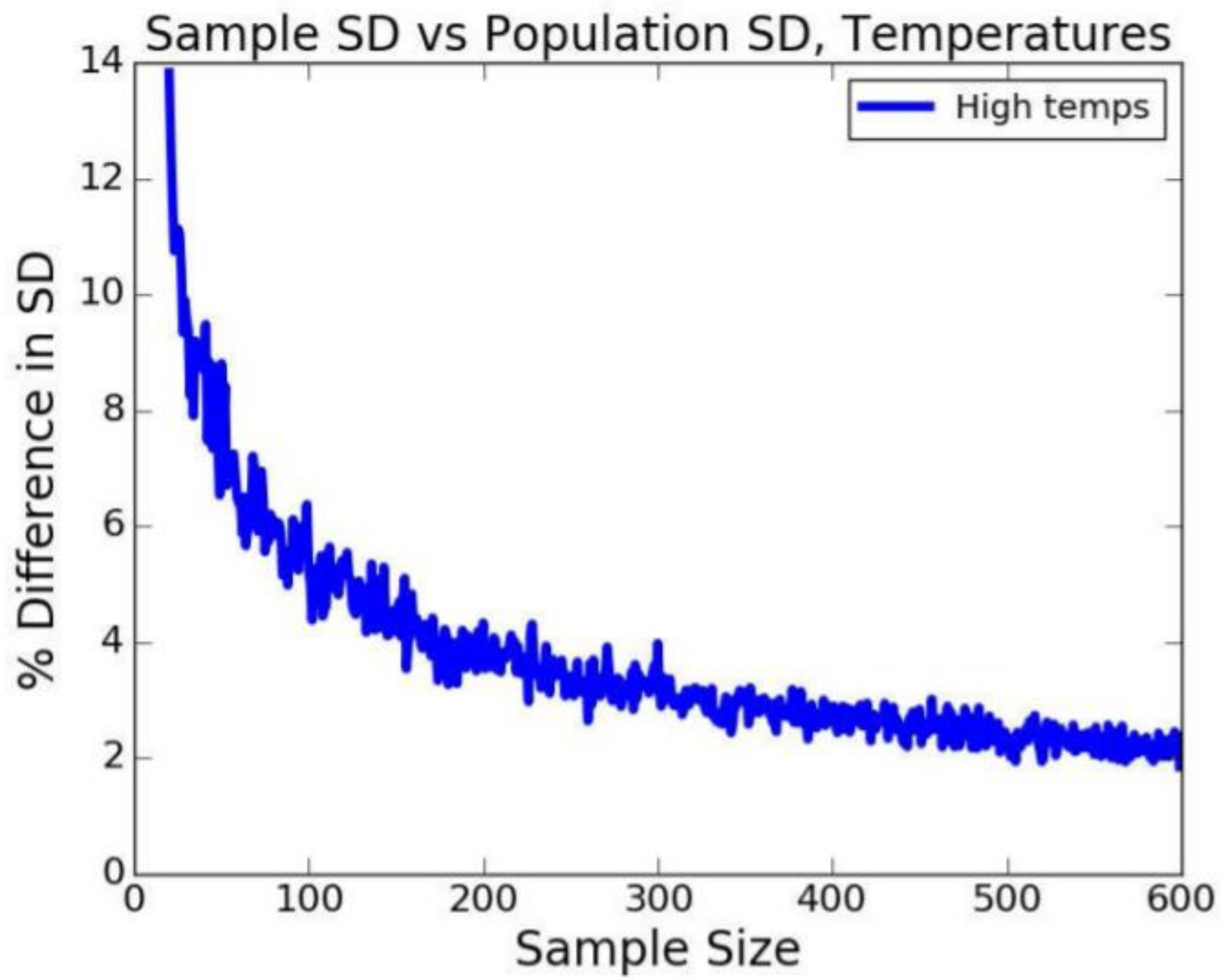
$$SE = \frac{\sigma}{\sqrt{n}}$$

But, we don't know standard deviation of population

How might we approximate it?

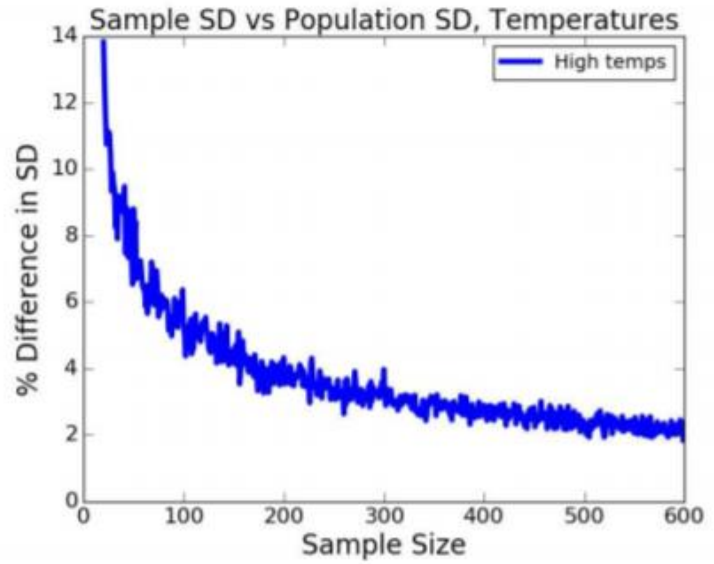


Sample SD vs. Population SD



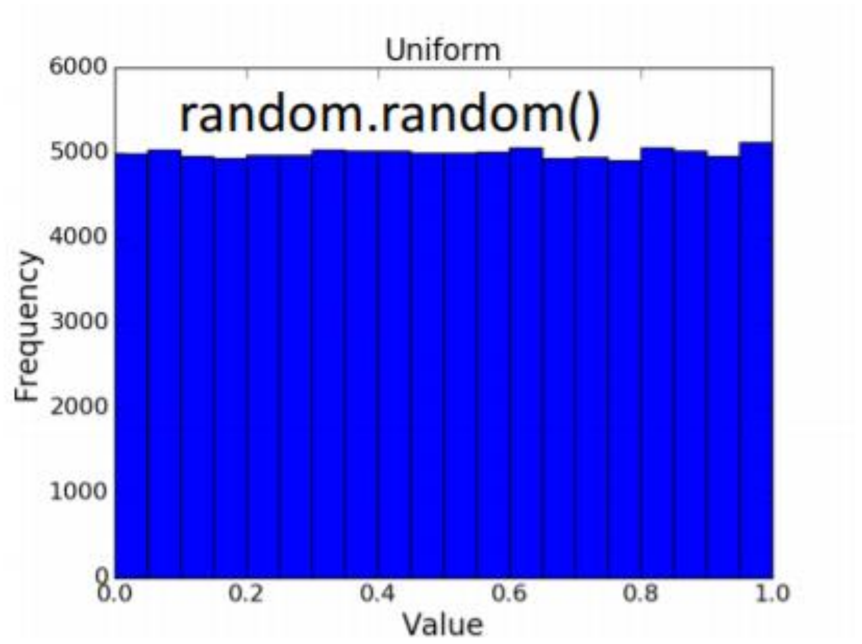
The Point

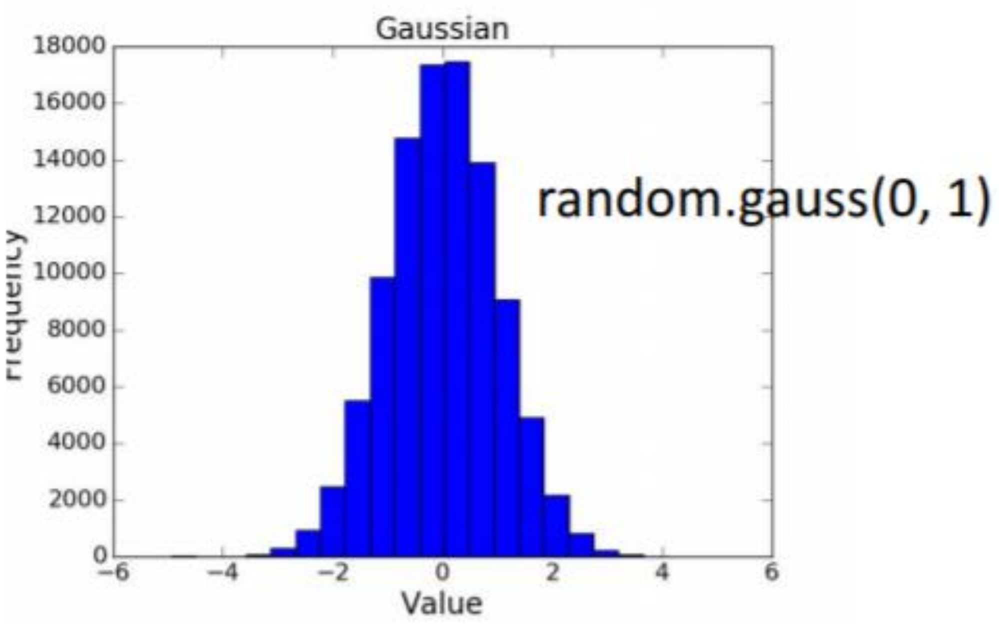
- Once sample reaches a reasonable size, sample standard deviation is a pretty good approximation to population standard deviation
- True only for this example?
  - Distribution of population?
  - Size of population?

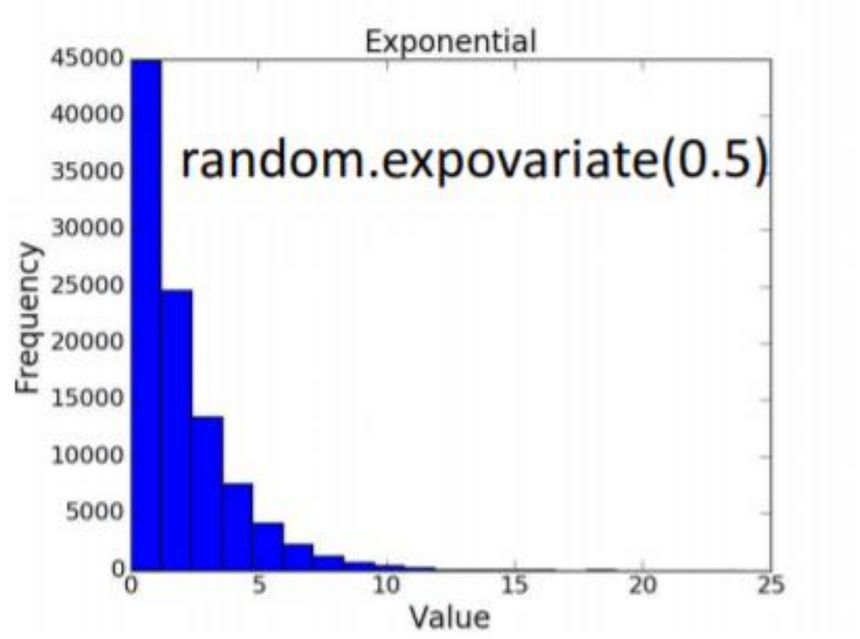


Looking at Distributions

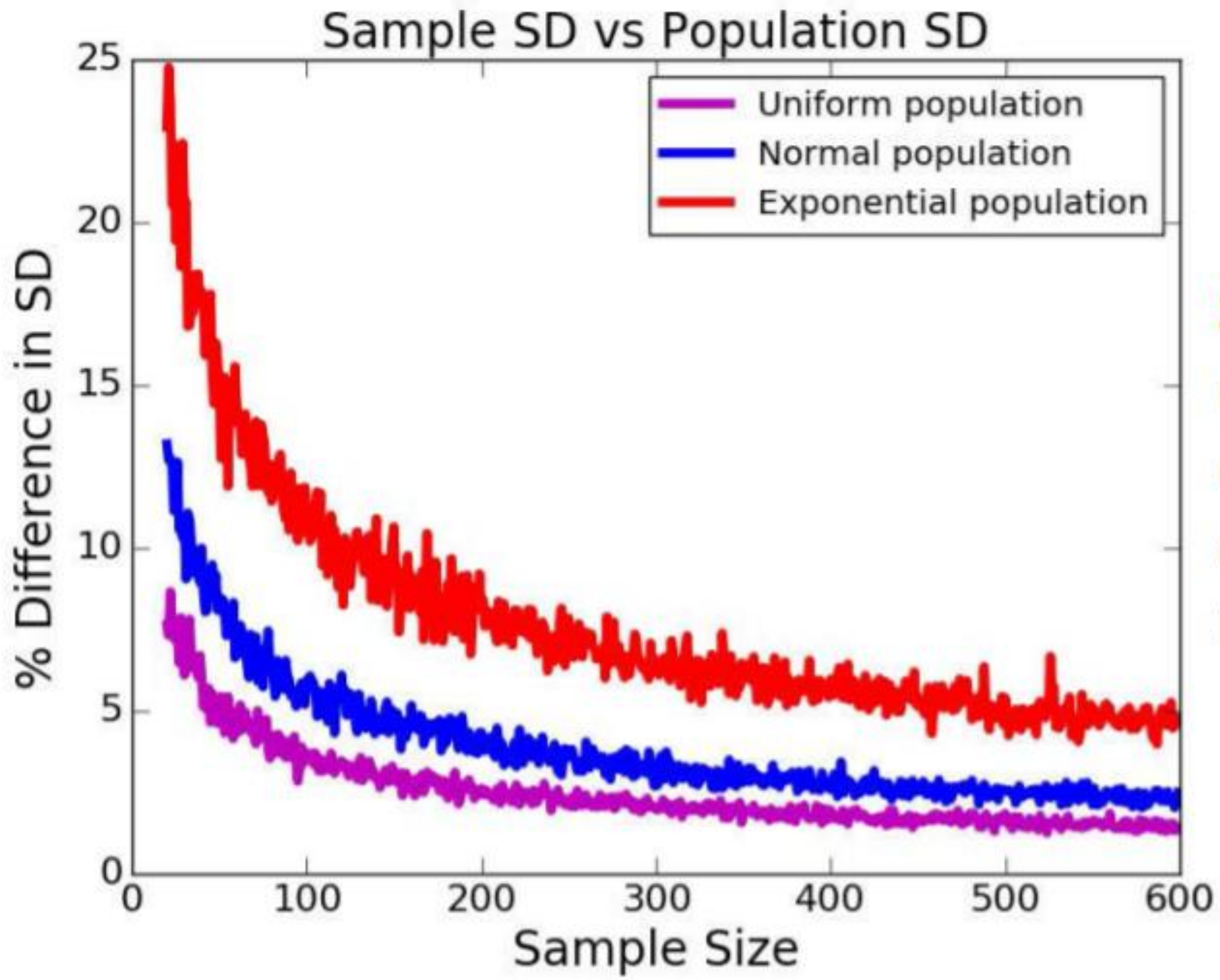
```
def plotDistributions():
    uniform, normal, exp = [], [], []
    for i in range(100000):
        uniform.append(random.random())
        normal.append(random.gauss(0, 1))
        exp.append(random.expovariate(0.5))
    makeHist(uniform, 'Uniform', 'Value', 'Frequency')
    pylab.figure()
    makeHist(normal, 'Gaussian', 'Value', 'Frequency')
    pylab.figure()
    makeHist(exp, 'Exponential', 'Value', 'Frequency')
```


Uniform
random.random()
Frequency
Value


Gaussian
random.gauss(0, 1)
Frequency
Value

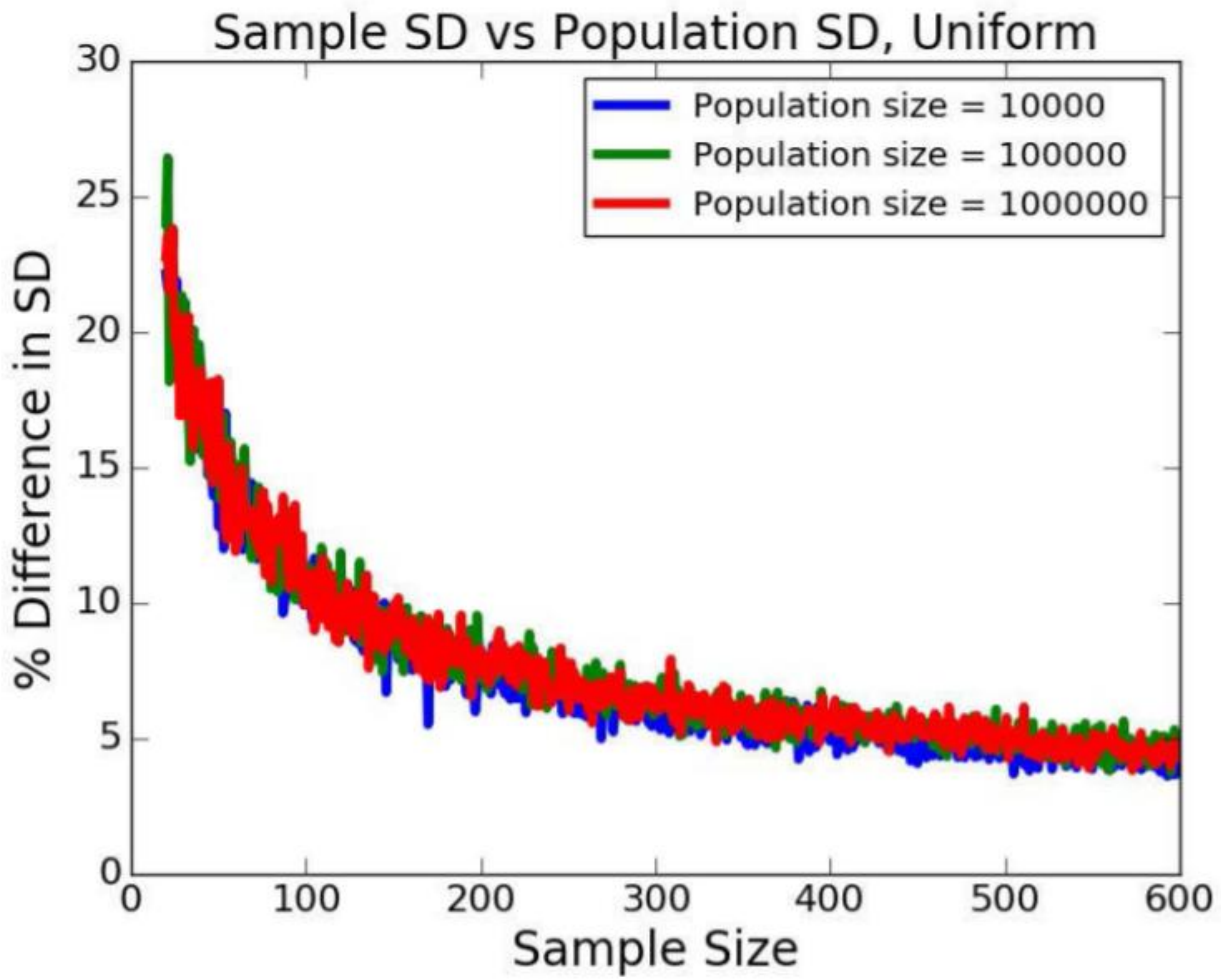

Exponential
random.expovariate(0.5)
Frequency
Value

Does Distribution Matter?



Skew, a measure of the asymmetry of a probability distribution, matters

Does Population Size Matter?



To Estimate Mean from a Single Sample

- 1) Choose sample size based on estimate of skew in population
- 2) Chose a random sample from the population
- 3) Compute the mean and standard deviation of that sample
- 4) Use the standard deviation of that sample to estimate the SE
- 5) Use the estimated SE to generate confidence intervals around the sample mean

Works great when we choose independent random samples.

Not always so easy to do, as political pollsters keep learning.

Are 200 Samples Enough?

```
numBad = 0
for t in range(numTrials):
    sample = random.sample(temps, sampleSize)
    sampleMean = sum(sample)/sampleSize
    se = numpy.std(sample)/sampleSize**0.5
    if abs(popMean - sampleMean) > 1.96*se:
        numBad += 1
print('Fraction outside 95% confidence interval =',
      numBad/numTrials)
```

```
Fraction outside 95% confidence interval = 0.0511
```