VARIANCE, CONFIDENCE INTERVALS, DISTRIBUTIONS

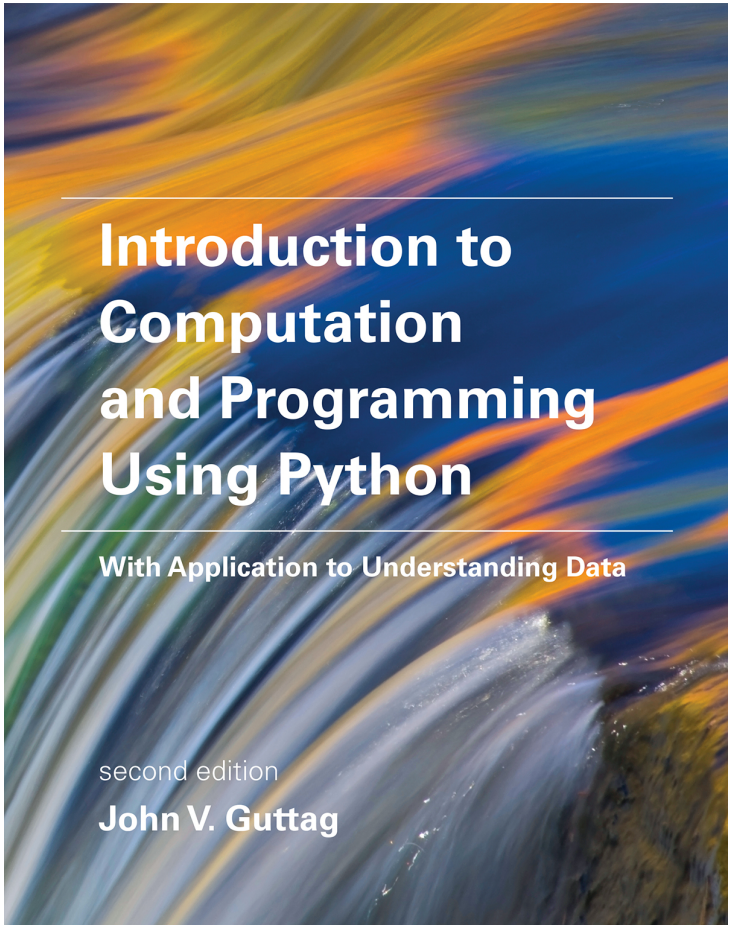
(download slides and .py files from Stellar to follow along)

John Guttag

MIT Department of Electrical Engineering and Computer Science

Assigned Reading

- Today:
  - Sections 15.3 – 15.4
- Next lecture:
  - Chapter 17



Sampling and Inferential Statistics

- Inferential statistics: making inferences about a populations by examining one or more random samples drawn from that population
  - Infer something about whole population based on statistics of sub-population
- With Monte Carlo simulation we can generate lots of random samples, and use them to estimate values
  - Roulette, dice rolling − estimate return on bets
  - Finding pi − estimate actual value of parameter
  - Integrating functions
- But how do we know when we have enough information to draw a conclusion with confidence?

Sampling Space of Possible Outcomes

- Never possible to guarantee perfect accuracy through sampling
- Not to say that an estimate is not precisely correct
- Key question:
  - How many samples do we need to look at before we can have justified confidence on our answer?
- Depends upon variability in underlying distribution

Quantifying Variation in Data (Recap)

$$variance(X) = \frac{\sum_{x \epsilon X}(x - \mu)^2}{|X|}$$

$$\sigma(X) = \sqrt{\frac{1}{|X|}\sum_{x \in X}(x - \mu)^2}$$

μ is the mean of the samples;
σ is the standard deviation of the samples

- Standard deviation simply the square root of the variance

A Simple Example (coinFlip.py)

- Flip a fair coin many times, measure heads/tails ratio and difference in heads and tails
- Let's do this for 20 trials and look at the mean and standard deviation over those trials

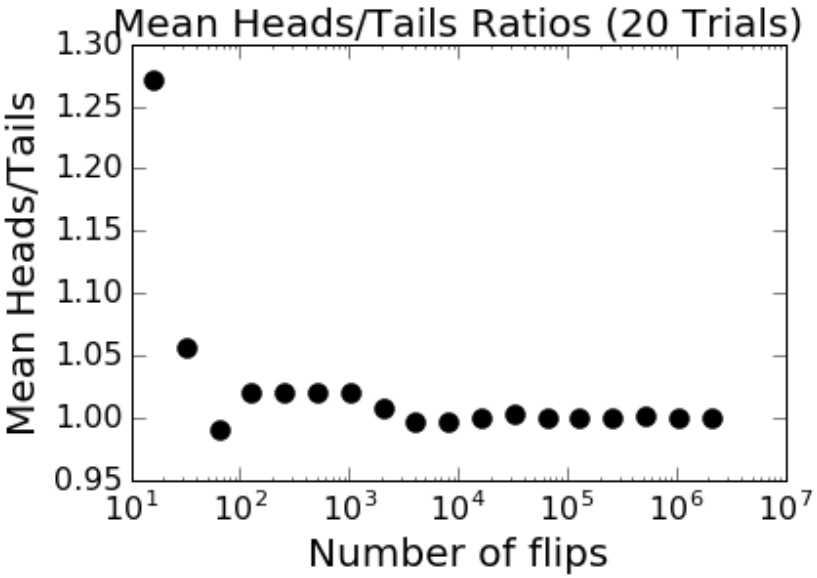
```
def runTrial(numFlips):
    numHeads = 0
    for n in range(numFlips):
        if random.choice(('H', 'T')) == 'H':
            numHeads += 1
    numTails = numFlips - numHeads
    return (numHeads, numTails)
```

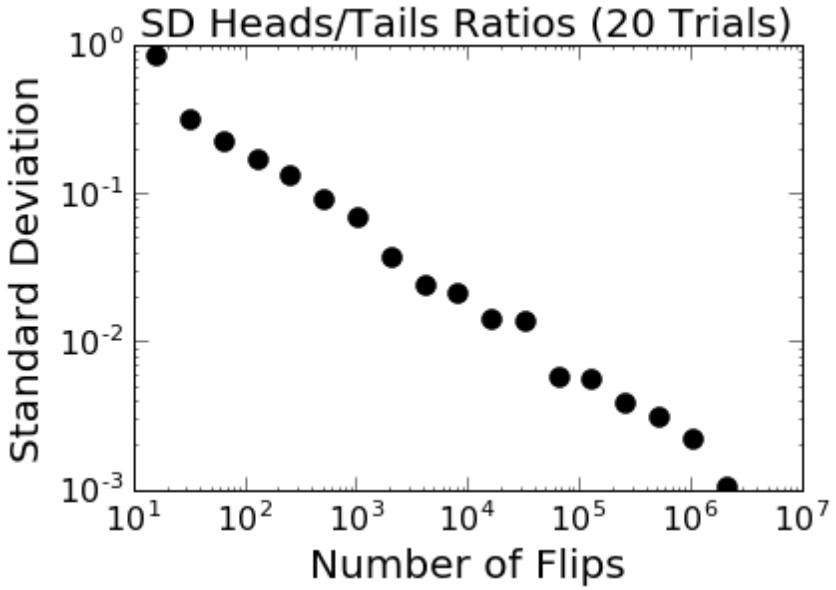
Code to Compute Statistics

```
def coinFlipPlot(minExp, maxExp, numTrials):
    """Assumes minExp, maxExp, numTrials ints >0; minExp < maxExp
    Plots summaries of results of numTrials trials of
    2**minExp to 2**maxExp coin flips"""
    ratiosMeans, diffsMeans, ratiosSDs, diffsSDs = [], [], [], []
    ratiosCVs, diffsCVs, xAxis = [], [], []
    for exp in range(minExp, maxExp + 1):
        xAxis.append(2**exp)
    for numFlips in xAxis:
        ratios, diffs = [], []
        for t in range(numTrials):
            numHeads, numTails = runTrial(numFlips)
            ratios.append(numHeads/numTails)
            diffs.append(abs(numHeads - numTails))
        ratiosMeans.append(sum(ratios)/numTrials)
        diffsMeans.append(sum(diffs)/numTrials)
        ratiosSDs.append(stdDev(ratios))
        diffsSDs.append(stdDev(diffs))
        ratiosCVs.append(CV(ratios))
        diffsCVs.append(CV(diffs))
```

- Plus some code to plot results

Interpreting Variance

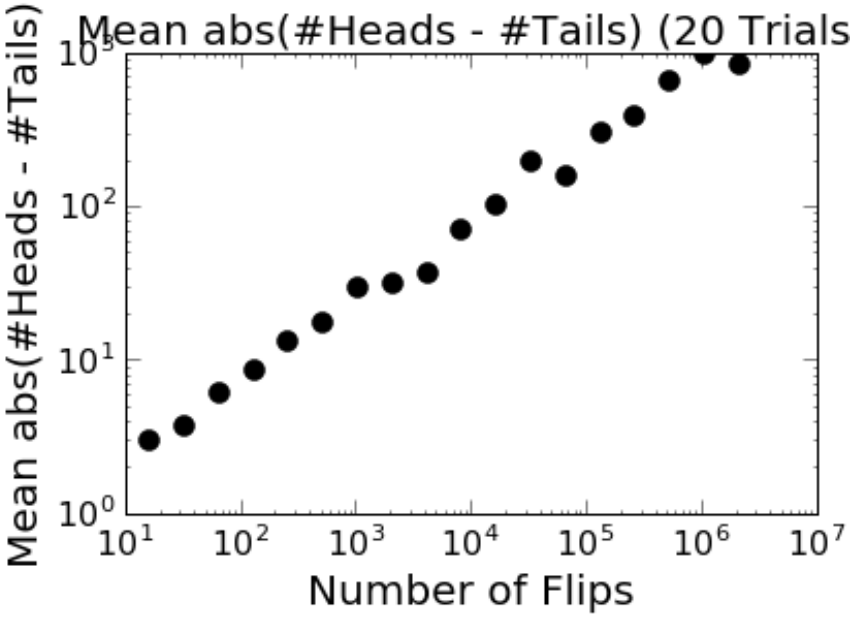


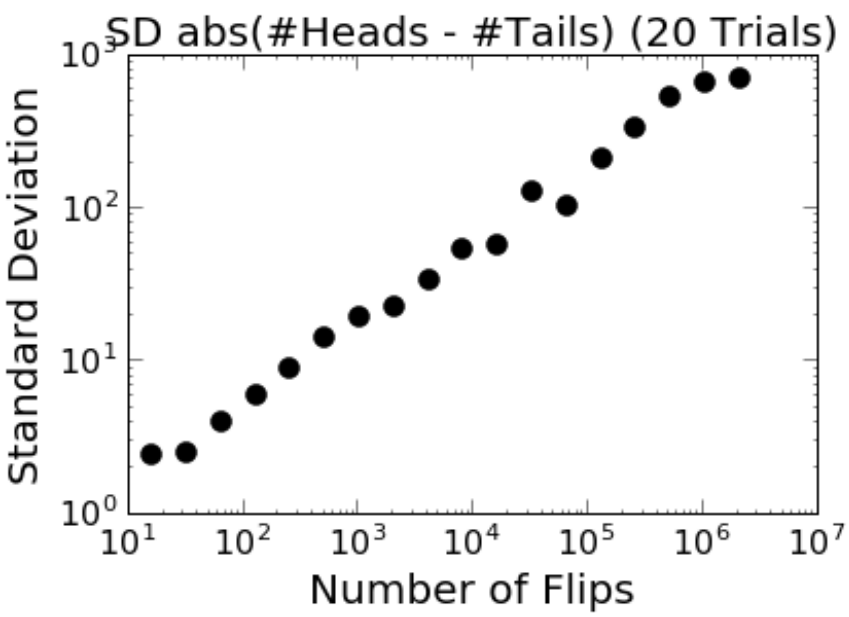


- Mean converging to 1; log of standard deviation decreasing linearly with log of number flips/trial
- At ~10^6 flips, standard deviation is 3 decimal orders of magnitude smaller than mean − so confident mean is ~1

Interpreting Variance

- Look at absolute difference in number of heads and number of tails





Mean |heads – tails| grows linearly with log number of flips

So does standard deviation?

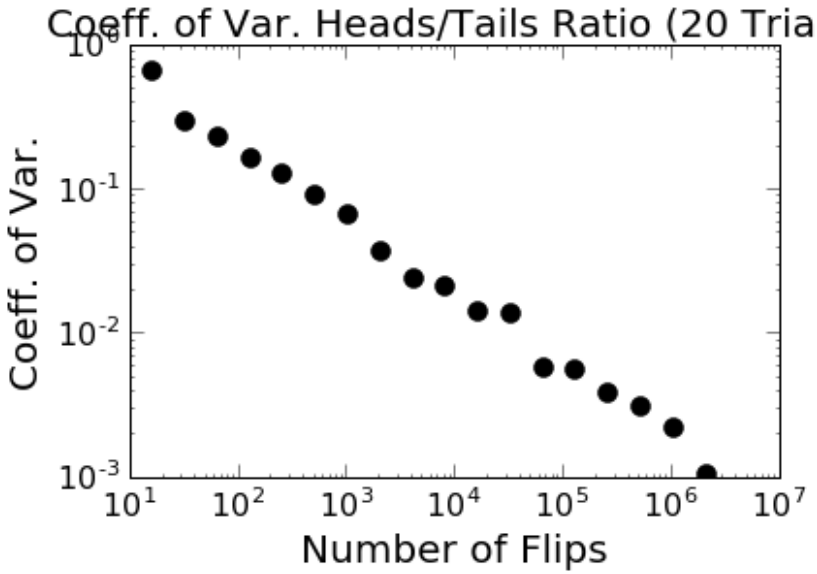
Does this mean less confidence in estimate of expected difference in heads and tails?

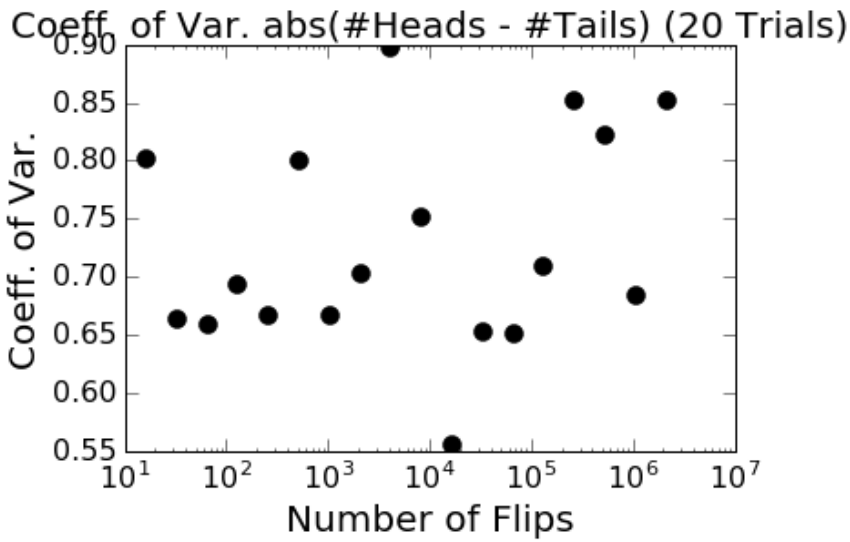
Coefficient of Variation

- Need to interpret standard deviation in context of mean
  - For example, mean of 1,000,000 and standard deviation of 100 is very different from mean of 100 and standard deviation of 100
- **Coefficient of variation:** ratio of standard deviation over mean

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

Coefficient of Variation





- CV for ratio looks just like standard deviation (which it should, because mean is essentially the same in all cases)
- CV for absolute difference has no trend, though seems to be settling into range 0.65 to 0.85
  - CV values less than 1 are considered low-variance

Summarizing

- Assume a population for which the probability of event E occurring is P
- Assume T trials of N examples each
- If N is large enough, each trial it will provide a good estimate of P
  - How large it needs to be depends upon variance of population
- To get a sense of variance of population we conduct multiple trials

Doesn't Have to Be a Probability

- Assume a population for which the mean value of some function of its members (e.g., weightOf) is some number M
- Assume T trials of N examples each
- Mean value of trials is best estimate of population mean
- If N is large enough, the mean value of each trial will approach that of the population
  - How large it needs to be depends upon variance of population
- To get a sense of variance of population we conduct multiple trials, rather than just increasing the trial size

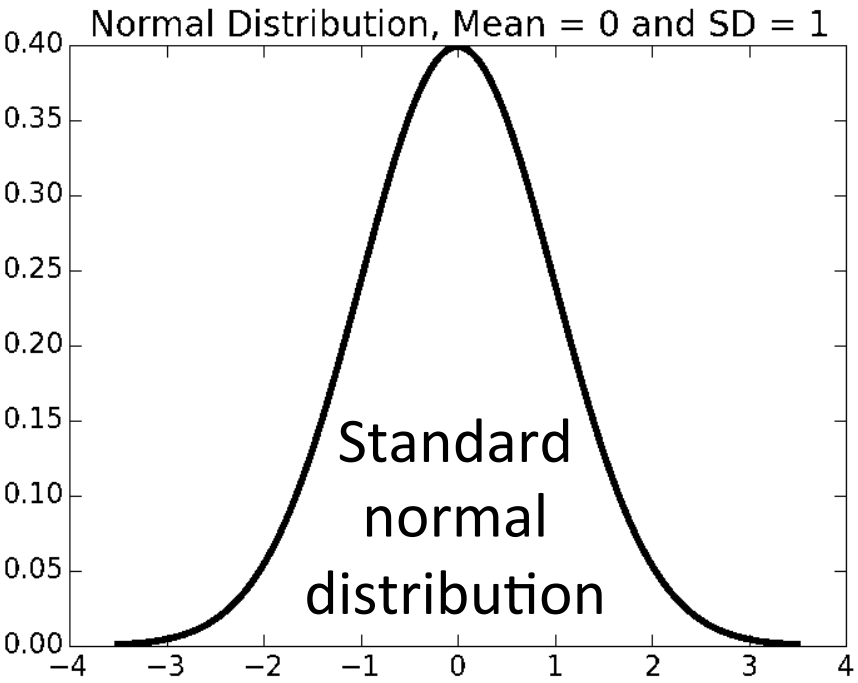
Choosing N Using Empirical Rule

- When the distribution of "errors" is normally distributed about the mean
  - ~68% of data within one standard deviation of mean
  - ~95% of data within 1.96 standard deviations of mean
  - ~99.7% of data within 3 standard deviations of mean
- Decide what confidence range and level you want
  - Increase N until you get there
- If N cannot be chosen (size of an experiment is fixed)
  - Report confidence level and range
  - At least you know your level of certainty

Normal Distributions

$P(x)=1/\sigma\sqrt{2\pi} * e\uparrow -(x-\mu)\uparrow 2 /2\sigma\uparrow 2$

Two parameters completely define a normal distribution

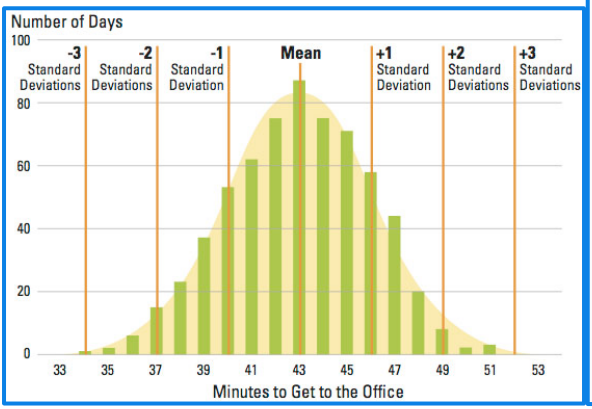


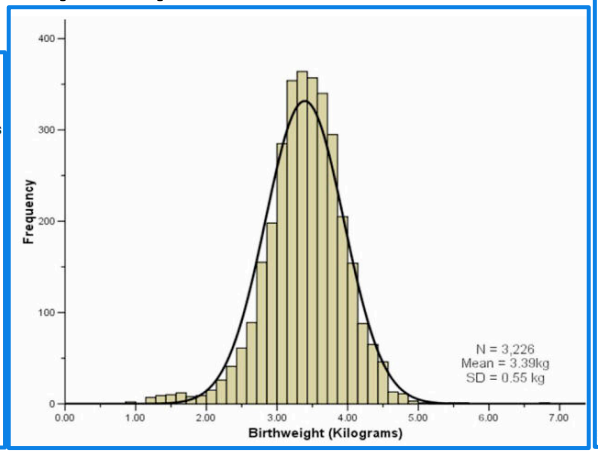
Symmetric about mean

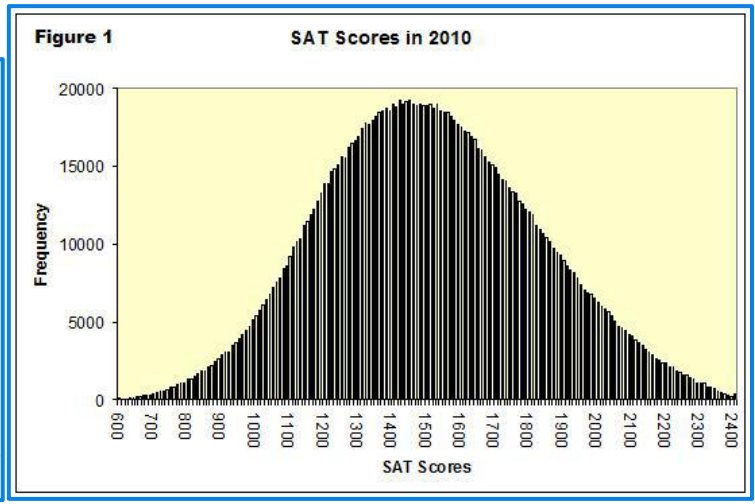
$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

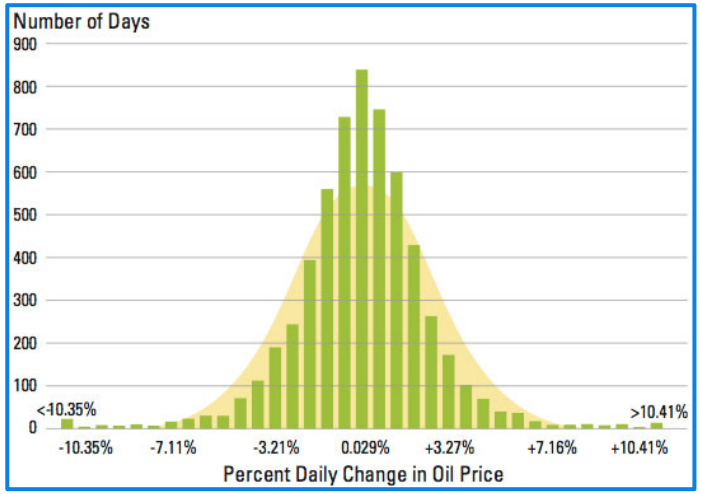
Everybody Likes Normal Distributions

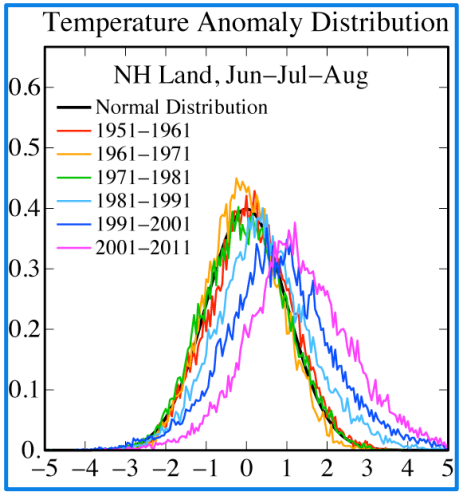
- Occur a lot!
- Nice mathematical properties

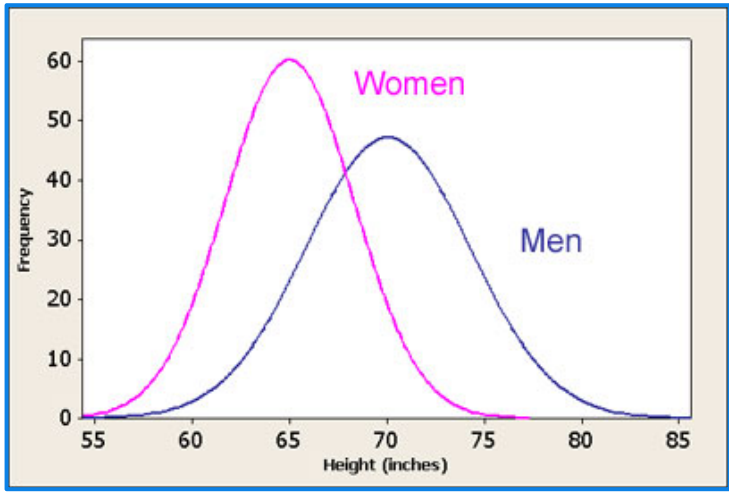










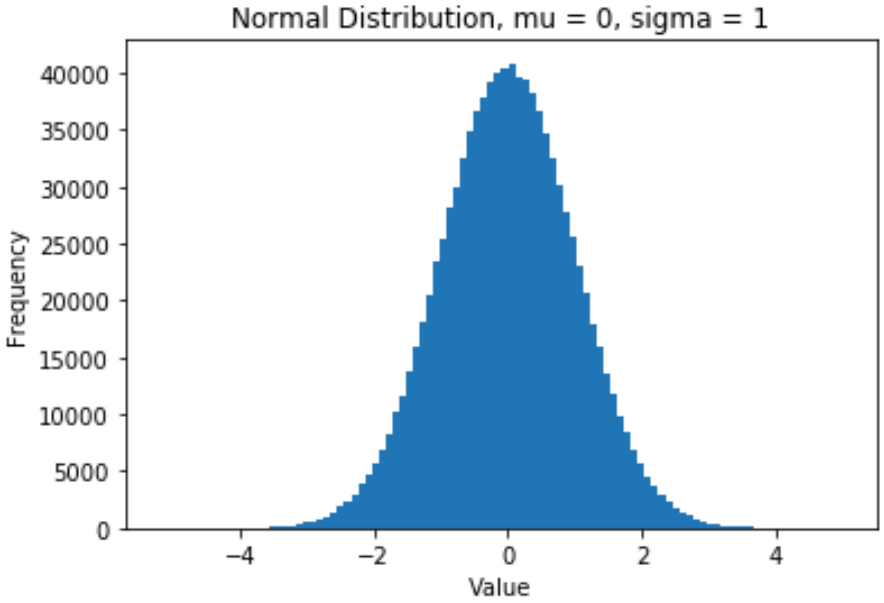


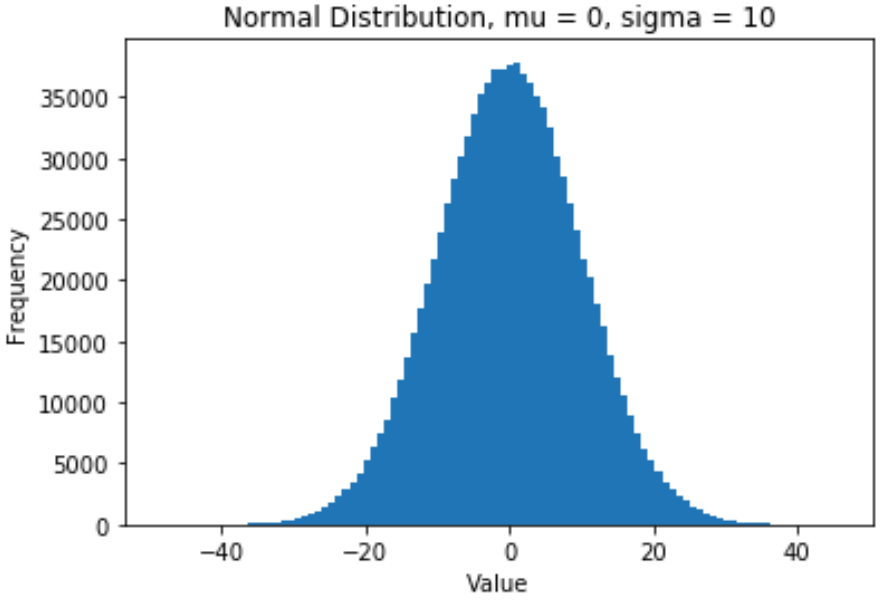
Generating Normally Distributed Data

```
mu = 0
numSamples = 1000000
for sigma in (1, 10):
    plt.figure()
    dist = []
    for i in range(numSamples):
        dist.append(random.gauss(mu, sigma))
    plt.hist(dist, bins = 100)
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.title('Normal Distribution, mu = ' + str(mu)\
              + ', sigma = ' + str(sigma))
```

Discrete
Approximation
of a Gaussian

Output


Normal Distribution, mu = 0, sigma = 1
Frequency
Value
40000
35000
30000
25000
20000
15000
10000
5000
0
−4
−2
0
2
4


Normal Distribution, mu = 0, sigma = 10
Frequency
Value
35000
30000
25000
20000
15000
10000
5000
0
−40
−20
0
20
40

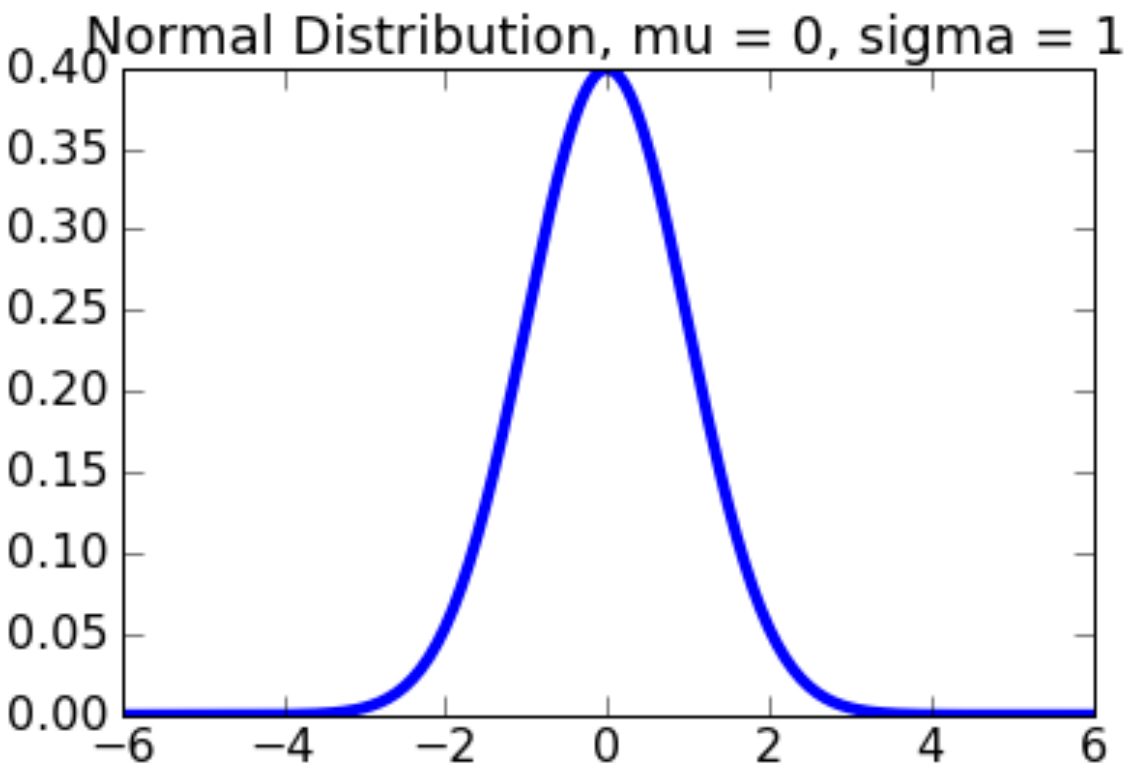
Defining Distributions

- Probability distribution captures notion of relative frequency with which a random variable takes on certain values
  - Discrete random variables drawn from finite set of values
  - Continuous random variables drawn from reals between two numbers (i.e., infinite set of values)
- For discrete variable, simply list the probability of each value, must add up to 1
- Continuous case trickier, can't enumerate probability for each of an infinite set of values

PDF's

- Distributions defined by *probability density functions* (PDFs)
- PDF at a point describes relative likelihood of that sample; more typically used to describe probability that a random variable lies between two points
- Defines a curve where the values on the x-axis lie between minimum and maximum value of the variable
  - Area under curve between two points is probability of example falling within that range
- For small range, PDF can be thought of as defining probability at a point

PDF for Standard Normal Distribution



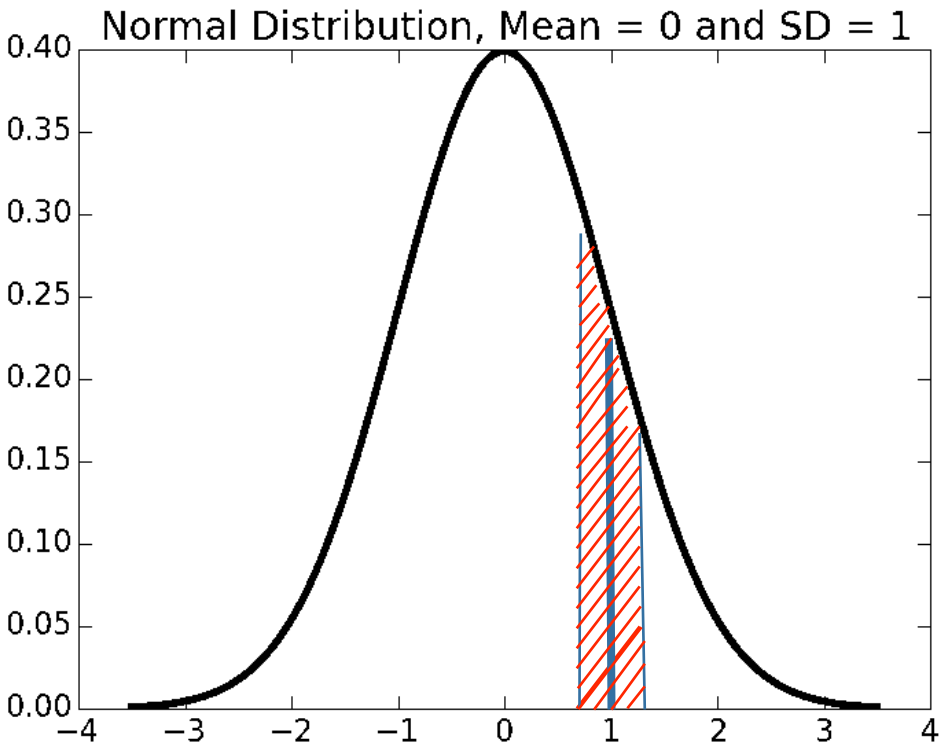
Are values on y-axis probabilities?

They are densities; i.e., derivative of cumulative distribution function.

Hence we use integration to interpret a PDF

PDF's define probabilities

- Area under curve over a span defines probability of value lying in that range



Limit as span tends to zero defines probability at a point

A Digression

- SciPy library contains many useful mathematical functions used by scientists and engineers
- scipy.integrate.quad has up to four arguments
  - a function or method to be integrated
  - a number representing the lower limit of the integration,
  - a number representing the upper limit of the integration, and
  - an optional tuple supplying values for all arguments, *except the first*, of the function to be integrated
- scipy.integrate.quad returns a tuple
  - Approximation to result
  - Estimate of absolute error

Checking the Empirical Rule

```
import scipy.integrate

def checkEmpirical(numTrials):
  for t in range(numTrials):
      mu = random.randint(-100, 100)
      sigma = random.randint(1, 100)
      print('For mu =', mu, 'and sigma =', sigma)
      for numStd in (1, 1.96, 3):
          area = scipy.integrate.quad(gaussian,
                                      mu-numStd*sigma,
                                      mu+numStd*sigma,
                                      (mu, sigma))[0]
          print(' Fraction within', numStd,
                'std =', round(area, 4))
```

```
checkEmpirical(5)
```

Checking the Empirical Rule

```
For mu = -66 and sigma = 73
 Fraction within 1 std = 0.6827
 Fraction within 1.96 std = 0.95
 Fraction within 3 std = 0.9973
For mu = 95 and sigma = 9
 Fraction within 1 std = 0.6827
 Fraction within 1.96 std = 0.95
 Fraction within 3 std = 0.9973
For mu = -35 and sigma = 16
 Fraction within 1 std = 0.6827
 Fraction within 1.96 std = 0.95
 Fraction within 3 std = 0.9973
For mu = 26 and sigma = 98
 Fraction within 1 std = 0.6827
 Fraction within 1.96 std = 0.95
 Fraction within 3 std = 0.9973
For mu = 15 and sigma = 61
 Fraction within 1 std = 0.6827
```

But All Distributions Are Not Normal

- Binomial distributions
- Uniform distributions
- Exponential distributions
- Many other, less common, distributions

Binomial Distributions

- What is the probability that a test succeeds exactly k times out of n independent trials (e.g., flip a coin n times, probability of exactly k heads)?
- If p is probability of success on one trial, then desired probability is:

$$P(k) = (n!k)p\uparrow k\,(1-p)\uparrow n-k$$

where

$$(n!k) = n!/k!(n-k)!$$

aka "n choose k"

- (Multinomial distribution generalizes to case of more than two, but a discrete, number of outcomes on each trial)

Binomial Distributions

- Let n be size of sample, and p be probability of event
- If n is large enough, then binomial distribution is approximated by a normal distribution, with

  mean = $n*p$ and variance = $n*p*(1-p)$

Generating Binomial

```
def choose(n,k):
    return math.factorial(n)/(math.factorial(k)*math.factorial(n-k))

def binomial(n,k,p):
    return choose(n,k)*(p**k)*((1-p)**(n-k))

def plotBinomial(n, p, title, xLabel, yLabel):
    xVals = []
    yVals = []
    for i in range(n+1):
        xVals.append(i)
        yVals.append(binomial(n,i,p))
    plt.figure()
    plt.title(title)
    plt.xlabel(xLabel)
    plt.ylabel(yLabel)
    plt.plot(xVals, yVals)

plotBinomial(10, 0.5, 'Binomial Distribution, p = 1/2',
             'Hits', 'Frequency')
plotBinomial(100, 0.5, 'Binomial Distribution, p = 1/2',
             'Hits', 'Frequency')
```

Checking Empirical Rule

```
def checkEmpiricalBinomial(n, prob):
    mu = n*prob
    sigma = (n * prob * (1-prob))**0.5
    for numStd in (1.0, 1.96, 3.0):
        tot = 0
        for i in range(int(round(mu-numStd*sigma)),
                       int(round(mu+numStd*sigma)) + 1):
            tot += binomial(n, i, prob)
        print(' Fraction within', numStd, 'std =',
              round(tot, 4))

for n in (50, 1000):
    for prob in (0.5, 0.75):
        print('For n =', n, 'prob =', prob)
        checkEmpiricalBinomial(n, 0.5)
```

Checking Empirical Rule?

- Empirical rule really applies to normal distributions
- But if binomial distribution approaches normal, is empirical rule a decent approximation?

```
For n = 50 prob = 0.5
 Fraction within 1.0 std = 0.7974
 Fraction within 1.96 std = 0.9672
 Fraction within 3.0 std = 0.9991
For n = 50 prob = 0.75
 Fraction within 1.0 std = 0.7974
 Fraction within 1.96 std = 0.9672
 Fraction within 3.0 std = 0.9991
For n = 1000 prob = 0.5
 Fraction within 1.0 std = 0.7033
 Fraction within 1.96 std = 0.9537
 Fraction within 3.0 std = 0.9974
For n = 1000 prob = 0.75
 Fraction within 1.0 std = 0.7033
 Fraction within 1.96 std = 0.9537
 Fraction within 3.0 std = 0.9974
```

Normal distribution:

- 68%
- 95%
- 99.7%

mean = $n*p$

variance = $n*p*(1-p)$

Uniform Distributions (already seen)

- All intervals of the same length have the same probability
- Probability that a value falls between x and y (where total range is a to b) is:

$$P(x,y) = \begin{cases} \frac{y-x}{b-a} & \text{if } x \geq a \text{ and } y \leq b \\ 0 & \text{otherwise} \end{cases}$$

- `random.uniform(min, max)` will draw an element within range with uniform probability
- Discrete version

$$P(x) = \begin{cases} 1/|S| & \text{if } x \in S \\ 0. & \text{otherwise} \end{cases}$$

- `random.choice(S)` will select an element from set with uniform probability

Uniform Distributions: Examples

- Coin flipping
- Dice rolling
- Roulette
- Quantization error in analog-to-digital conversion
- Don't often occur in nature

Mean and Variance

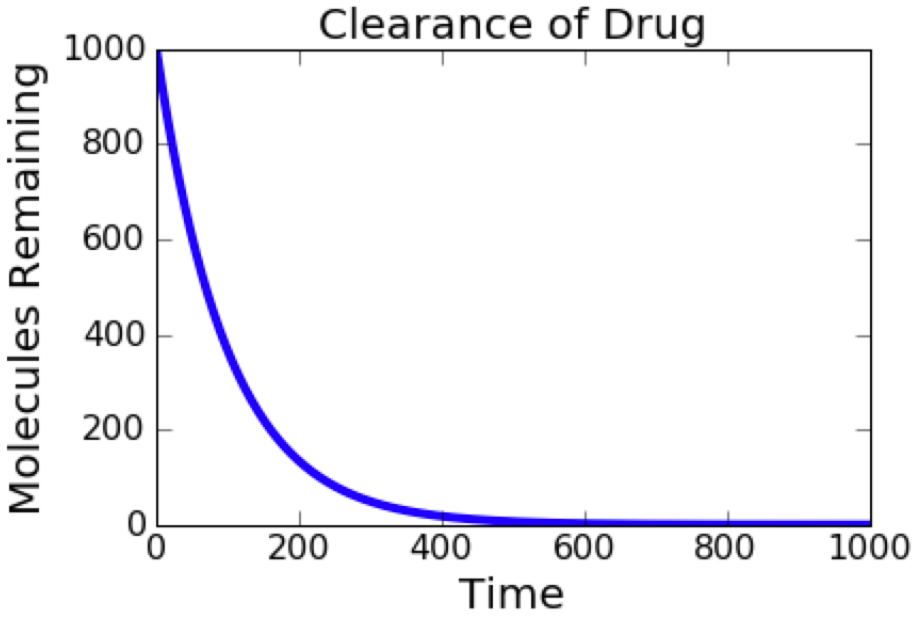
- Mean of a uniform distribution from *a* to *b* is *(b-a)/2*
- Standard deviation is *(b-a)/(12**0.5)*
- Does empirical rule work?

```
def checkEmpiricalUniform(n, a, b):
    mu = (b-a)/2
    sigma = (b-a)/(12**0.5)
    print('mu =', mu, '  sigma =', sigma)
    for numStd in (1, 1.96, 3):
        tot = 0
        for i in range(n):
            val = random.uniform(a, b)
            tot += (1 if abs(mu-val) <= numStd*sigma\
                    else 0)
        print('  Fraction within', numStd, 'std =',
              tot/n)
```

Exponential Distributions

- Suppose p is probability of an event occurring (e.g., a molecule of a drug being cleared from the body)
- Probability event has not occurred after t time steps (e.g., molecule still in body):

$(1-p)\uparrow t$



Exponential Distributions

- Probability density function

$$f(x)=\begin{cases}\lambda e^{-\lambda x} & if\ x \geq 0\\ 0. & if\ x<0\end{cases}$$

- Mean: $1/\lambda$
- Variance: $1/\lambda^2$
- Cumulative distribution function

$$F(x)=\begin{cases}1-e^{-\lambda x} & if\ x\geq 0\\ 0 & if\ x<0\end{cases}$$

- Can generate exponential distributions using `random.expovariate(lambd)`, where `lambd` is 1 divided by mean of distribution
- Discrete version called the geometric distribution

Exponential Distributions: Examples

- Modeling inter-arrival times, for example:
  - cars entering a highway,
  - or requests for a Web page,
  - or job requests on a server
- Time for a radioactive particle to decay (clicks on a Geiger counter)
- Time until default on payment to debt holders
- Service time of agents in a system (how long a bank teller takes to serve a customer)

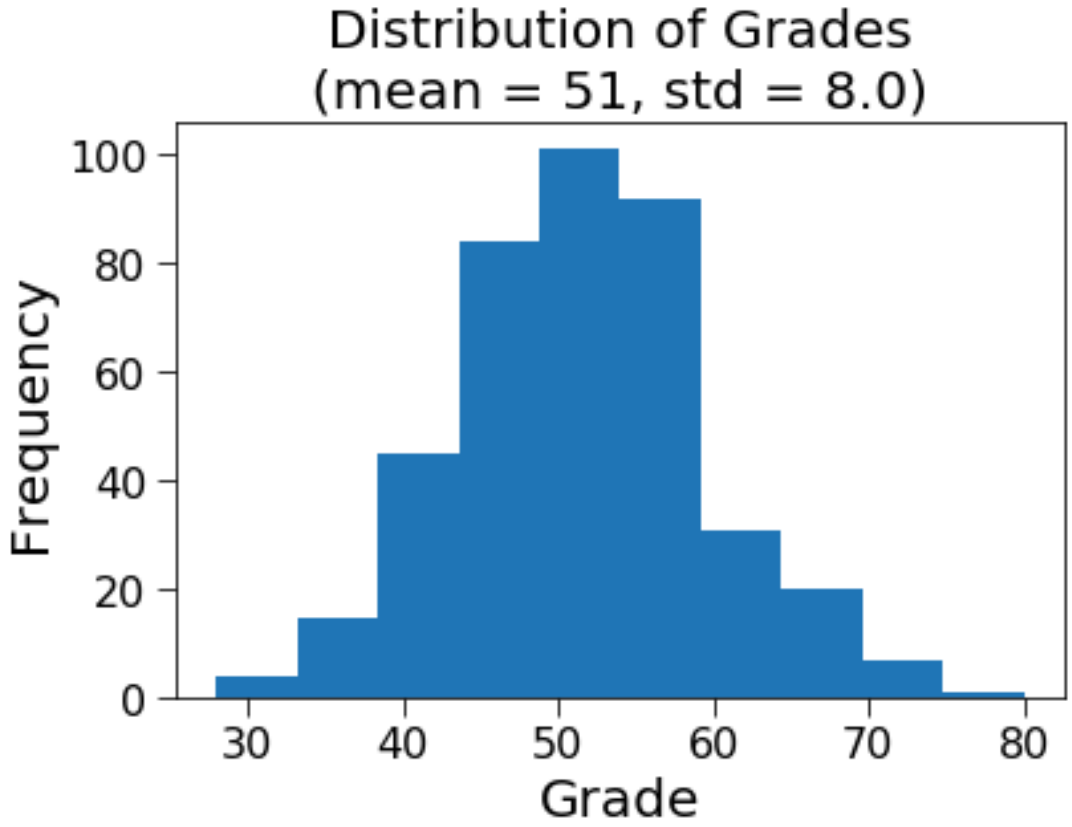
Checking Empirical Rule

```
def checkEmpiricalExponential(n, lambd):
    mu = 1/lambd
    sigma = mu
    print('mu =', mu, '  sigma =', sigma)
    for numStd in (1, 1.96, 3):
        tot = 0
        for i in range(n):
            val = random.expovariate(lambd)
            tot += (1 if abs(mu-val) <= numStd*sigma\
                    else 0)
        print('  Fraction within', numStd, 'std =',
              tot/n)

checkEmpiricalExponential(100000, 1/10)
```

Distribution Matters

- I'm only one and a half standard deviations from the mean and one 5 point question was mis-graded, I should get at least a B

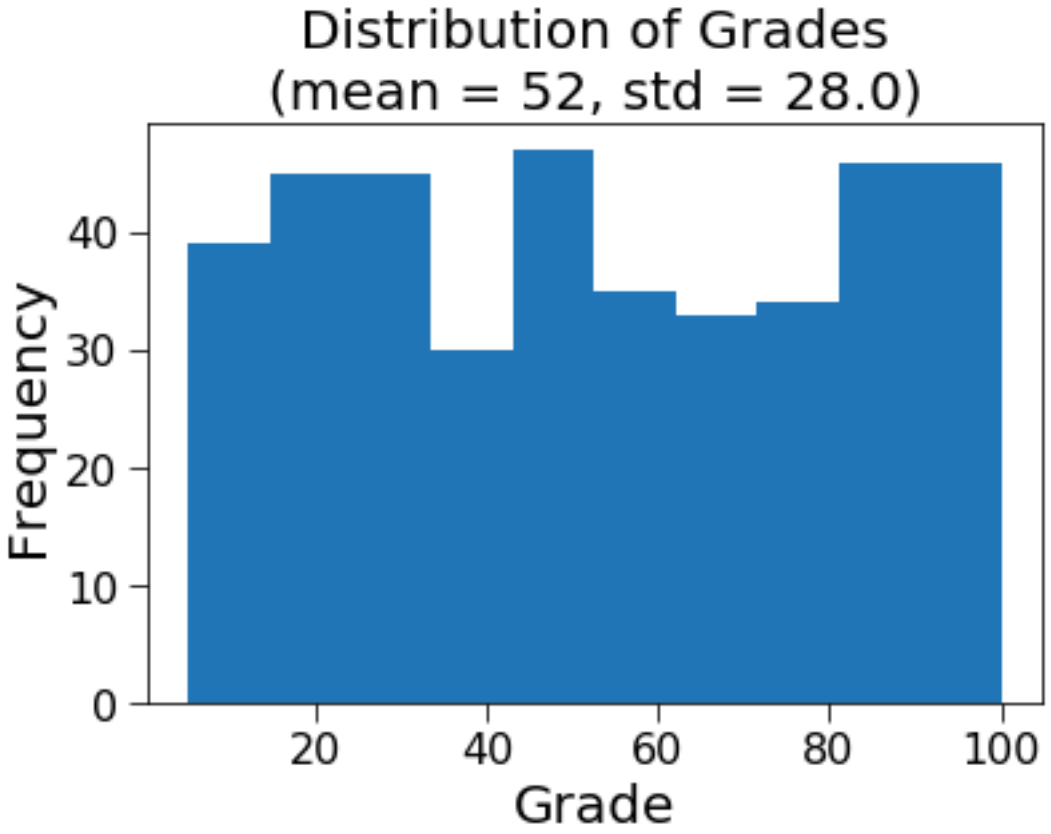


Your grade was 39.

Regrading one question might help

Distribution Matters

- I'm only one and a half standard deviations from the mean and one 5 point question was mis-graded, I should get at least a B

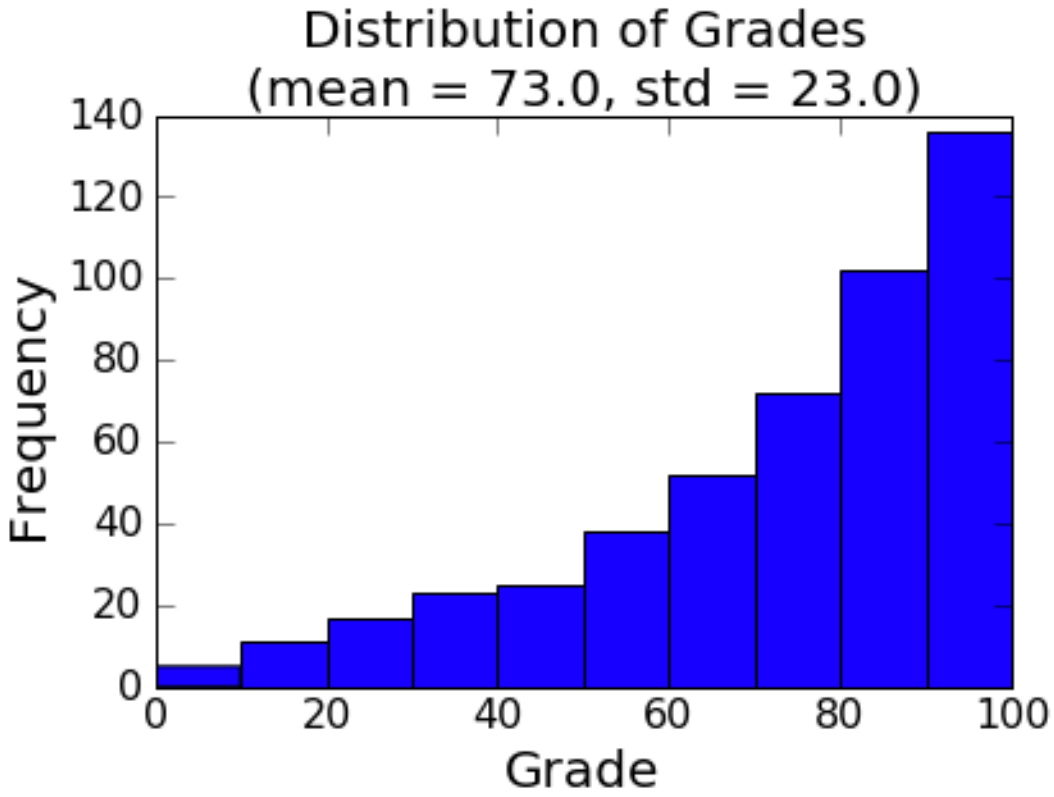


Your grade was 10!

Regrading one question won't help

Distribution Matters

- I'm only one and a half standard deviations from the mean and one 5 point question was mis-graded, I should get at least a B



Your grade was 38!

Regrading one question won't help

Summary

- Want to infer information from samples of space of possible outcomes of an experiment
- Can measure variation in outcome using standard deviation and coefficient of variation
- Empirical rule allows us to state results with level of confidence
- Empirical rule assumes zero mean error, and distribution of errors is normal (Gaussian)
- Many distributions are normal, but what if our distribution is not?
  - **Next lecture**


IT'S
QUIZ
TIME

Example of Variance and Deviation

- Estimate with different numbers of needles; track standard deviation in estimate, over 100 trials each

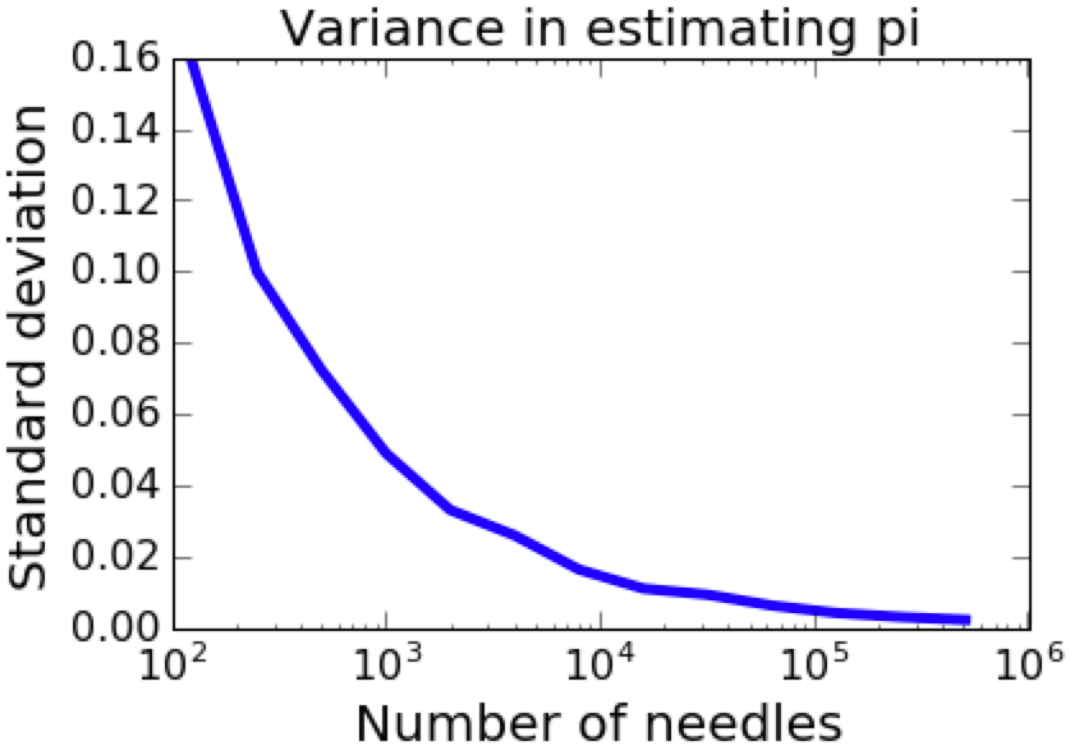
Est. = 3.14528, Std. dev. = 0.158152, Needles = 125
Est. = 3.13408, Std. dev. = 0.099927, Needles = 250
Est. = 3.13824, Std. dev. = 0.072391, Needles = 500
Est. = 3.13156, Std. dev. = 0.049081, Needles = 1000
Est. = 3.14386, Std. dev. = 0.033109, Needles = 2000
Est. = 3.14076, Std. dev. = 0.025974, Needles = 4000
Est. = 3.14007, Std. dev. = 0.016402, Needles = 8000
Est. = 3.14363, Std. dev. = 0.011073, Needles = 16000
Est. = 3.139702, Std. dev. = 0.00935, Needles = 32000
Est. = 3.141468, Std. dev. = 0.006298, Needles = 64000
Est. = 3.142094, Std. dev. = 0.004231, Needles = 128000
Est. = 3.141627, Std. dev. = 0.003094, Needles = 256000
Est. = 3.141542, Std. dev. = 0.002309, Needles = 512000

George-Louis Leclerc, Comte de Buffon

Example of Variance and Deviation

- Note how standard deviation shrinks as we increase number of needles in each trial
  - In general, increasing number of samples per trial tightens confidence

At what point would we say we have a tight enough estimate to be confident in that value?



Note x axis is on a log scale; Reduction in deviation slows down with increase in samples

Exploring Variance and CV

- Roll n dice (e.g., 3)
- Do this `rolls` times, count how often all n dice have the same value
- Repeat `numTrials` times, and compute mean and standard deviation
- What do we observe?

Exploring Variance and CV

```
Gathering stats using 100 trials
Data using  100 rolls, over  100 trials
Number hits 2.6 deviation 1.414213562373095
Data using  1000 rolls, over  100 trials
Number hits 27.39 deviation 5.051524522359562
Data using  10000 rolls, over  100 trials
Number hits 275.16 deviation 15.696318039591326

Gathering stats using 1000 trials
Data using  100 rolls, over  1000 trials
Number hits 2.828 deviation 1.6656578280066947
Data using  1000 rolls, over  1000 trials
Number hits 27.669 deviation 4.99534173005211
Data using  10000 rolls, over  1000 trials
Number hits 277.423 deviation 15.443512262435618

Gathering stats using 10000 trials
Data using  100 rolls, over  10000 trials
Number hits 2.7899 deviation 1.6600475866672437
Data using  1000 rolls, over  10000 trials
Number hits 27.7867 deviation 5.185865704971538
Data using  10000 rolls, over  10000 trials
Number hits 277.8204 deviation 16.63052446076193
```

Percent of rolls with hits seems roughly constant across number of trials and number of rolls

But standard deviation seems to be growing with number of rolls?

Exploring Variance and CV

```
Gathering stats using 100 trials
Data using  100 rolls, over  100 trials
Number hits 2.68 deviation 1.63 CV 0.608
Data using  1000 rolls, over  100 trials
Number hits 28.5 deviation 5.017 CV 0.176
Data using  10000 rolls, over  100 trials
Number hits 281.63 deviation 18.113 CV 0.064

Gathering stats using 1000 trials
Data using  100 rolls, over  1000 trials
Number hits 2.878 deviation 1.725 CV 0.6
Data using  1000 rolls, over  1000 trials
Number hits 27.933 deviation 5.39 CV 0.193
Data using  10000 rolls, over  1000 trials
Number hits 277.728 deviation 16.18 CV 0.058

Gathering stats using 10000 trials
Data using  100 rolls, over  10000 trials
Number hits 2.7681 deviation 1.634 CV 0.59
Data using  1000 rolls, over  10000 trials
Number hits 27.7618 deviation 5.218 CV 0.188
Data using  10000 rolls, over  10000 trials
Number hits 277.9116 deviation 16.256 CV 0.058
```

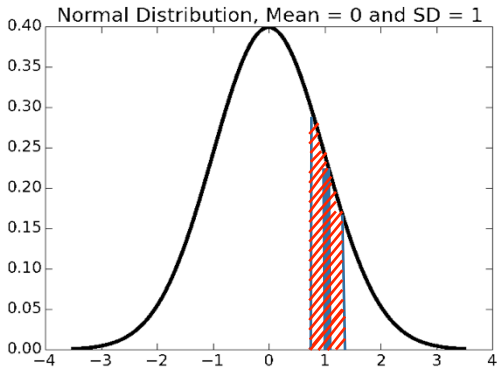
Percent of rolls with hits seems roughly constant across number of trials and number of rolls

But now coefficient of variation roughly constant across number of trials, decreasing with number of rolls

PDF's More Formally

- Let f(x) denote the probability density function
  - E.g. for a normal distribution

$$f(x)=1/\sigma\sqrt{2\pi}\ e\uparrow-(x-\mu)\uparrow 2\ /2\sigma\uparrow 2$$



- Probability of value lying in range near x:

$$P(x)=\int x-\varepsilon\uparrow x+\varepsilon\ f(t)dt$$

- Cumulative distribution function (probabili
variable is less than or equal to x:

$$F(x)=\int-\infty\uparrow x\ f(t)dt$$

- Note, integral over full range of values is 1, but value of PDF at a point can be greater than 1 (which is why we want integral in range, or area under the curve)