Inferential Statistics and Probability, Segment 3

Empirical Rule

~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

- Two key assumptions
  - The mean estimation error is zero
  - The distribution of the errors in the estimates is normal

Bell Curve

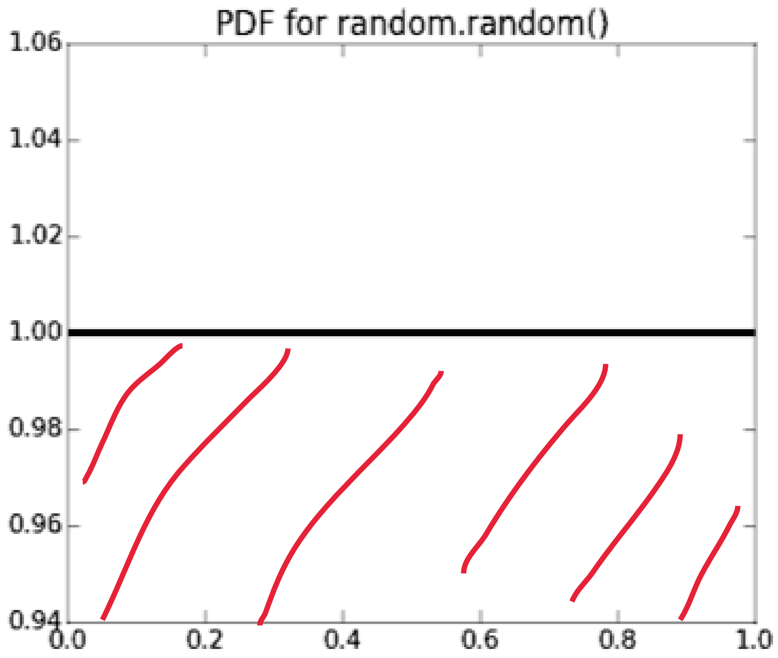
Defining Distributions

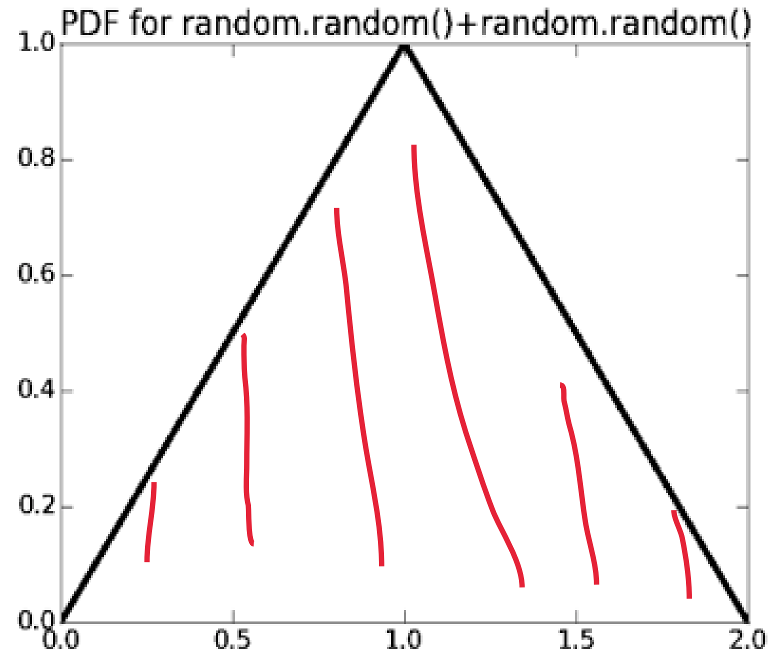
- Use a probability distribution
- Captures notion of relative frequency with which a random variable takes on certain values
    - Discrete random variables drawn from finite set of values
    - Continuous random variables drawn from reals between two numbers (i.e., infinite set of values)
- For discrete 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)
- Probability of a random variable lying between two values
- 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

Two PDF’s





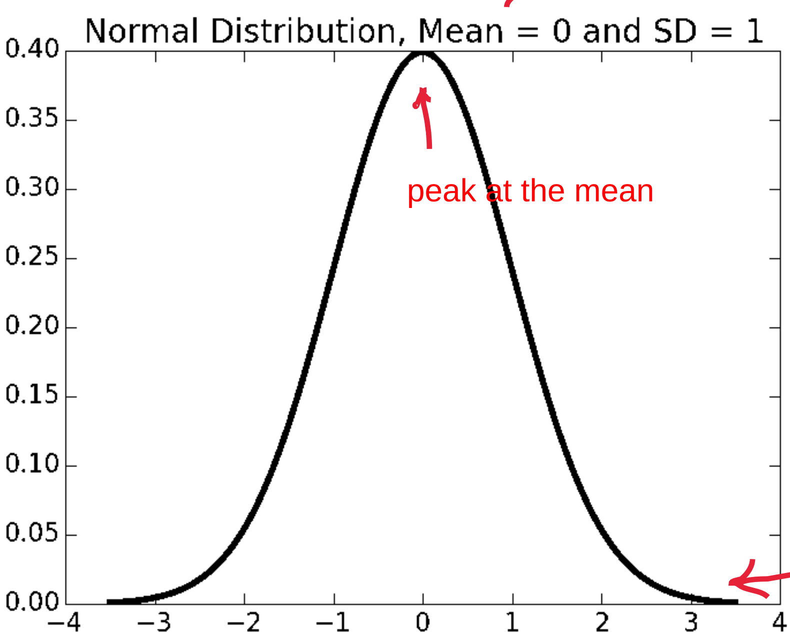
probability distribution functions

Normal Distributions

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

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

two parameters



μ = mean

σ (sigma) = SD

e → euler's number

2.718

in continues normal distribution, never really get to zero

Why We Like Normal Distributions

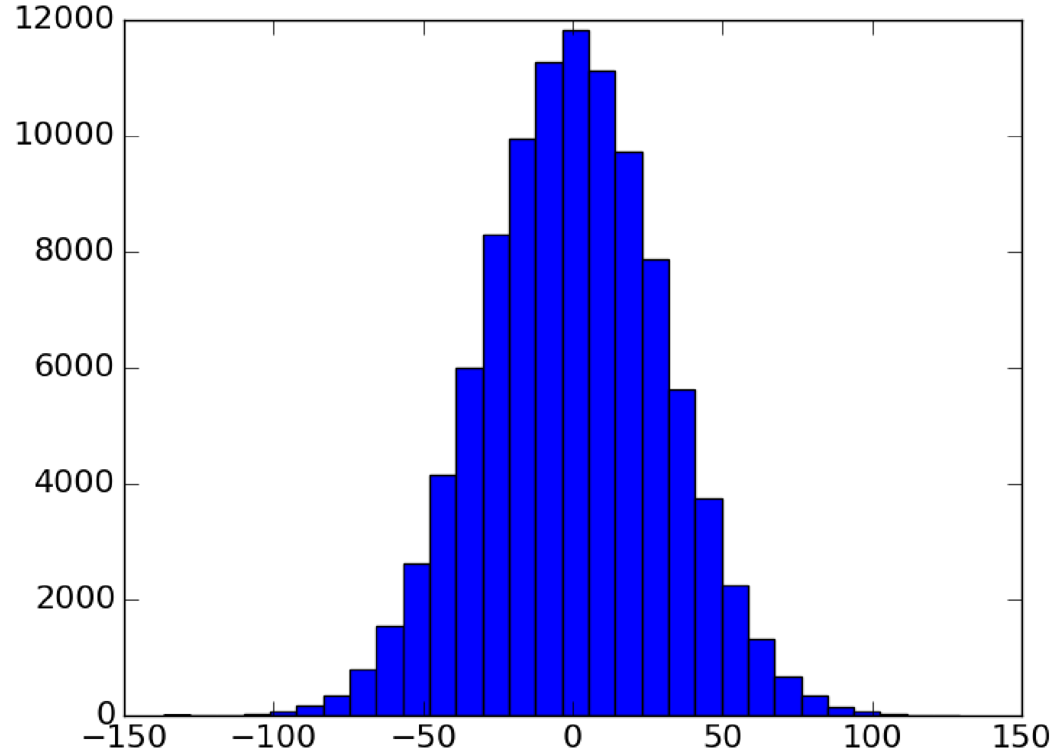
- Nice mathematical properties
- Occur a lot!

Generating Normal Distributions

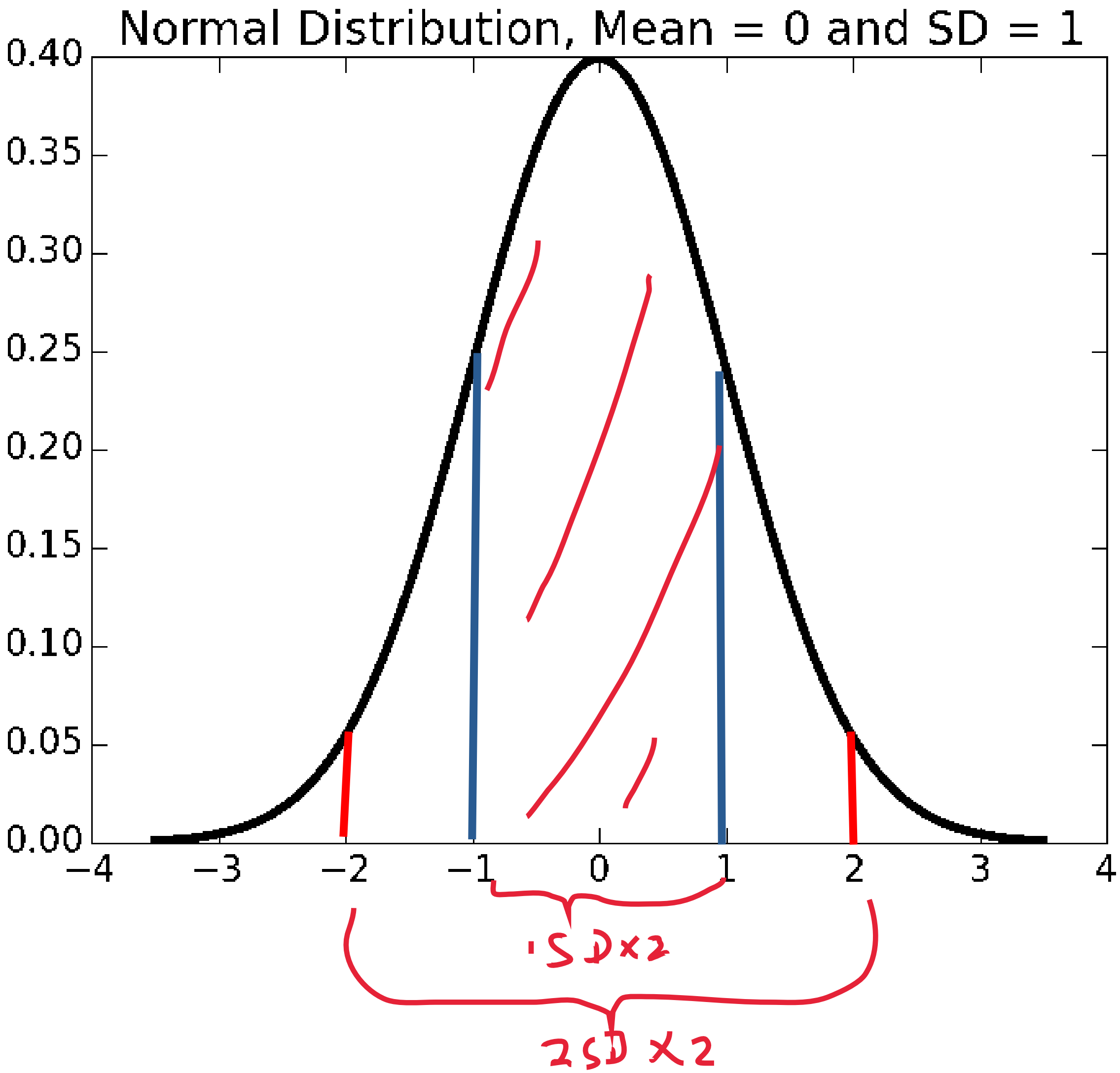
```
dist = []
for i in range(100000):
    dist.append(random.gauss(0, 30))
pylab.hist(dist, 30)
```

μ σ

Discrete Approximation



Empirical Rule



A Digression

- SciPy library contains my 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
    - a number representing the upper limit of the integration, and b
    - 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 Empirical Rule

```
import scipy.integrate

def gaussian(x, mu, sigma):
  factor1 = (1.0/(sigma*((2*pylab.pi)**0.5)))
  factor2 = pylab.e**-(((x-mu)**2)/(2*sigma**2))
  return factor1*factor2

def checkEmpirical(numTrials):
  for t in range(numTrials):
     mu = random.randint(-10, 10)
     sigma = random.randint(1, 10)
     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))
```

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

Result

emperica

For mu = 9 and sigma = 6
  Fraction within 1 std = 0.6827
  Fraction within 1.96 std = 0.95
  Fraction within 3 std = 0.9973
For mu = -6 and sigma = 5
  Fraction within 1 std = 0.6827
  Fraction within 1.96 std = 0.95
  Fraction within 3 std = 0.9973
For mu = 2 and sigma = 6
  Fraction within 1 std = 0.6827
  Fraction within 1.96 std = 0.95
  Fraction within 3 std = 0.9973

But Not All Distribution Are Normal

- Empirical works for normal distributions
- But are the outcomes of spins of a roulette wheel normally distributed?
- No, they are uniformly distributed
  - Each outcome is equally probable
- So, why does empirical rule work?