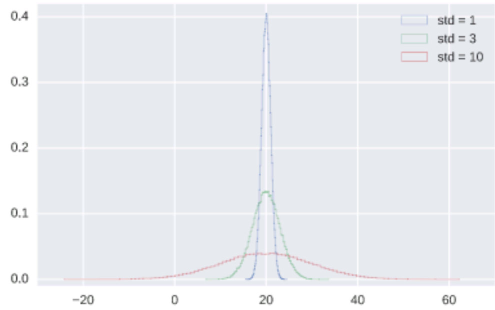
Ch 4 Thinking probabilistically-- Continuous variables

November-12-17 10:58 PM

```
#The Normal PDF
# Draw 100000 samples from Normal distribution with stds of interest: samples_std1, samples_std3, samples_std10
#mean = 20, std = 1, 3, 10
samples_std1 = np.random.normal(20, 1, size = 100000)
samples_std3 = np.random.normal (20, 3, size = 100000)
samples_std10 = np.random.normal (20, 10, size = 100000)

# Make histograms
_=plt.hist (samples_std1, bins = 100, normed=True, histtype = 'step')
_=plt.hist (samples_std3, bins = 100, normed=True, histtype = 'step')
_=plt.hist (samples_std10, bins = 100, normed = True, histtype = 'step')

# Make a legend, set limits and show plot
_ = plt.legend(('std = 1', 'std = 3', 'std = 10'))
plt.ylim(-0.01, 0.42)
plt.show()
```

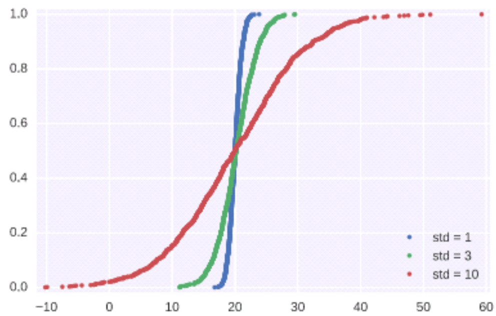



```
#The Normal CDF
# Generate CDFs
x_std1, y_std1 = ecdf (samples_std1)
x_std3, y_std3 = ecdf (samples_std3)
x_std10, y_std10 = ecdf (samples_std10)

# Plot CDFs
_=plt.plot (x_std1, y_std1, marker = ".", linestyle = 'none')
_=plt.plot (x_std3, y_std3, marker = ".", linestyle = 'none')
_=plt.plot (x_std10, y_std10, marker = ".", linestyle = 'none')

# Make 2% margin
plt.margins(0.02)

# Make a legend and show the plot
_ = plt.legend(('std = 1', 'std = 3', 'std = 10'), loc='lower right')
plt.show()
```

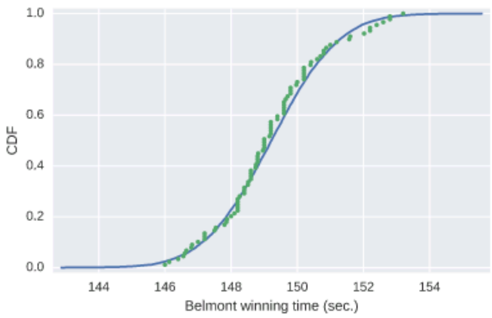



```
#Are the Belmont Stakes results Normally distributed?
# Compute mean and standard deviation: mu, sigma
mu = np.mean (belmont_no_outliers)
sigma = np.std (belmont_no_outliers)

# Sample out of a normal distribution with this mu and sigma: samples
samples = np.random.normal (mu, sigma, size = 10000)

# Get the CDF of the samples and of the data
x_theor, y_theor = ecdf (samples)
x, y = ecdf (belmont_no_outliers)

# Plot the CDFs and show the plot
_ = plt.plot(x_theor, y_theor)
_ = plt.plot(x, y, marker='.', linestyle='none')
plt.margins(0.02)
_ = plt.xlabel('Belmont winning time (sec.)')
_ = plt.ylabel('CDF')
plt.show()
```




```
#What are the chances of a horse matching or beating Secretariat's record?
# Take a million samples out of the Normal distribution: samples
samples = np.random.normal (mu, sigma, size = 1000000)

# Compute the fraction that are faster than 144 seconds: prob
prob = np.sum (samples <= 144)/len(samples)

# Print the result
print('Probability of besting Secretariat:', prob)
```

```
#If you have a story, you can simulate it!
def successive_poisson(tau1, tau2, size=1):
    # Draw samples out of first exponential distribution: t1
    t1 = np.random.exponential (tau1, size)
```

```
    # Draw samples out of second exponential distribution: t2
    t2 = np.random.exponential (tau2, size)

    return t1 + t2
```

```
#Distribution of no-hitters and cycles
# Draw samples of waiting times: waiting_times
waiting_times = successive_poisson (764, 715, 100000)

# Make the histogram
_=plt.hist(waiting_times, bins = 100, normed=True, histtype = 'step')


# Label axes
_=plt.xlabel ('Waiting Time (number of games)')
_=plt.ylabel ('PDF')

# Show the plot
plt.show ()
```