**Conjoint Probability:**

Writing means, the probability of occurrences A and B being both true.

In case of two independent occurrences, such as tossing a coin, the probability of both 2 occurrences to have heads is:

when two occurances are independent

However, if two occurrences are dependent, such as A being raining today and B being raining tomorrow, it is intuitive to infer that .

Thus, the probability of a conjunction is:

**Bayes Theorem**

Bayes theorem can be derived by picking up from above concept.

so, rewriting this is:

which leads to Bayes Theorem:

A simple example which utilizes this equation is, the cookie problem:

Suppose there are two bowls of cookies. Bowl 1 contains 30 vanilla cookies and 10 chocolate cookies. Bowl 2 contains 20 of each.

Now suppose you choose one of the bowls at random and, without looking, select a cookie at random. The cookie is vanilla. What is the probability that it came from Bowl 1?

Denoting B1 for the hypothesis that the cookie came from Bowl 1 and V for the vanilla cookie, we could write the problem as solving following equation:

Then assuming selecting either Bowl 1 or 2 is random, .

Selecting vanilla cookie from Bowl 1 is and selecting vanilla cookie out of total cookies, .

Thus,

**Bayesian Theorem with Diachronic Interpretation**

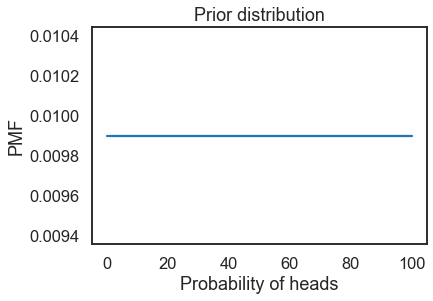
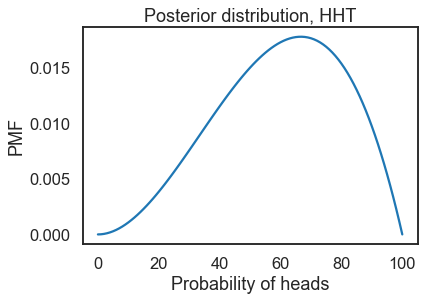
A more general way of interpreting Bayesian Theorem is to utilize it for updating the probability of a hypothesis, *H*, in light of some body of data, *D*.

This way of thinking about Bayes’s theorem is called the diachronic interpretation. “Diachronic” means that something is happening over time; in this case the probability of the hypotheses changes, over time, as we see new data.

In this interpretation, each term has a name:

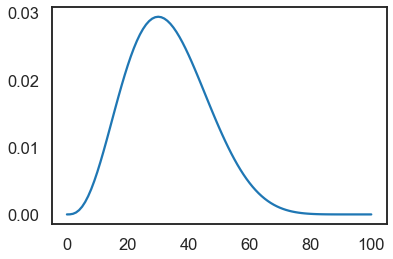
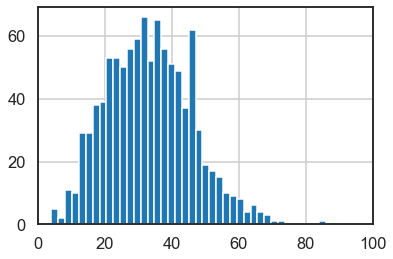
* is the probability of the hypothesis before we see the data, called the prior probability, or just **prior**.
* is what we want to compute, the probability of the hypothesis after we see the data, called the **posterior**.
* is the probability of the data under the hypothesis, called the **likelihood**.
* is the probability of the data under any hypothesis, called the **normalizing constant**

**Examples of Bayesian Statistics using Python**

1. **Euro Problem**  
     
     
   This problem requires proceeding in two steps. The first is to estimate the probability that the coin lands face up. The second is to evaluate whether the data support the hypothesis that the coin is biased.  
     
   Any given coin has some probability, *x*, of landing heads up when spun on edge. If a coin is perfectly balanced, we expect x to be close to 50%, but for a lopsided coin, x might be substantially different.  
     
   The key point in using Bayesian Statistics for solving this problem is not about the search for the exact value of *x*, per se, however the distribution of *x*.  
     
   Thus, *x* should be on a range between 0 to 100, where is the hypothesis that probability of heads is . Within python setup, one could starting the prior with a uniform distribution where the probability of is the same for all *x*, as such:  
     
      
     
   Then with incoming data, new information can be updated into the likelihood probability of coin toss resulting in heads with a function like:  
   If is true, the probability of heads is and the probability of tails is .  
     
   **def** Likelihood(self, data, hypo):  
    x = hypo  
    **if** data == ‘H’**:**  
    return x / 100.0  
    **else:**  
    return 1 – x / 100.0  
     
   If so, having updated with 3 sequential occurrences data, ie. **Head –** **Head -** **Tail**, the updated posterior distribution would look like:  
     
      
     
   In such matter, one could argue with 3 input data that, the maximum probability for having a head with a coin toss is 67% with a general average probability of 60% likelihood.  
     
   Again, the key take-away from this exercise is that, while one wishes to find the probability for the result of a coin toss, the output result from a Bayesian statistical analysis is not a single value but rather a probability distribution which holds the information for the general likelihood.

"When spun on edge 250 times, a Belgian one-euro coin came up heads 140 times and tails 110. 'It looks very suspicious to me,' said Barry Blight, a statistics lecturer at the London School of Economics. 'If the coin were unbiased, the chance of getting a result as extreme as that would be less than 7%.' "

From “The Guardian” quoted by MacKay, Information Theory, Inference, and Learning Algorithms.

1. **Random Selection with a Custom Probability Distribution**  
   Numpy library in python, gives some powerful tools for generating random selection data.  
     
   Let’s say we would like to select a number between 0~100, under a custom probability distribution, such that the distribution could look like below:  
     
      
     
   Then, numpy has following function to make a random selection within above distribution:  
     
     
   Using this, one can create a 1000 sample of selections as following:  
     
      
   As can be observed, the sample roughly follows the distribution as designed to be.

np.random.choice(a, size=None, replace=True, p=None)

key-Parameters

-----------

a : 1-D array-like or int

If an ndarray, a random sample is generated from its elements.

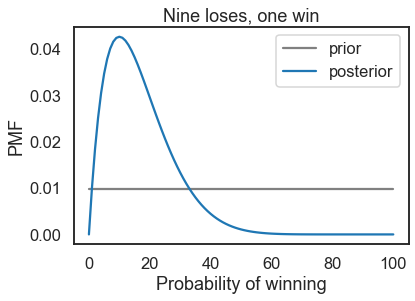
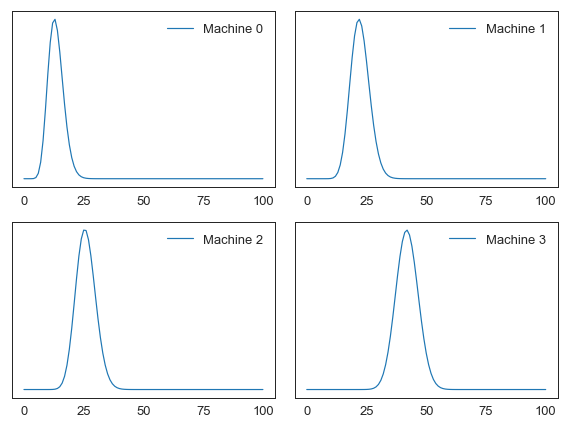
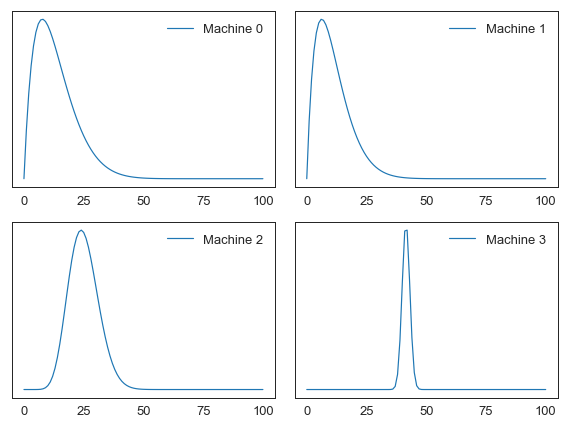
If an int, the random sample is generated as if a were np.arange(a)

p : 1-D array-like, optional

The probabilities associated with each entry in a.

If not given the sample assumes a uniform distribution over all

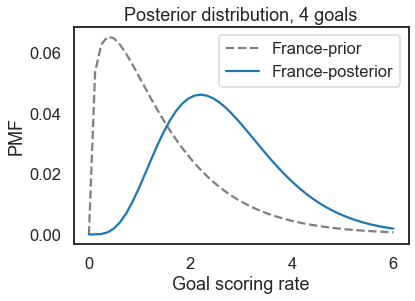
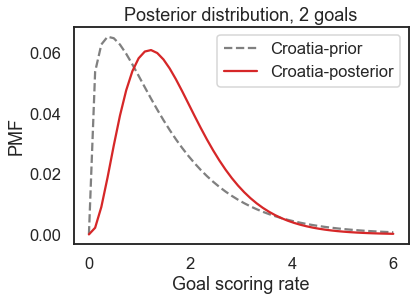
entries in a.

1. Bandit Problem  
     
     
   Considering that one could argue that an investment strategy is analogical to creating a pick-and-choose slot-machine strategy (ie. choosing the right investment strategy), this example is crucial for applying Bayesian Statistics for investment strategy.  
     
   All being equal in process with that of the “euro problem,” we’ll use the same selecting a number between 0~100 as our prior for eventually creating a success probability distribution for each slot machine, and the same likelihood update function. Therefore, if one has a slot-machine which ran a sequence of 1 win and 9 loses, hereafter denoted as ‘WLLLLLLLLL’, the posterior distribution would look like:  
     
      
   \*where grey line is the initial prior distribution and blue is the updated posterior after updating the occurrence data.  
     
   A distribution as such, gives meaningful information such as that the overall weighted-average probability of winning is 16.7% whereas the occurrence with the highest percentage for wining is 10%.  
     
   Now, say we have 4 slot machines with 10, 20, 30, 40% of winning probabilities and create random sequences of win or lose occurrence for each slot machines. For 100 sequence of occurrences for each slot machine, the winning distributions for each slot-machine would look like this:  
     
      
   Resulting average winning probabilities are, 13.39%, 22.32%, 25.89%, 41.96%.  
     
   Since, these winning probability distributions are updated after every round of plays by 4 slot-machine, one can use this data to choose which slot-machine to bet on with every round of play with the updated distribution.  
     
   By setting up the right kind of generating functions, following 1000 iterations of slot-machine plays can set-off following result:  
     
   num\_plays = 1000  
   count\_win = 0  
   for I in range(num\_plays):  
    count\_win = choose\_play\_update(  
    beliefs, record=True, count=count\_win)  
     
      
     
   After the run, when counting how many times each slot-machine was selected, the result is:  
     
   Slot-Machine 0 : 13 times  
   Slot-Machine 1 : 16 times  
   Slot-Machine 2 : 46 times  
   Slot-Machine 3 : 925 times  
     
   And the total number of wins was 397 times which is approximately 40% win out of 1000 runs.  
     
   However, on the other hand, if one does a random pick and choose slot-machine strategy, the result would have come up as:  
     
   Slot-Machine 0 : 252 times  
   Slot-Machine 1 : 248 times  
   Slot-Machine 2 : 257 times  
   Slot-Machine 3 : 243 times  
     
   And the total number of wins are 235 times which is approximately 23.5% win out of 1000 runs.  
     
   Thus, updating the model and choosing slot-machine through Bayesian approach not only **improved the outcome** but also **successfully allocated one’s bet to the most winning-probable slot-machine** through empirical trials.

Suppose you have several "one-armed bandit" slot machines, and reason to think that they have different probabilities of paying off.

Each time you play a machine, you either win or lose, and **you can use the outcome to update your belief about the probability of winning.**

Then, to decide which machine to play next, you can use the "Bayesian bandit" strategy.

1. **Synthetic Outcome Generation with Bayesian Statistics**  
   When predicting complex probability, Bayesian statistics can become handy for generating likely occurrences and deriving the outcome probability.  
     
   Also, it is important to know the different probability distributions available and choose the appropriate distribution to approximate the phenomena.  
     
   One type of problem might be solving the World Cup problem:  
     
     
   In order to setup a probability distribution for the number of goals a team is like to make, one could use a **Poisson distribution** which is non-negative and discrete (ideal for this case).  
     
   First one could set a prior distribution which would set the max-probable average number of goals for any team to be 1.4 goals (based on data from previous World Cups).  
   Then after, one could update the prior with 4 goals for France and 2 goals for Croatia, which would have an updated PMF as below:  
     
       
     
   Now that distributions for each team has been updated, this can become the basis for creating random game scores and predict the likelihood for the match-up outcome.

In the 2018 FIFA World Cup final, France defeated Croatia 4 goals to 2. Based on this outcome, we can answer the following questions:

1. How confident should we be that France is the better team?
2. If the same teams played again, what is the chance Croatia would win?