# An Introduction to Bayesian Parameter Learning (Version 2)

Murray Cantor, Ph.D.
IBM Distinguished Engineer (Ret.)
mcantor@cantorconsultingllc.com

*Ode to Bayes Theorem*

*Bayes Theorem, a cornerstone of probability*
*A way to update our beliefs, with clarity and sobriety*
*It tells us how to revise, when faced with new information*
*To move from doubt to certainty, with elegant precision*

*We start with a hypothesis, a thought, or a conjecture*
*We seek out evidence, to see if we can verify*
*We use Bayes Theorem, to calculate the chance*
*That our hypothesis holds true, in this circumstance*

*It's a mathematical rule, that helps us to deduce the likelihood of something,*
*With our current produce*
*Of knowledge and data, we can use it to infer*
*The truth of a matter, without any demur*

*So let us embrace Bayes, and all that it can teach us*
*To update our views, and not become too suspicious*
*For with this powerful tool, we can find the right path*
*And make better decisions, with math*

- ChatGPT

## Introduction

Parameters are ‘constants’ that are used to characterize various kinds of systems. Examples include:

- Physical – such as the temperature and mass of an object, and Hubble’s constant
- Statistical – population characteristics such as means and standard deviations
- Processes – such as efficiency, throughput, and time-to-failure

Parameters are determined by taking measurements so that any estimate will have some uncertainty depending on the size of the data and the underlying model. To take action based on a parameter depends not only on its estimated value but also on its uncertainty. If we are not confident we have a good value, we might be more hesitant to act.

One of the critical issues in learning a parameter is deciding when our confidence is high enough to take action. This especially applies when there isn't enough data to use classical frequentist methods.

Some example applications include:

- Measuring a constant c with a noisy measurement instrument with sparse data like the James Webb telescope or the Large Hadron Collider. You might assume the noise is Gaussian, so measures are of the form *c + normal(mean = 0, std = s) = normal(mean=c, std = s)*. Both c and s are unknown and need to be estimated from the observations, and so are themselves random variables.
- The process of generating the measures is not controllable in the Deming sense. This can happen when measuring the progress in novel development efforts. This also can be treated as a Gaussian process with unknown parameters. (Some might prefer Weibull.)
- The measurements are expensive to obtain. This can happen for expensive products' reliability testing (time to failure). In this case, one can treat the λ and k of the Weibull distribution as random variables.
- One wants to learn the half-life of a decay process. In this case, one would treat the λ in an exponential distribution as a random variable.
- One wants to determine the bias of an unknown coin when one doesn't want to do thousands of flips.

This leads us to take a Bayesian perspective: Every measure has some uncertainty. The probability of the measured quantity is the confidence one should have in believing its value.

Bayesian parameter learning (BPL) is a remarkably powerful technique for estimating parameter values and their uncertainty from a small set of measures. There are two key ideas behind are found:

1. The quantity you want to estimate is a parameter of a known probability distribution for which you have a formula. For example,
    a. The bias of a coin would be the p in the Bernoulli distribution.
    b. A quantity measured with a noisy instrument would be the mean of a normal distribution.
2. The parameter is a continuous random variable.

These can be found in in (Fenton & Neil, 2019) and (Kruschke, 2014).

The formula used in the first assumption gives P(measurent|parameters) for each measurement. What you want is P(parameter|measurents). This article describes how to do this with repeated application of Bayes theorem with each measurement. It has three parts:

1. An overview of random variables
2. The toolkit needed to complete BPL

3. Parameter learning

As a how-to paper, it contains discussions of how to implement the techniques with examples in python, with occasional code snippets. I assume the reader is familiar with elementary probability theory, including random variables, probability distribution functions, cumulative distribution functions, conditional and joint probability, and Bayes theorem. That said, I will provide a summary review of the concepts as they are used.

In this article, I used discretization for the calculations. It is particularly well-suited for the problems mentioned above. In each case, there are only one or two parameters. Higher-dimensional problems may require more advanced numerical methods such as MCMC or variational.

To fully understand the techniques, I strongly recommend that the reader write programs to implement them and see if they can reproduce the examples. In Python, discretization methods for Bayesian learning can be implemented using NumPy and SciPy. The examples in this paper were built using my own PDF class module and am considering making it open-source. Anyone interested in this module can contact me on LinkedIn.

## Random variables

Random variables are used to specify uncertain quantities. Their values are not specified by a single value like regular variables. Instead, they are specified by a probability density function (PDF). Recall that a PDF is a non-negative function with total integral one.

Also, recall that a subset, E, of the support of the PDF is called an event. The probability of the random variable having a value in E is the integral of the PDF over E. That is,

$$P(E) = \int_E PDF(t)dt$$

*Equation 1*

Examples of random variables include:

- The bias towards coming up heads when flipping an unknown coin.
- The value of a physical constant when measured by an instrument through a noisy channel.
- The values resulting from a development or manufacturing process that may or may not be controllable.

# The Toolkit

This section introduces the tools needed to apply BPL to a dataset. These are useful in their own right, and anyone interested in applying probability theory should know them. There are four tools in the kit:

1. Building empirical PDFs from a set of samples
2. Generating samples from a PDF
3. Computing functions of PDFs to get a new PDF
4. Applying Bayes theorem

This section describes each of these in detail with examples and hints for implementing in python.

## Empirical PDF's

Building empirical PDFs There are only a small finite set of PDFs specified by parameters (normal, uniform, exponential, Weibull, …) and an uncountably many PDFs. Most PDFs found in nature are not parameterized. These are often called empirical, as they arise from measurements. They also arise in simulations.

Here is a straightforward numerical way to build an empirical PDF from a large data set:

1. Collect data from the population or process you are interested in. This can be done through experiments, observations, or simulations.
2. Divide the data into equally spaced bins. The number of bins will depend on the size of your data set and the level of detail you want in your PDF.
3. Calculate the frequency of each bin, which is the number of data points in the bin divided by the total number of data points.
4. Build a discrete function using bin centers as the domain, and the bin frequency as the range.
5. Use interpolation to create a continuous function from the lowest bin center to the highest bin center.
6. Numerically find the total integral of the continuous function.
7. Find the PDF by normalizing the continuous function by dividing by the integral. This ensures the total integral = 1.

These steps can be easily implemented in python using NumPy and SciPy functions.

To show how this works in practice, I applied the above algorithm to various random samples of a normal distribution with mean = 10 and standard deviation = 2 (Figure 1).

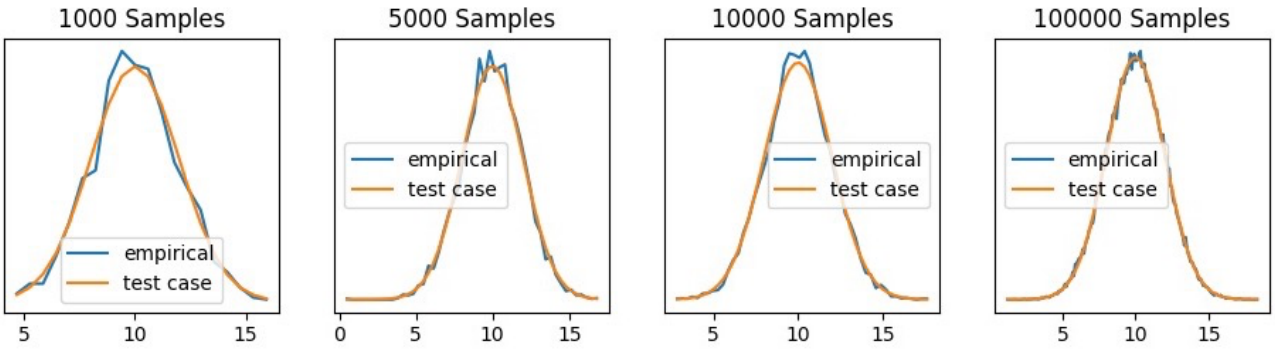


Figure 1. Examples of matching empirical PDFs to normal with mean= 10 and sd= 2

## Random Sampling

Taking random samples from A PDF Recall that a cumulative distribution function (CDF) is a function that describes the probability that a random variable will take on a value less than or equal to a given value. The CDF is defined as:

$$CDF(x) = P(X \leq x) = \int_{-\infty}^{x} pdf(t)dt$$

*Equation 2*

where X is the random variable and x is a specific value the random variable might take. The CDF is a non-decreasing function that ranges from 0 to 1, with the properties that CDF(x) = 0 for all x less than the minimum possible value of X, and CDF(x) = 1 for all x greater than or equal to the maximum possible value of X.

Here is the general procedure for generating a random sample of a random variable's PDF:

1. Compute the cumulative distribution function, the CDF. Note that since the CDF is monotonically increasing, it has a well-defined inverse,
2. Numerically compute $CDF^{-1}$.
3. Take a random sample, s, from the unit interval, [0,1].
4. Then $CDF^{-1}(s)$ is a random sample of the random variable.

Steps 1. and 2. are readily done in python using the sci-kit learn integrator and interpolation functions. For step 3, one can use python's NumPy's uniform distribution sampler.

Figure 2 shows why this works.

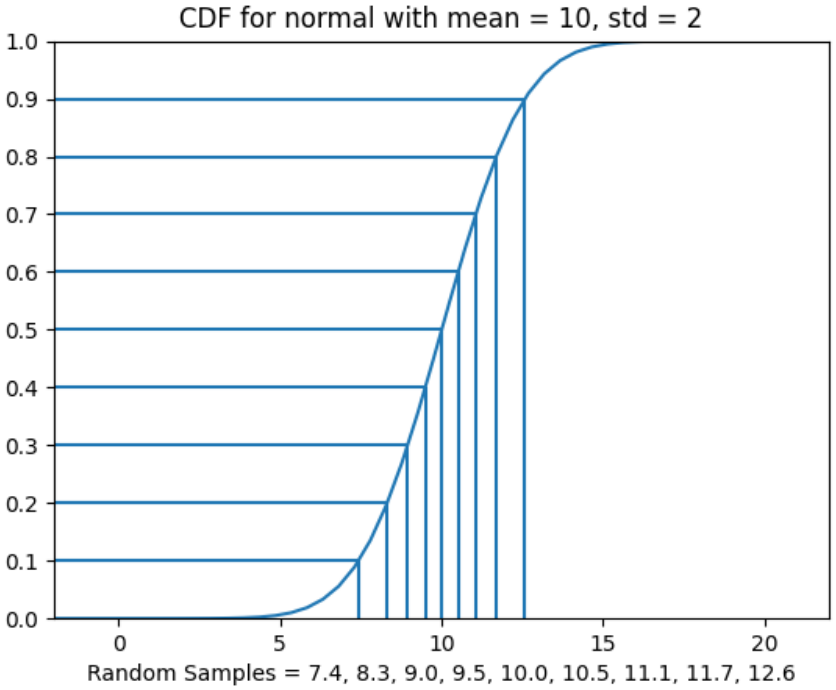


*Figure 2.*

Note that evenly distributed values of the CDF map to values bunched around the PDF mean, as one should expect.

## Functions of random variables

A function of a random variable is a mathematical function that takes the random variable as input and produces a new random variable. The function can be any function, such as a linear function, a polynomial function, an exponential function, or a trigonometric function. Functions of random variables are often used in probability and statistics to analyze and model real-world phenomena, such as multi-step processes with uncertain values.

For a simple real-world example, suppose you have to estimate the cost of a task, and you are uncertain of the number of hours and the cost per hour. Then the cost would be the quotient of these two random variables.

A practical way to compute the PDF of any function of one or more random variable is to use Monte Carlo simulation. Suppose you have a set of random variables (V1, V2, … Vs), and you want to compute F(V1, V2, … Vs).

1. Generate samples from the distribution of each random variable.
2. Evaluate the function on the set of samples using array methods.
3. Find the PDF of the array found in step 2 using the above method for finding empirical PDFs.

For example, suppose you want to estimate the cost of painting a bridge. However, you are unsure of how many painter-hours hours it will take, and you are uncertain of the cost per hour. You can use Pareto 3-point estimates (least, expected, most) to specify the uncertain values. Suppose:

- Duration 3-point estimate is (150, 200, 300) hours.
- The rate 3-point estimate is (15, 20, 35) dollars/hour.

What is needed is the product of the two estimates. To find the product, apply the Monte Carlo procedure described above.

- Convert the estimates into triangular distributions (see Figure 3).
- Take a large number, N, of random samples of each of the triangular distributions to get two arrays.
- Multiply the arrays element by element
- Find the empirical PDF of the array of products

Figure 3 shows the two factor PDFs and an approximation of their product. If the jaggies are bothersome, they can be removed with a smoother.

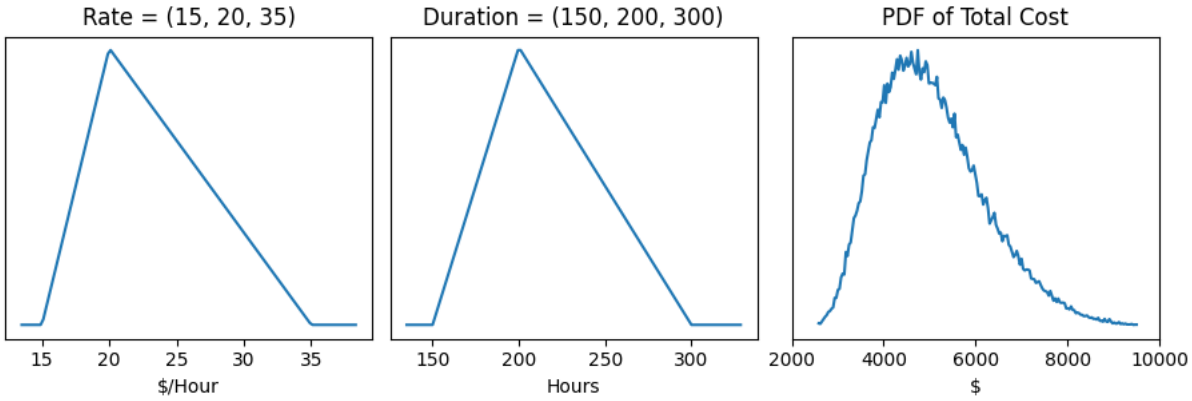


*Figure 3*

## Conditional and Joint Probability

Recall *conditional probability* is a measure of the probability of an event occurring, given that another event has already occurred. It is defined as the probability of event A occurring, given that event B has occurred and is written as P(A|B). It is central to causal analysis and machine learning. It allows us to make more informed predictions about the likelihood of future events based on past events or other information that we have.

The formula for calculating conditional probability is Equation 3

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

*Equation 3*

Where $P(A \cap B)$ is the probability of both events A and B occurring, and P(B) is the probability of event B occurring. $P(A \cap B)$ is called the *joint probability*. Often it is denoted P(A, B).

The motivation for Equation 3 can be seen in the Venn diagram (Figure 4).

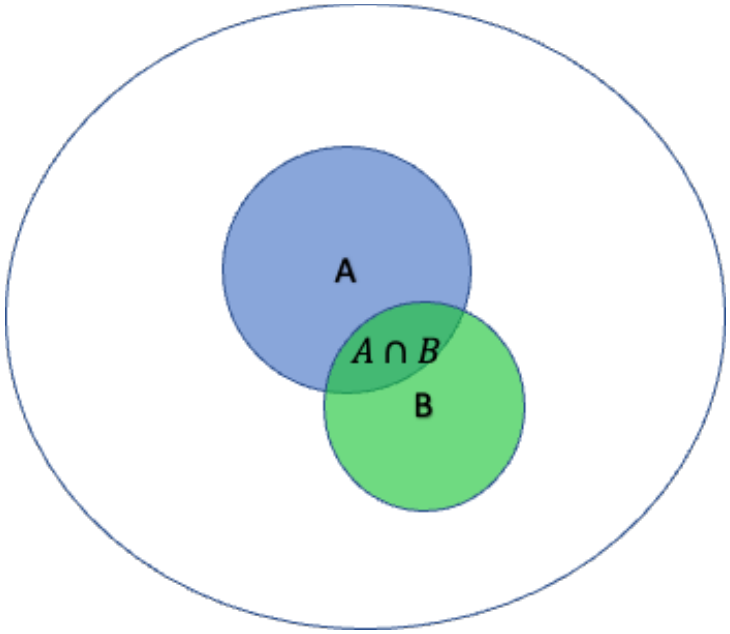


*Figure 4*

The probability of event A occurring is represented by the area of circle A, and the probability of event B occurring is represented by the area of circle B. The probability of both events happening at the same time is represented by the area of the overlap between the two circles. If we know B has occurred, then P(A|B) is seen as the quotient of the two areas.

Recall that if two random variables, X and Y, are *independent*, then

$$P(X, Y) = P(x)P(Y)$$

*Equation 4*

## Bayes Theorem

Note it is evident from Figure 4 that $P(A|B) \neq P(B|A)$, even though they are often confused.The relationship between the two is both elementary and powerful. It is called Bayes theorem. (Equation 5).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

*Equation 5*

Bayesians consider probability as a measure of belief in the likelihood of an event. They name the terms of the equation as follows:

- *P(A)* is the prior belief
- *B* is sometimes called an observation
- P(B|A) is the likelihood
- P(A|B) is the posterior
- P(B) is called the marginal

Bayesians say we update the prior to account for the observation to get the posterior.

Here is a standard discrete example:

Suppose there is a test for a disease that afflicts .02 of your risk group. Table 1 has the statistics for the test results. These are typical probabilities.

| Test Result | If Sick | If Well |
|---|---|---|
| Pos | 0.95 | 0.05 |
| Neg | 0.03 | 0.97 |

*Table 1:*

We want to know P(Sick|Test Pos), but the test gives P(Test Pos|Sick) = .95. So, we should apply Bayes theorem.

$$P(Sick|Test\ Pos) = \frac{P(Test\ Pos|Sick)P(Sick)}{P(Test\ Pos)}$$

*Equation 6*

We know P(Sick) = .02 and so P(well) = .98. To get P(Test Pos), we sum up the probabilities of how 'Test Pos' could arise.

$$\begin{aligned} P(\text{Test Pos}) &= P(\text{Test Pos})|\text{Well})P(\text{Well}) + P(\text{Test Pos}|\text{Sick})P(\text{Sick}) \\ &= (.05)(.98) + (.95)(.02) \\ &= .068 \end{aligned}$$

Substituting these values into Equation 6 gives us P(Sick|Test Pos) ≅ .28. Having only 28% confidence that you are sick leads to very different actions than believing you are 95% likely to be sick.

Many find this calculation surprising, but the confusion of P(A|B) with P(B|A) is common and has serious consequences. One example is the 'prosecutor's fallacy': a logical error that occurs in criminal trials when a prosecutor argues that the probability of a defendant being innocent, given the evidence, is very low. This is problematic because it confuses the probability of the evidence given the defendant's guilt (which is typically high) with the probability of the defendant's guilt given the evidence (which is typically lower). This has led to many false convictions. Another example is the overestimate of the number of Covid infections given the results of the home test kits.

### Marginal Probability

We will need one more theorem.

*Given two random variables, X and Y and the pdf, f(x,y), for the joint probability P(X,Y), then the pdf, $f_x$, for X is*

$$f_x(p) = \int_{support(y)} f(x,y)dy$$

*Equation 7*

Simply, one can find the individual pdf of X from the joint pdf, by taking the partial integral over the Y support. This is often called *marginalizing.*

## Bayesian Parameter Learning

This part is intended to be a gentle introduction to the topic. There are two examples.

1. **The one-parameter case** – The classical coin bias problem. This is both elementary and amusing.
2. **The two-parameter case** - Estimating the measured quantity value with a noisy measurement process.

The key idea is to treat the parameters as random variables and learn their distributions using successive applications Bayes theorem (Equation 8).

$$P(parameters|data) = \frac{P(data|paramters)P(parameters)}{P(data)}$$

*Equation 8*

The choice of the likelihood function is based on what is being modeled. For example, population measurements are usually normally distributed, while one would use Weibull for time-to-failure analysis.

## One Parameter

Examples of one-parameter distributions in the Bernoulli, the binomial, and the exponential. Let's consider the coin bias problem and the Bernoulli distribution. It is both instructive and amusing:

Imagine you have a coin that will be used in a betting game. You will agree to the game if you are sufficiently confident the coin is fair. However, you do not have time to make thousands of flips. Somehow, you have access to a BPL program for learning the bias. You decide to take the bet if you are 75% confident that the bias is between .45 and .55,

To apply BPL to this problem, we note that the bias towards heads is the p in a Bernoulli distribution (Equation 9).

$$P(heads|p) = p$$
$$P(tails|p) = 1 - p$$

*Equation 9*

The bias p is a value in the unit interval. If p =1, all the flips are heads. If p= 0, all the flips are tails. If p = .5, heads and tails are equally likely.

Normally, p is considered to be a scaler. To apply PBL, we treat the bias as a random variable. Also note that while coin flipping is a discrete process, the bias is a continuous random variable. The PDF of p is a function over the interval [0,1].

Let's try this out. Figure 5 shows the graphs of the sequence of the learned PDF using a set of observations (Heads, Heads, Tails, Tails, Tails, Tails, Heads, Tails) and an initial uniform prior.

Here are the steps for the first observation:

- Choose the initial prior pdf for the parameter based on your contextual information.

  For example, if the coin is known to be from the US Mint, it is likely to be fair, and so you might choose the prior to be a normal distribution with a .5 mean and a small standard deviation, consistent with the mint manufacturing specifications. The implication of this choice is that you believe the coin is fair, and it will take substantial evidence to convince you otherwise.

On the other hand, if you know nothing about the coin, the safest prior is that all values of p are equally likely. Since the support of the parameter is the closed interval,[0, 1], we would set Prior(p) = 1 for all p in the support. . This is called a 'uniform prior.' Using a uniform prior is sometimes called 'the principle of indifference'. It is generally the best choice to avoid your own prejudices.

- Discretize the support of the prior, the unit interval, into *equal* intervals, ($p_1$, …,$p_n$). In most cases, n can be 100.
- Build an array L= ($l_1$,…,$l_n$) by applying the numerator of equation 6 for each $p_i$ and equation 7 for the likelihood of the observation.

  For example, if we have a prior uniform, n =100, and the observation is Heads. Then $p_i$ = i/100, $P(p_i) = .01$ for each I, $P(Heads|p_i) = p_i$ , and so $l_i = p_i/100$.

- Calculate the posterior of equation 6 by interpolating L into a function and normalizing it so its integral is one area over the domain. See Figure 5 below. This normalization step is essential as accounts for the dominators in the Bayes quotient.

For the following observations, use the above steps and the posterior of the previous calculation as the prior for the next. This technique is called Bayesian refinement.

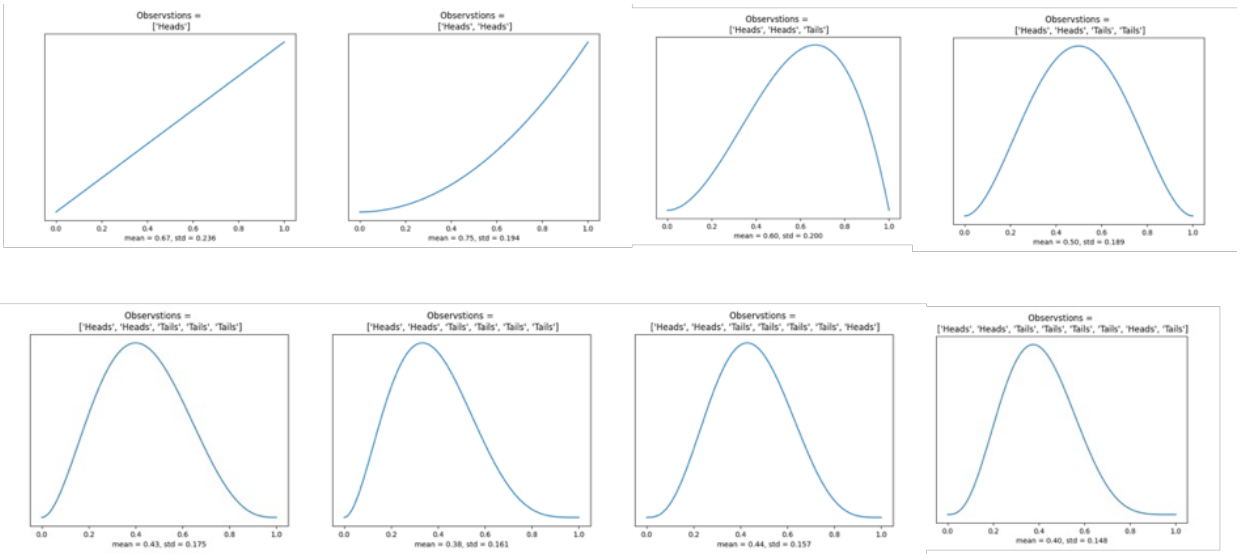


*Figure 5*

Note that for the first two runs, the observations contain no tails. So, the bias PDF is monotonically increasing, and there is some likelihood that the bias is near 1. However, as the graph for the third run shows, when there is a tail, the algorithm shows zero probability that the bias is 1.

Recall that the criterion for the coin being fair is that we P(E|p) ≥ .75, with E being the interval [.45, .55]. This is visually displayed in Figure 6 below. The examples show the PDFs for 100, 200, 300, and 400 flips with 53% heads. The green areas are the regions above the fair interval. Note, as expected, the PDF's narrow and the fair region covers an increasing percentage of the area under the curve.

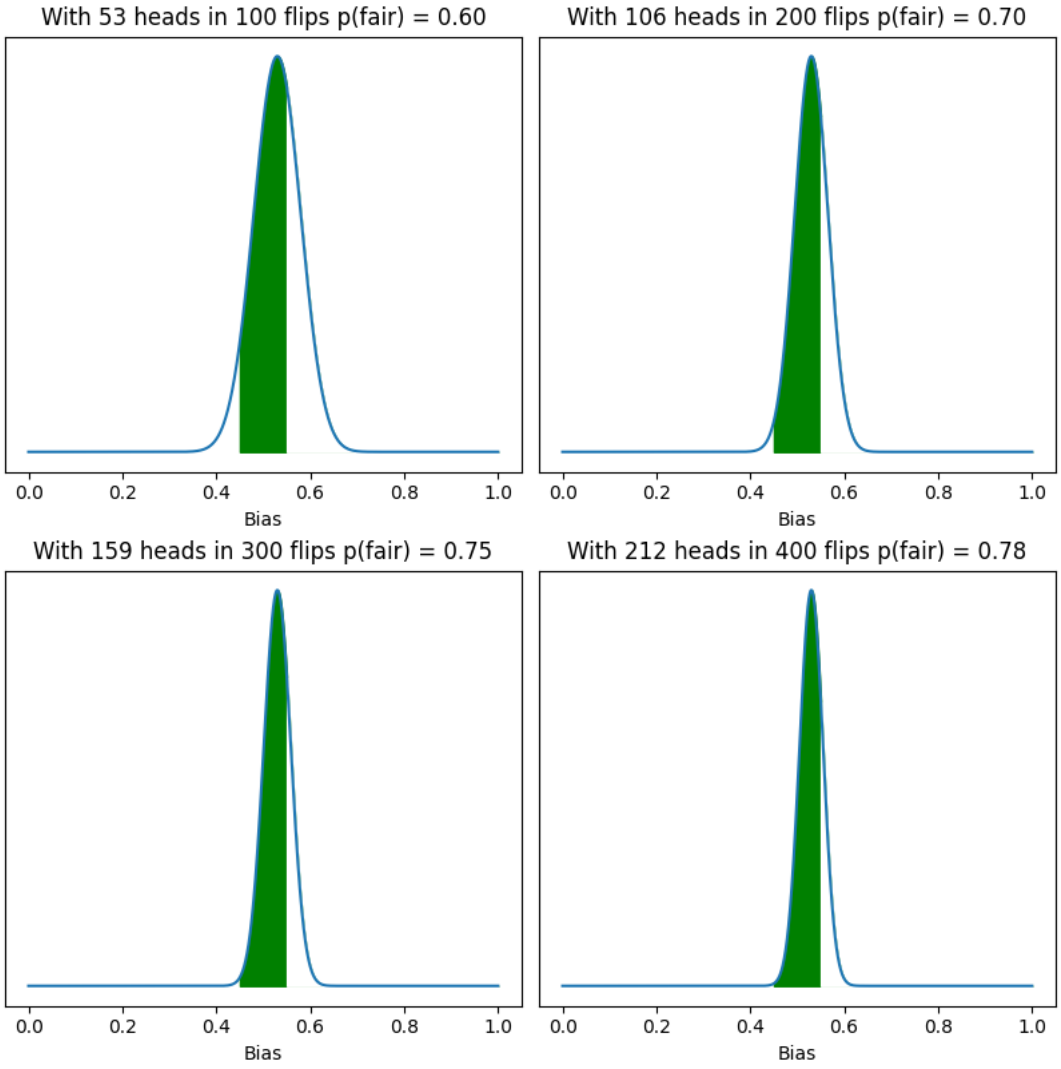


Figure 6

You can decide the coin is fair enough with 300 flips and you can be more confident at 400 flips.

To further test the algorithm, let's consider some edge cases. As shown in Figure 7:

- When there are zero heads in 20 flips, the curve is shifted to the left with a peak at 0.
- When there are 0 heads in 200 flips, it is virtually certain that the bias is 0.

- When there is 1 head in 20 flips, the curve looks very different. It is still shifted to the left but shows 0 probability at b = 0 since there was a head.

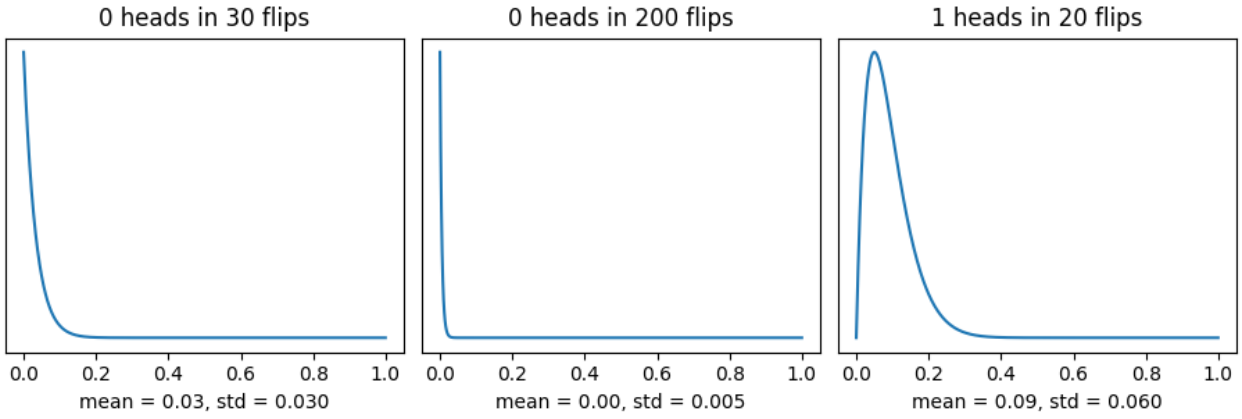


*Figure 7*

## Two Parameters

This section describes how to extend the above 1-parameter method to estimate both parameters of a 2-parameter distribution from a small set of observations. This technique will work for any two-parameter probability distribution function. These include normal, log-normal. Weibull, and beta distributions. Note that in each of these have independent parameters. So, to learn the parameters, we can apply Equation 4 and Equation 8 to get Equation 10.

$$P(p_1, p_2|Obs) = \frac{P(Obs|p_1, p_2)P(p_1)\mathrm{P}(p_2)}{P(Obs)}$$

*Equation 10*

We will elaborate the digitized approach as the successive refinement of the 1-parameter case . In that case, we built an array of likelihood values for the parameter of the local of the parameter using Bayes theorem. For the 2-parameter version, digitize each parameter and build a 2-dimensional array of likelihoods using Equation 10. Then to get the likelihood for and then apply by taking the margins of the array to get discrete arrays of the parameter pdfs. Then interpolate and normalize them to get the

Assume we have:

- A 2-parameter pdf, P($p_1$, $p_2$, x)
- Assume a finite set of observations, Obs = {O1, …, On}.

Like the 1-dimensional case, we initialize the process with the first observation.

- Choose the initial priors for the parameters, $p_1$ and $p_2$, based on your contextual information.

  For example, suppose we want to learn the mean and standard deviation from some a set of samples using a normal likelihood function. A good choice of prior of the mean depends on the nature of the subject matter beliefs. Two reasonable choices be a uniform distribution or a triangular distribution with the low being the lowest possible observation, the high being the highest, and the mode being a likely middle value. A prior for the standard deviation could be a uniform distribution with low equal 0, and high being the range of the support of mean prior.

$$prior_1(x) = triangular(low, expected, high, x)$$

$$prior_2(x) = uniform(0, high - low, x)$$

*Equation 11*

- Discretize the support of the priors, $prior_1$ and $prior_2$, into *equal* intervals, $(p_{1,1}, \ldots, p_{1,n})$ and $(p_{2,1}, \ldots, p_{2,n})$.In most cases, n can be 100.
- Build a 2d array $L= (L_{i,j})$ by applying the numerator of Equation 10 to pair $(p_{1,i}, p_{2,j})$ and the observation

  For example, if we have a normal model, the likelihood function is

$$L_{i,j} = \frac{1}{p_{2,j}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{obs - p_{1,i}}{p_{2,j}}\right)^2} prior_1(p_{1,i}) prior_2(p_{2,j})$$

- Apply a discrete version of Equation 4, by summing the rows and columns to a get a discretized version of the likelihoods of the parameters.

$$parm_{1,i} = \sum_j L_{i,j}\,, \qquad parm_{2,j} = \sum_i L_{i,j}$$

- To get the posterior pdfs of the parameters interpolate *$parm_1$ and $parm_2$* arrays into functions and normalize them so that their integrals have unit area over their domain. See Figure 5 below.

For the following observations, proceed with Bayesian refinement using the above steps with the posteriors of the previous calculation as the priors for the next.

Just knowing the parameters' pdf can be useful. For example, suppose we want to measure a constant *c* in a noise environment. If the noise is Gaussian, then the process is normal with mean *c*. Then with experimental measures of *c*, the pdf of the learned mean gives us the probability estimate of the constant. We may not care about measuring the noise. If we need to measure the noise, we can use the standard deviation distribution.

Now that we have the pdfs for the parameters, there is one more step to get to learned pdf of the process that generated the observations. The trick here is to do a Monte Carlo simulation

using the parameters' pdfs to get an array of samples of learned pdf using the sample generation described above and then generate the final pdf by using the empirical pdf (also described above) from the array of samples. Specifically, if we have a model 2-parameter model, pdf, M(parm1, parm2)

- Make an array, p1, of 10,000 samples of the parm1 pdf.
- Make an array, p2, of 10,000 samples of the parm2 pdf.
- For n = 1, 10,000, take a sample of the pdf $M(p1_n, p2_n)$ to get an array of samples of the pdf to be learned.
- Normalize the histogram of the above array and interpolate the learned pdf.

## Two Examples

### 1. Gaussian Process

For the first, assume the observations come from a Gaussian process with a normal distribution. The subject matter experts have picked a triangular prior with parameters (1, 4, 15). See Equation 8. With that choice the prior sigma is chosen to be uniform from .1 to 14, the span of the mu priors (Equation 9)

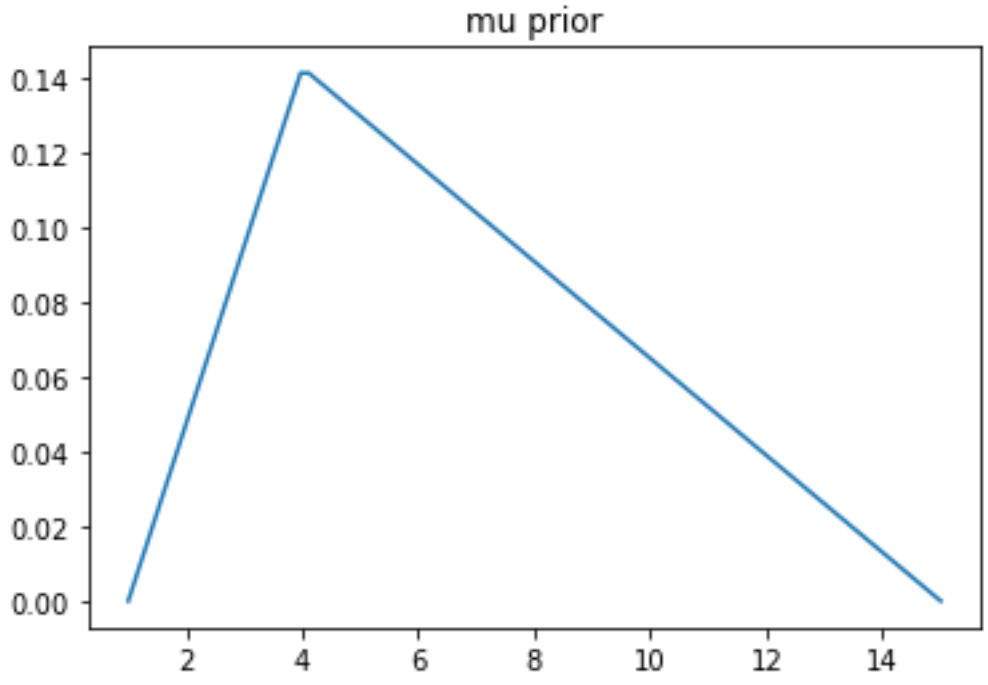


*Figure 8*

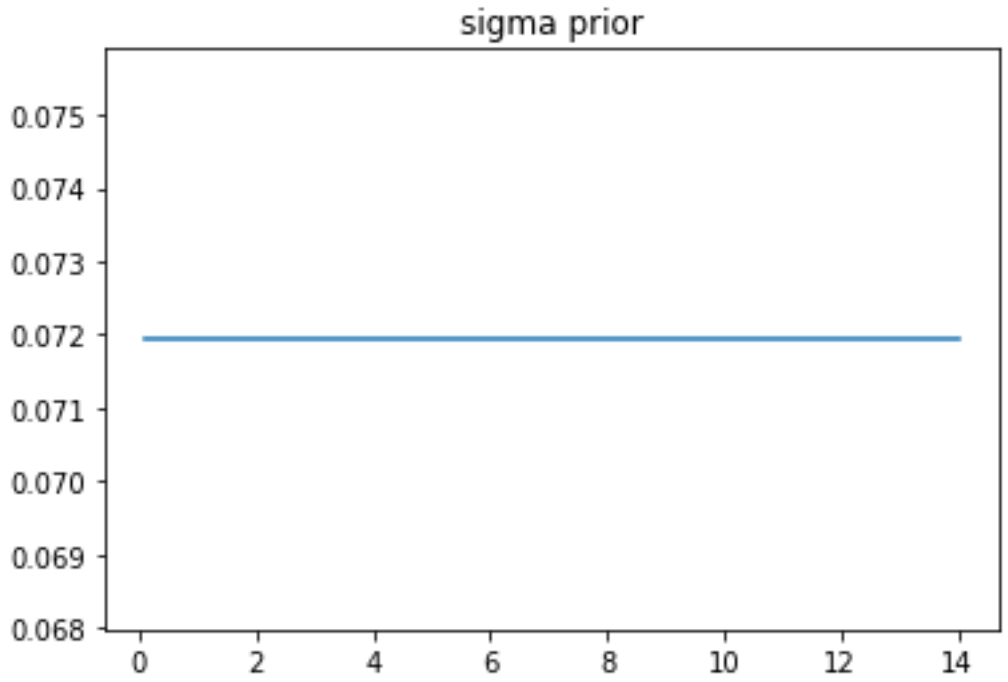


*Figure 9*

To test the algorithm, I generated samples from a normal distribution with mu = 10 and sigma = 3. Note that these choices are within range of the priors, but significantly different. For example, the mean of the mu prior is 6.6666…. The following figures show the results of the algorithm for

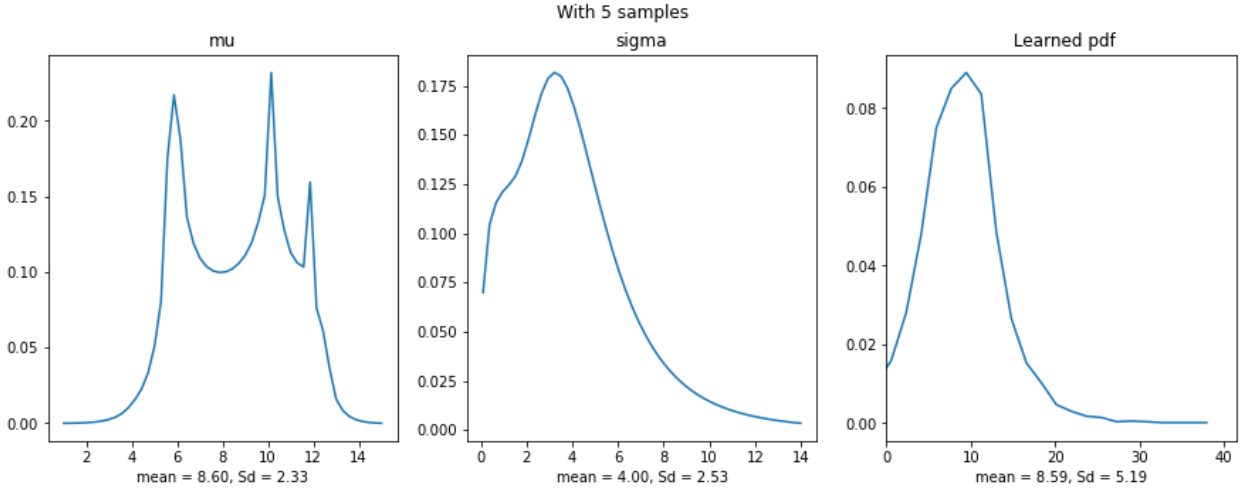


*Figure 10*

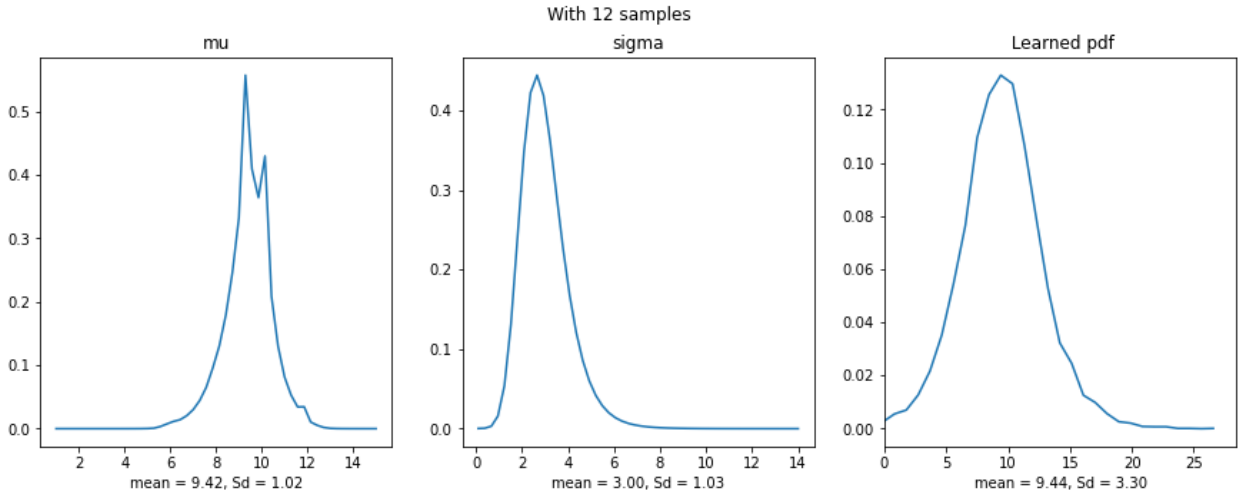


*Figure 11*

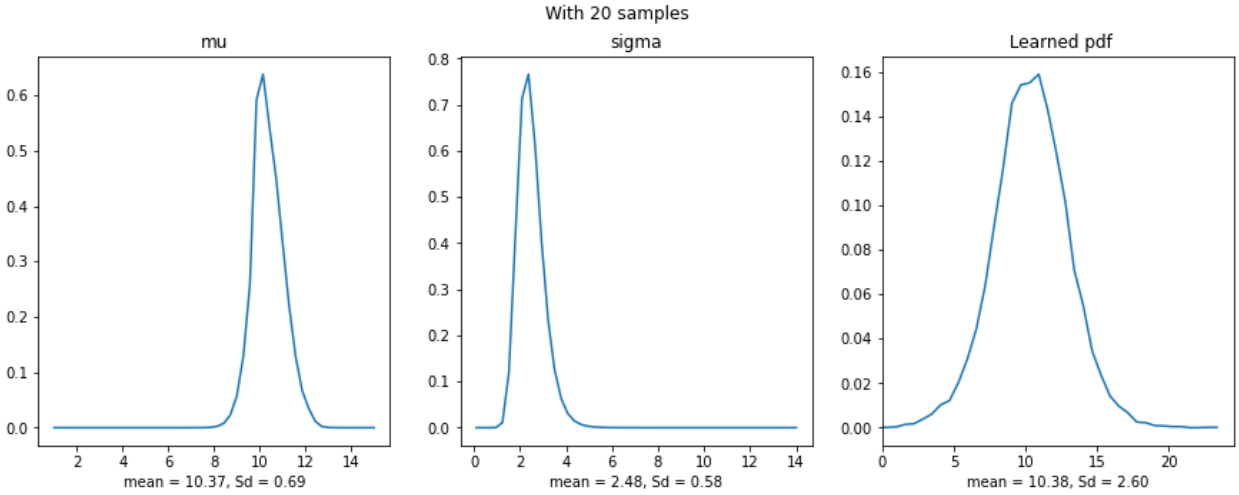


*Figure 12*

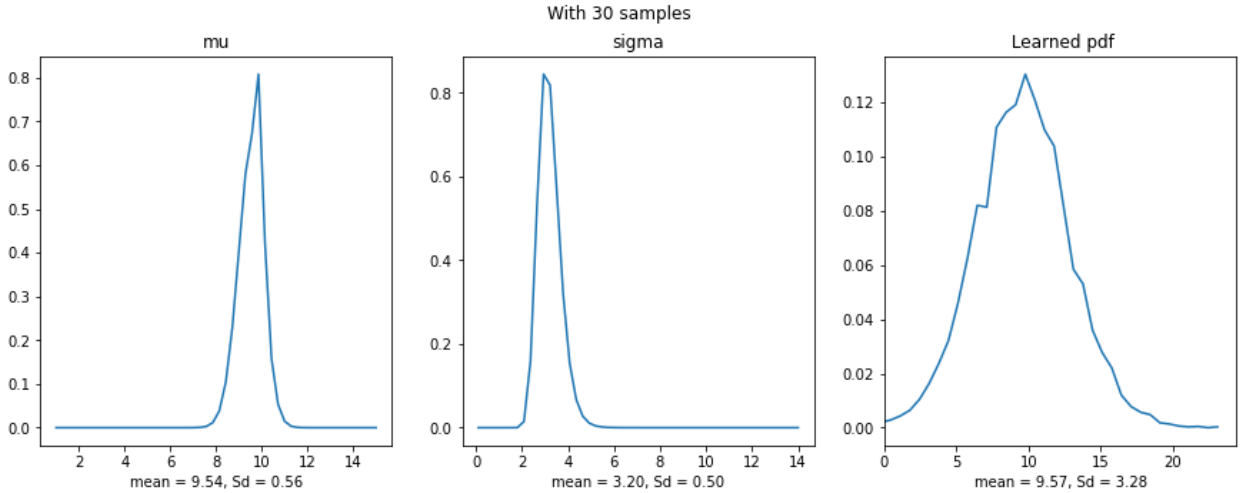


*Figure 13*

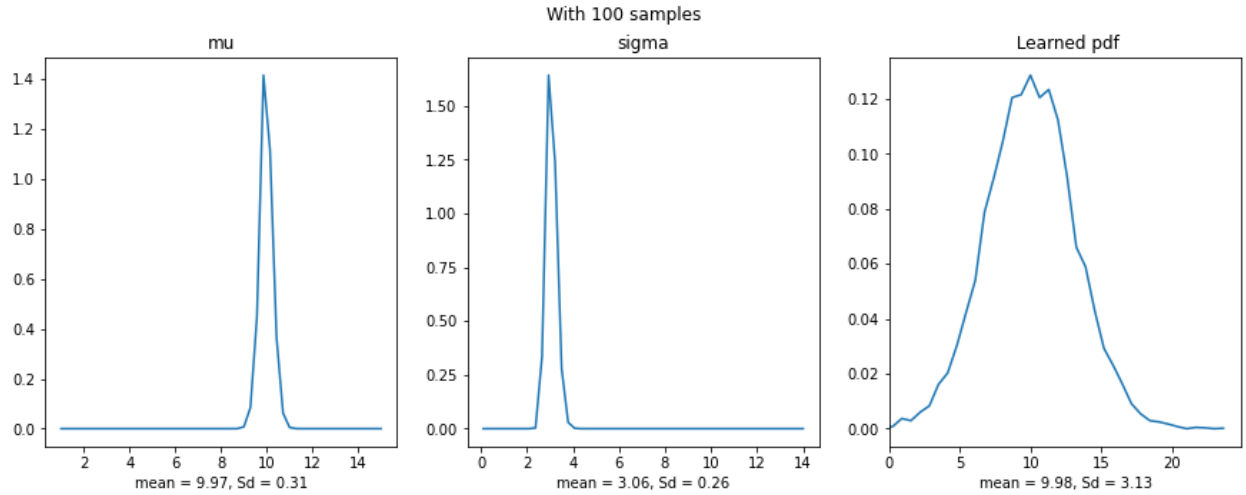


*Figure 14*

As one might expect, the distributions narrow around the test-case parameters. Also, with as few as 7 random samples, the method provides an inkling of the parameter values. With only 100 samples, there is little uncertainty in the parameter values.

### Example2. Lognormal Process

For a second example, let's try to learn the pdfs of the mu and sigma of a lognormal distribution. While normal distributions model additive processes, lognormal model multiplicative processes. In this case, mu is not the mean, but a scale parameter, and sigma is the shape parameter. See this Wikipedia article for more explanation.

In this example, we will use uniform priors for mu, (0, 2), and sigma, (.1, 5). See Figure 15 and Figure 16.

The observations were generated by sampling a lognormal pdf with mu = 1 and sigma = 0.3.

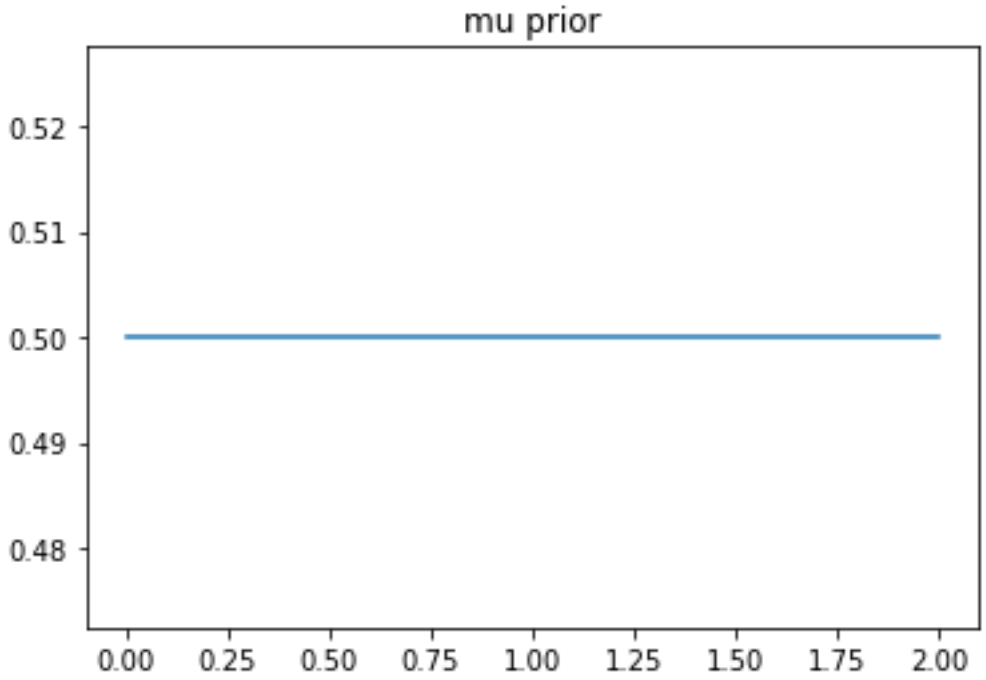


*Figure 15*

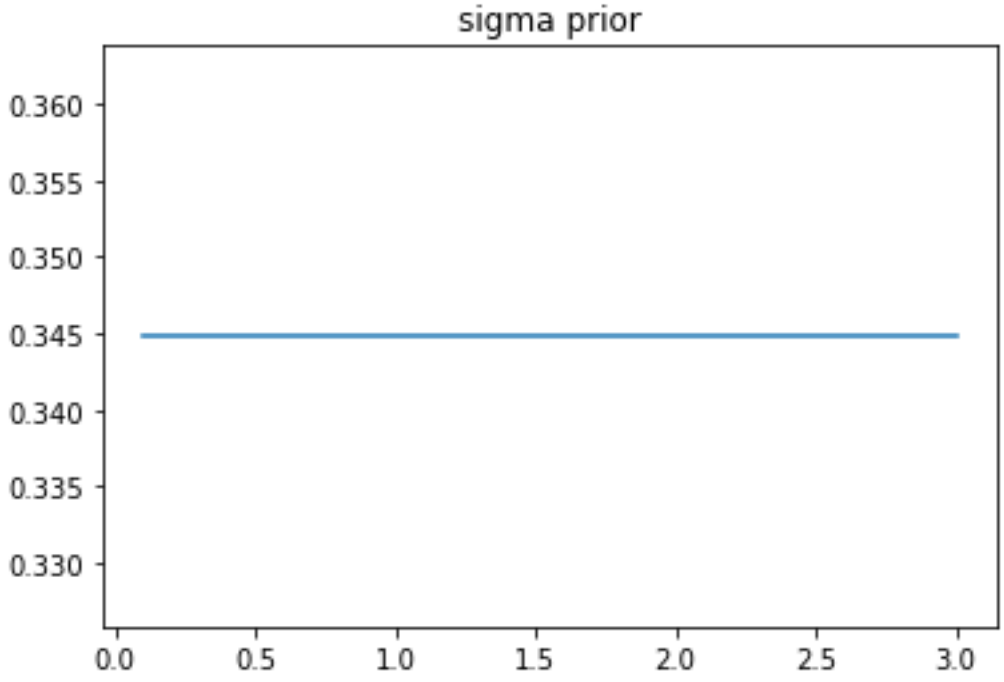


*Figure 16*

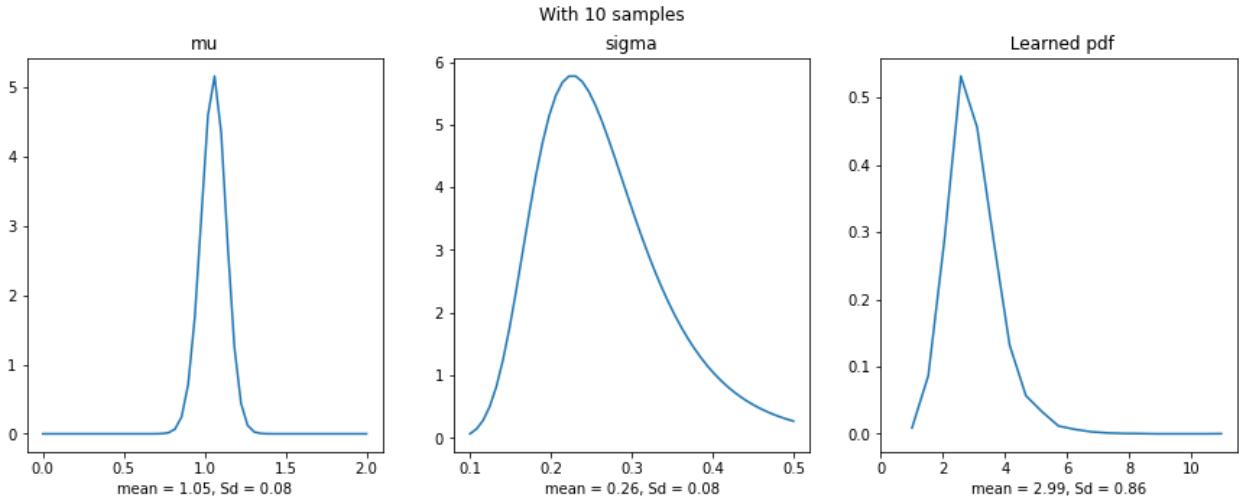


*Figure 17*

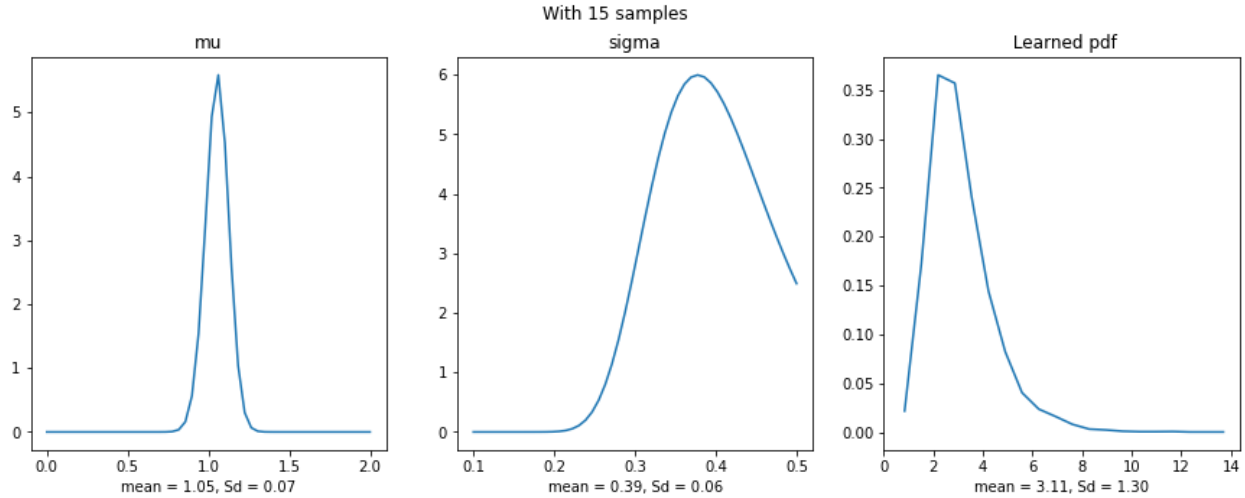


*Figure 18*

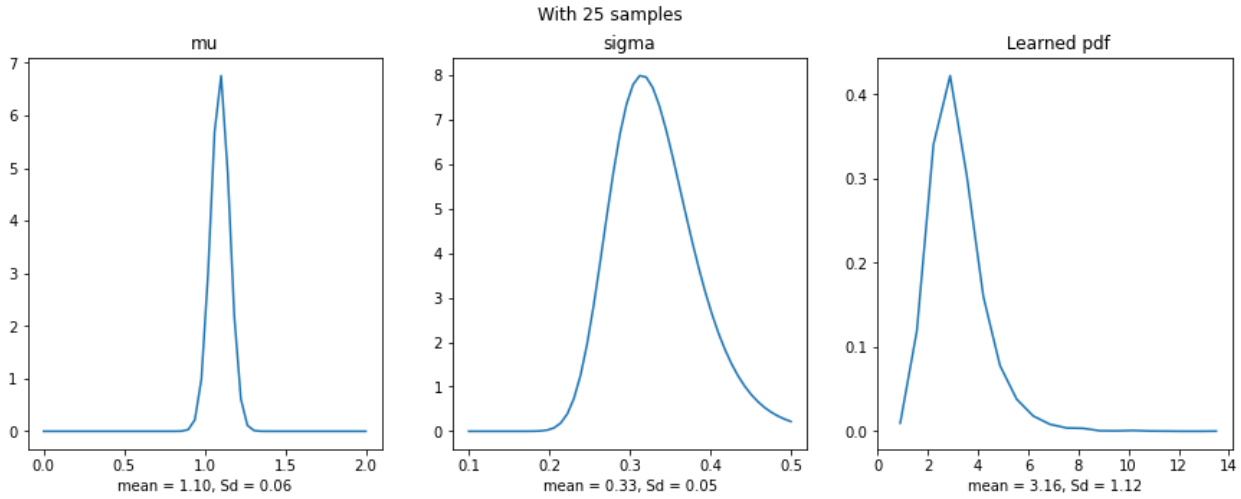


*Figure 19*

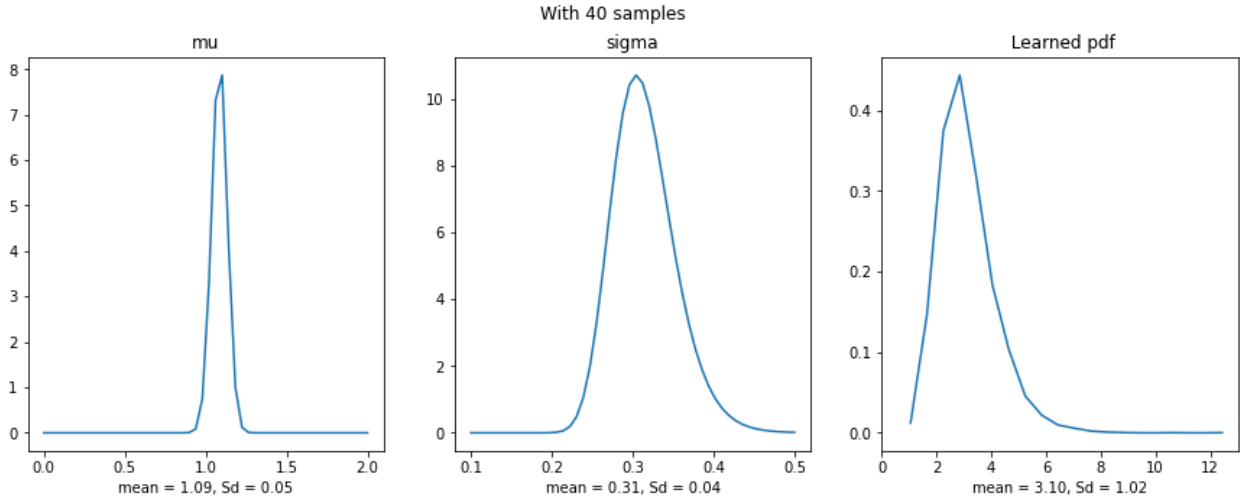


*Figure 20*

## Acknowledgments

The author is grateful to the reviewers of the early drafts of this article. Their comments and suggestions were essential in. making the paper clear and concise. The reviewers include Judi Taylor Cantor, Jim Cantor Ph.D., Quinn Jackson CSci, Michael D'Eath, and Pujeethaa Jakka.

# Bibliography

Fenton, N., & Neil, M. (2019). *Risk Assessment and Decision Analysis with Bayesian Networks, Second Edition.* Boca Raton, Fl.: CRC Press.

Kruschke, J. (2014). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan 2nd Edition.* Academic Press;.