Monte Carlo Integration

Jonathan Navarrete

Introduction

In addition to being able to generate random variables, we should also be able to apply these simulation methods to solve more complex problems. We'll begin with Monte Carlo Integration methods.

Topics to be covered:

1. Classic Monte Carlo Integration
2. Importance Sampling

Monte Carlo Integration

Given a function $h(x)$ for which we wish to integrate over region $[a, b]$, $\int_a^b h(x)dx$, we can solve this numerical problem by treating it a as a random variable we are trying to find the expectation for. Treat X as a random variable with density $f(X = x)$, then the mathematical expecation of the random variable $h(X = x)$ is

$$E[h(X)] = \int_{\mathcal{X}} h(x)f(x)dx = \theta$$

If a random sample $X_1, \ldots, X_n$ is generated from $f(x)$, an unbiased estimator of $E[h(X)]$ is the sample mean. Review: Sample mean (http://www.math.uah.edu/stat/sample/Mean.html)

$$\bar{h}_n = \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

Monte Carlo Integration

Suppose we have a function $h(x) = 3x^2$ for which we wish to integrate over the interval $[0, 2]$.

- Can apply deterministic numerical approximation methods (see R's `integrate`)
- or we could treat x as a random variable, $X = x$, from a $Unif(0, 2)$ whose pdf is simply $f(x) = \frac{1}{2-0} = \frac{1}{2}$.

If we now generate some n random values from $f(x)$ and evaluate them at $h(x)$, then take the mean, we'd be calculating the expected value of $h(x)$,

$$\begin{aligned}
\theta &= \int_0^2 h(x)dx \\
&= (\frac{2-0}{2-0}) \times \int_0^2 h(x)dx = 2 \times \int_0^2 h(x)\frac{1}{2}dx \\
&= 2 \times E[h(X)] = 2 \times \int_{-\infty}^{\infty} h(x)f(x)dx \\
&\approx 2 \times \frac{1}{n}\sum_{i=1}^{n} h(x_i) \\
&= \hat{\theta} \approx \theta
\end{aligned}$$

Monte Carlo Integration

If we now simulate this, we will see we approximate the true solution $\int_0^2 h(x)dx = 8$

```
n = 50000 ## sample size
h <- function(x) { 3*x^2 } ## function of interest, h(x)
X <- runif(n = n, min = 0, max = 2)  ## samples from f(x)
v = 2 * mean(h(X)) ## 2 * E[h(x)]
print(v) ## approximately 8
```

```
## [1] 8.00568
```

```
integrate(f = h, lower = 0, upper = 2)
```

```
## 8 with absolute error < 8.9e-14
```

Monte Carlo Integration

Now, to generalize the method used. Given a function $h(x)$ whose integral is well defined, where we wish to evaluate at interval a to b. Then

$$\begin{aligned}\theta &= \int_a^b h(x)dx \\ &= (b-a)\int_a^b h(x)\frac{1}{b-a}dx \\ &= (b-a)\int_a^b h(x)f(x)dx\end{aligned}$$

where $f(x) = \frac{1}{b-a}$ is $Unif(a,b)$, and $x \sim Unif(a,b)$.

The algorithm to calculate $\hat{\theta}$ is as follows:

1. Find a density $f(x)$ from which we can sample x
2. Generate $x_1, \ldots, x_n \sim f(x)$
3. Compute $(b-a) \times \bar{g}_n$, where $\bar{g}_n = \frac{1}{n}\sum_{i=1}^n h(x_i)$

Monte Carlo Integration, Variance Estimation

Now that we can calculate an estimate of statistic θ, $\hat{\theta}$, we should also be able to calculate the standard error in order to build confidence intervals (CIs).

The variance can be written as

$$Var(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \theta)^2$$

We will use this form to calculate the variance of $\hat{\theta}$

$$\begin{aligned} & Var(h(x_i)) \\ & = \frac{1}{n}\sum_{i=1}^{n}(g(x_i) - \hat{\theta})^2 \\ & = \sigma^2 \end{aligned}$$

And

$$\frac{\sigma^2}{n} = \frac{1}{n^2}\sum_{i=1}^{n}(h(x_i) - \hat{\theta})^2$$

Monte Carlo Integration, Variance Estimation

So, the standard error estimate is

$$\frac{\sigma}{\sqrt{n}} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(h(x_i) - \hat{\theta})^2}$$

which we can then use to construct 95% confidence intervals

$$\hat{\theta} \pm 1.96 \times \frac{\sigma}{\sqrt{n}}$$

Monte Carlo Integration, Variance Estimation

We can now calculate the standard error for the former example.

```
N = 10000 ## sample size
h <- function(x) { 3*x^2 } ## function of interest, h(x)
X <- runif(n = N, min = 0, max = 2)  ## samples from f(x)
h_values <- 2 * h(X)

cumMean <- function(x, n){
  num = cumsum(x)  ## numerator
  denom = 1:n  ## denominator
  result = num/denom
  return(result)
}
```

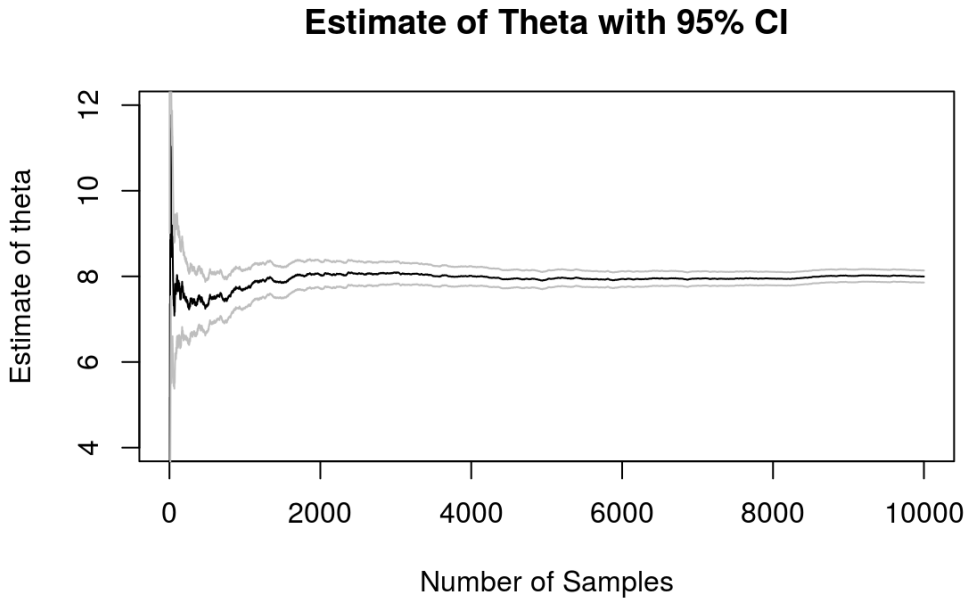
Monte Carlo Integration, Variance Estimation

```
cumSE <- function(x, n){
  m = mean(x)
  num = sqrt(cumsum((x - m)^2)) ## numerator
  denom = 1:n ## denominator
  result = num/denom ## cummulative mean of (x_i - theta)^2
  return(result)
}

thetas = cumMean(h_values, N)
SE = cumSE(h_values, N)
```

Monte Carlo Integration, Variance Estimation

```
plot(x = 1:N, y = thetas, type = "l", ylim = c(4, 12), xlab = "Number of Samples",
     ylab = "Estimate of theta",
     main = "Estimate of Theta with 95% CI")
lines(x = 1:N, y = thetas + 1.96*SE, col = "gray") ## CI
lines(x = 1:N, y = thetas - 1.96*SE, col = "gray") ## CI
```



The plot shows how our parameter estimate (and CI) converges closer to the true value as N (sample size) increases. If we were to have had a small sample size, say, 100, our parameter esimate would have performed poorly. But at $N = 10,000$, our estimate $\hat{\theta}$ performs well.

Monte Carlo Integration, Variance Estimation

```
## final estimate
thetaHat = mean(h_values)
se <- sd(x = h_values)/sqrt(N)
ci <- thetaHat + 1.96*c(-1,1) * se

print(thetaHat) ## theta estimate
```

```
## [1] 7.99777
```

```
print(ci) ## 95% CI
```

```
## [1] 7.857001 8.138538
```

Strong Law of Large Numbers

Monte Carlo Integration relies on the Strong Law of Large Numbers (SLLN) to approximate $E_f[h(X)] = \int_{\mathcal{X}} h(x)f(x)dx$. The method is to generate a sample $\mathbf{X} = (x_1, \ldots, x_n)$ from pdf $f(x)$ and use the empirical average of $h(x)$ as an approximation. This Monte Carlo estimate *almost surely* converges to $E_f[h(X)]$ by the SLLN.

For a given random sample $X_1, \ldots, X_n$ the empirical mean, $\bar{X}$, can be used as an estimator for $E[x] = \mu$, and given a *sufficiently* large sample, this estimate approximates the true expectation μ almost certainly with a probability value of 1. Or more formally,

$$P(\lim_{n\to\infty} \bar{X} \to E[X]) = 1$$

See reference: Strong Law of Large Numbers (http://www.math.uah.edu/stat/sample/LLN.html)

Central Limit Theorem

The CLT implies that for our Monte Carlo estimator $\hat{\theta}$,

$$\frac{\hat{\theta} - E[\hat{\theta}]}{\sqrt{Var(\hat{\theta})}} \sim N(0,1)$$

and by the Strong Law of Large Numbers this happens almost surely as $n \to \infty$. See: Central Limit Theorem (http://www.math.uah.edu/stat/sample/CLT.html)

$$\hat{\theta} \sim N(\theta, \sigma^2)$$

The Weak Law of Large Numbers then states that for any margine of error ϵ, the average almost surely do not deviate further than ϵ away from the true θ,

$$\lim_{n \to \infty} P(|\hat{\theta} - \theta| > \epsilon) = 0$$

The SLLN speaks to convergence to the expected value, where the WLLN speaks to convergence in probability.

Normal p-value estimate for unbounded region

We will next explore how to generate $N(\mu, \sigma)$ p-values using a $Unif(0, x)$ distribution.

$$\Phi(x) = P(X \leq x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} \, e^{(t-\mu)^2/2\sigma} dx$$

Now, suppose that $\mu = 0$ and $\sigma = 1$, such that we are only dealing with the standard Normal CDF. We are interested in calculating a Monte Carlo estimate for $\Phi(x)$

$$\Phi(x) = P(X \leq x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \, e^{t^2/2} dx$$

Normal p-value estimate for unbounded region

Our first problem is dealing with bounds $(-\infty, x)$. We cannot direcly apply our MC integration algorithm directly because the integral range $(-\infty, x)$

A solution around this issue is breaking up this problem into two sections exploiting the Normal distribution's symmetry around μ.

The two sections will be $(-\infty, 0)$ and $(0, x)$.

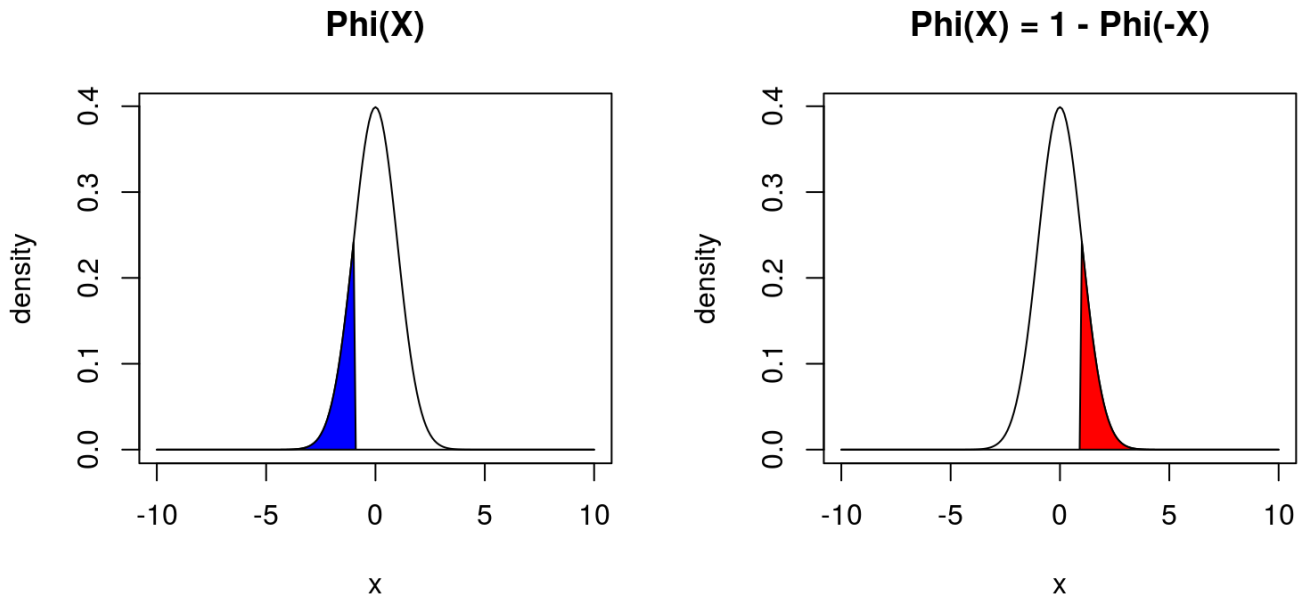
With the given symmetry property, we know that any random variable from $N(\mu, \sigma)$ will have an equal probability of being on the positive end or negative. Thus, let's set $P(X \leq 0) = 1/2$. Now, all we need to concern ourselves with is the region $(0, x)$.

Now, assuming $x > 0$, then we use the Monte Carlo estimator as is

$$\begin{aligned}\Phi(x) &= P(X \leq x) \\ &= P(X \leq 0) + \int_0^x \frac{1}{\sqrt{2\pi}} e^{t^2/2} dx \\ &= \frac{1}{2} + \int_0^x \frac{1}{\sqrt{2\pi}} e^{t^2/2} dx\end{aligned}$$

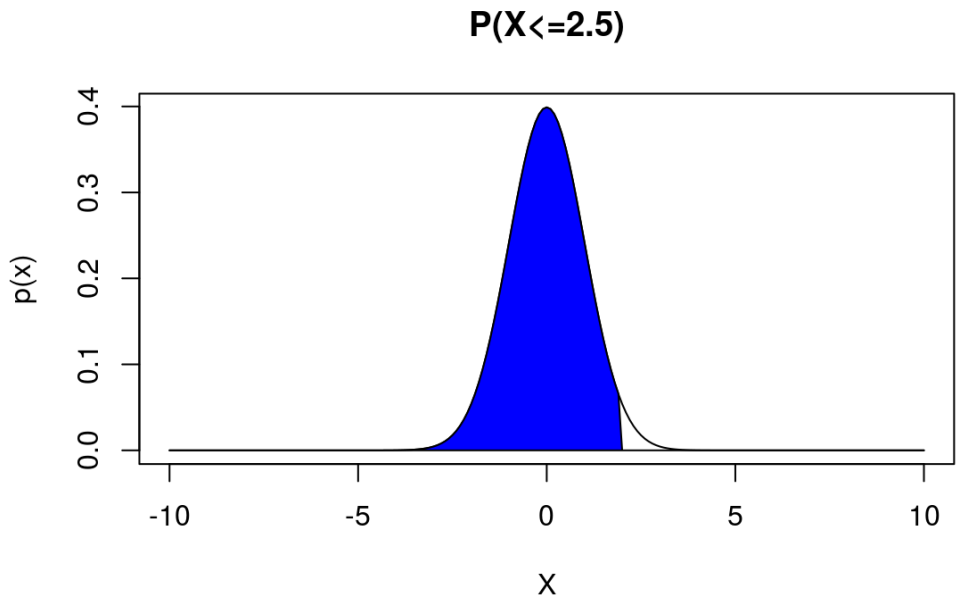
Normal p-value estimate for unbounded region

However, if $x < 0$ such as $x = -1$, then we would use the symmetry property of the Normal distribution to calculate $\Phi(-1)$ by $1 - \Phi(-(-1)) = 1 - \Phi(1)$



Normal p-value estimate for unbounded region

For example, if $x = 2.5$, then we would calculate a MC estimate for $\Phi(X \leq 2)$.



Normal p-value estimate for unbounded region

Suppose we'd like to calculate $\Phi(2.5) = P(X \leq 2.5)$

```
N = 10000
x = 2.5
t = runif(N, 0, 2.5)
p = 0.5 + 1/sqrt(2*pi) * mean(x * exp(-t^2 / 2))

print(p)
```

```
## [1] 1.000627
```

```
print(x = pnorm(q = 2.5, lower.tail = TRUE))
```

```
## [1] 0.9937903
```

Normal p-value estimate for unbounded region

In the above example, we simulated from a $Unif(0, 2.5)$ distribution. But if we'd like to maintain a general algorithm from which we are interested in generating all samples from $Unif(0, 1)$ distribution, then we'd make a slight variable change. In order to calculate $P(X \leq x) = \theta$,

$$\begin{aligned}\theta &= \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \\ &= \frac{1}{2} + \int_{0}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \times \frac{x}{x} \times dt \\ &= \frac{1}{2} + \int_{0}^{1} x \frac{1}{\sqrt{2\pi}} e^{-t^2/2} \frac{1}{x} dt\end{aligned}$$

Normal p-value estimate for unbounded region

and let $Y = t/x \rightarrow t = xY$ such that

$$
\begin{aligned}
&= \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^1 x e^{-(x \times Y)^2/2} \frac{1}{x} dy \\
&\approx \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \frac{1}{N} \sum_{i=1}^{N} x_i e^{-(x_i \times Y)^2/2} \\
&= \hat{\theta}
\end{aligned}
$$

for $x > 0$ and $x_i \sim Unif(0, 1)$.

Calculating N(0,1) tail probabilities

Calculating tail probabilities can be tricky. Suppose $Z \sim N(0,1)$, then $\Phi(x) = P(Z \geq x) = E[I_{Z \geq x}] \approx \frac{1}{N} \sum_{i=1}^{N} I_{z_i \geq x}$ where I is an indicator function returning 1 if the condition is satisfied, else 0.

```
N = 10000
x = 3.2
Z = rnorm(N)
p_val = mean(Z >= x)
print(p_val)
```

```
## [1] 9e-04
```

```
print(sum(Z > x)) ## few samples with Z > x
```

```
## [1] 9
```

Variance estimation will be problematic for rare events like these

Importance Sampling

Our Monte Carlo integration algorithm looks like

$$E[h(x)] = \int_{\mathcal{X}} h(x) f(x) dx = \theta$$

Where f is generally a uniform density. However, this can be costly if we need to calculate the tail probabilities. For example, say $Z \sim N(0, 1)$ and we wish to calculate $P(Z > 4.5)$.

In order to obtain a stable result using a uniform distribution for f we would need to simulate many, many samples. This can be time and computationally costly, and this process can be improved.

Solution: we can weight samples x by density g such that this would improve results for tail probabilities.

Importance Sampling

We can rewrite our previous result as

$$\begin{aligned} E[h(x)] &= \int_{\mathcal{X}} h(x) f(x) dx \\ &= \int_{\mathcal{X}} h(x) \frac{f(x)}{g(x)} g(x) dx \\ & E[\frac{h(x) f(x)}{g(x)}] \\ &= \theta \end{aligned}$$

Calculating tail probabilities

Given $x \sim N(0,1)$ with a sample of size N, the approximation of

$$\Phi(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

By Monte Carlo integration we approximate $P(X \geq x)$ by

$$\Phi(x) = \frac{1}{N} \sum_{i=1}^N I_{\{t_i \geq x\}}$$

Calculating tail probabilities

```
N = 10000; x = 4
t = rnorm(N) >= x ## looking at the lower tail
p_val = sum(t)/N
print(sum(t)) ## only 1 sample greater or equal to 4
```

```
## [1] 1
```

```
print(p_val)
```

```
## [1] 1e-04
```

```
## compare this with
RPval = pnorm(q = 4, lower.tail = FALSE) ## lower tail
print(RPval)
```

```
## [1] 3.167124e-05
```

In the above example, with $N = 10,000$ samples, we saw only 1 observation with $x \geq 4$. To have a stable answer (and variance), we will need to see more samples at this tail. Thus, we turn to importance sampling.

Calculating tail probabilities

For an instrumental importance density, we will need to use a density with a long and heavy tail. Some examples of heavy-tailed distributions are Student's t, Chi-squared and exponential density. We will use the latter for its simplicity and memoryless property.

To continue with the above example, we'll first define our functions h, f, and g to be 1, $N(0,1)$ and $Exp(1)$ truncated at 4. Our instrumental importance function g will be of the form $p(x|x \geq 4) = p(x)/p(x \geq 4) = p(x-4)$. See below a workout of the math, and this link for more information of the memoryless property (http://pages.cs.wisc.edu/~dsmyers/cs547/lecture_9_memoryless_property.pdf)

$$\begin{aligned} g(x) &= \frac{e^{-x}}{\int_4^\infty e^{-x} dx} \\ &= e^{-x} \frac{1}{-e^{-\infty} + e^{-4}} \\ &= e^{-x} \frac{1}{e^{-4}} \\ &= e^{-x} e^{4} \\ &= e^{-x+4} \\ &= e^{-(x-4)} \end{aligned}$$

Calculating tail probabilities

$$\begin{aligned}\frac{h(x_i)f(x_i)}{g(x_i)} &= \frac{1}{\sqrt{2\pi}}e^{-x_i^2/2} \times e^{-(x_i-4)} \\ &= \frac{1}{\sqrt{2\pi}}e^{-x_i^2/2 - (x_i-4)} \\ &= \frac{1}{\sqrt{2\pi}}e^{-(x_i^2/2 + x_i-4)}\end{aligned}$$

Finally, we can now begin to generate samples from g, and use the arithmetic mean to calculate our p-value.

$$E\left[\frac{h(x)f(x)}{g(x)}\right] = \frac{1}{N} \times \frac{1}{\sqrt{2\pi}}\sum_{i=1}^{N} e^{-(x_i^2/2 + x_i-4)}$$

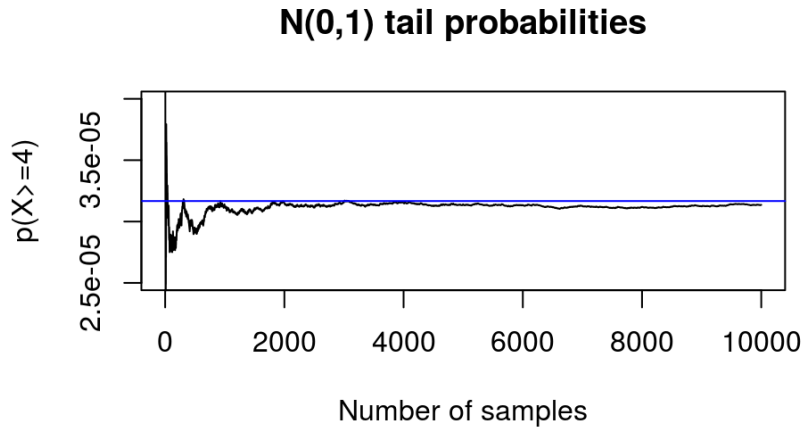
Calculating tail probabilities

```
N = 10^4
t = 4
X = rexp(N) + t
w = dnorm(X)/dexp(X - t) ## importance sampling weights dnorm/dexp(x-4)

theta = mean(w)
cumTheta = cumsum(w) / 1:N
```

Calculating tail probabilities

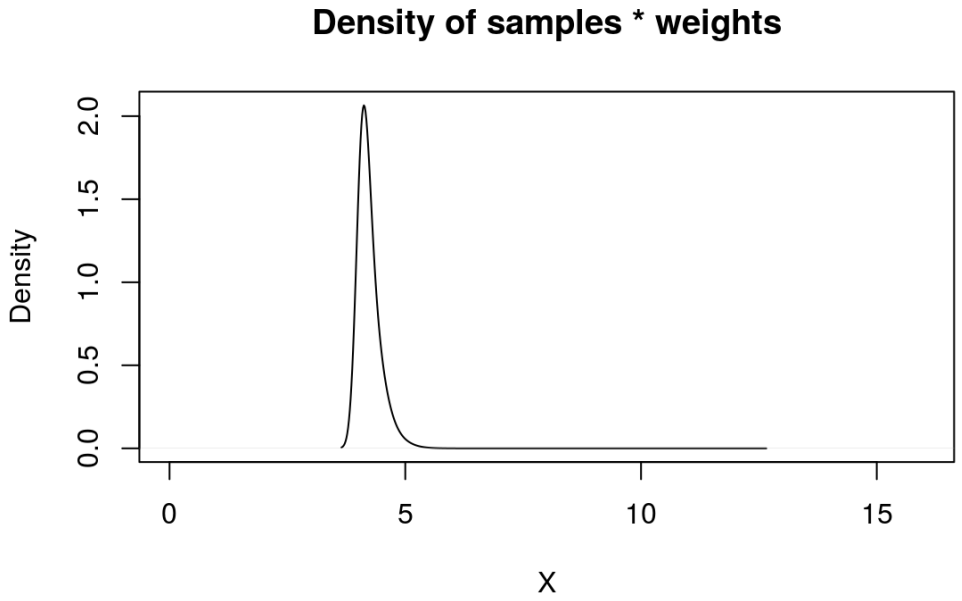
```
plot(cumTheta, type = "l", main = "N(0,1) tail probabilities",
     xlab = "Number of samples", ylab = "p(X>=4)",
     ylim = c(2.5 * 10^(-5), 4 * 10^(-5)))
abline(a = pnorm(t, lower.tail = FALSE), b = 0, col = "blue")
```




```
## Exercise: Compare the results
#print(pnorm(t, lower.tail = FALSE))
#print(theta)
```

Calculating tail probabilities

```
plot(density(X, weights=w/sum(w)), main = "Density of samples * weights",
     xlab = "X", ylab = "Density",
     xlim = c(0,16)) ## plot density of samples from Exp(x - 4) * weights
```



Calculating Expectation of Laplace Distribution

Suppose we have a random variable $X \sim f(x|\mu, b)$, which is the Laplace or Double Exponential distribution.

$$f(x) = \frac{1}{2b} exp\{- \frac{|x-a|}{b}\},\ x \in (0, \infty)$$

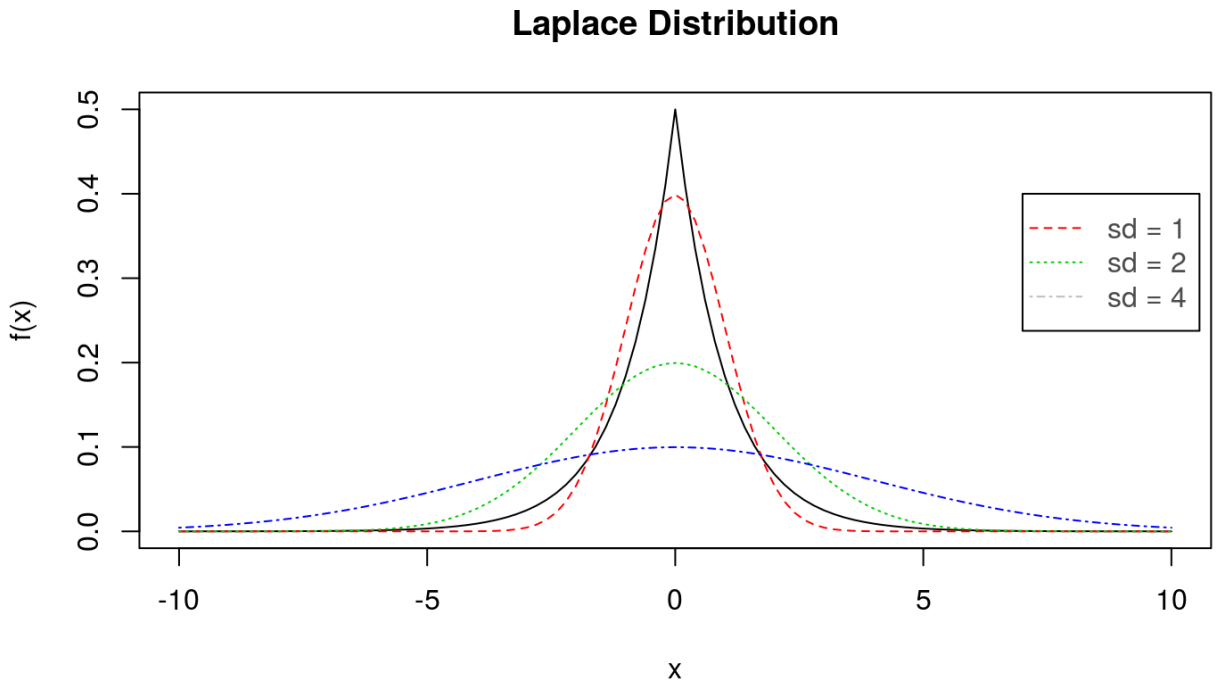
If we let $a = 0$ and $b = 1$, then $f(x) = \frac{1}{2}e^{-|x|}$.

Then, if we would want to calculate $E[X^2]$, then we would need to solve the integral.

$$\begin{aligned} E[X^2] &= \int_{-\infty}^{\infty} x^2 f(x) dx \\ &= \int_{-\infty}^{\infty} x^2 \frac{1}{2b} e^{-|x|} dx \end{aligned}$$

Calculating Expectation of Laplace Distribution

Suppose (for whatever reason) we are unable to generate samples from an exponential distribution. But (for whatever reason) we are able to generate samples $x_i \sim N(0, \sigma)$. Then, we could use $N(0, \sigma)$ as our instrumental density to sample from and use the importance weights $w_i = f(x_i)/g(x_i)$ to improve the convergence to f. Let's try $\sigma = 4$



Calculating Expectation of Laplace Distribution

```
f <- function(x){
  out = (1/2) * exp(x = -1 * abs(x))
  return(out)
}

curve(expr = f, from = -10, to = 10, type = "l",
      main = "Laplace Distribution")

curve(expr = dnorm(x, sd = 1), from = -10, to = 10, type = "l",
      lty = 2, col = 2,
      add = TRUE)

curve(expr = dnorm(x, sd = 2), from = -10, to = 10, type = "l",
      lty = 3, col = 3,
      add = TRUE)


curve(expr = dnorm(x, sd = 4), from = -10, to = 10, type = "l",
      lty = 4, col = 4,
      add = TRUE)

legend(7, 0.4, c("sd = 1", "sd = 2", "sd = 4"), col = c(2, 3, 64),
       text.col = "gray30", lty = c(2, 3, 4), pch = c(NA, NA, NA),
       merge = TRUE)
```

Calculating Expectation of Laplace Distribution

Since we need to have samples that cover the entire domain of the Laplace distribution, it makes sense to set σ higher than 1. Otherwise, we would need many, many samples to obtain a stable answer.

```
N = 10000
x = rnorm(N)
w = f(x)/dnorm(x, sd = 1)
theta <- mean(x = x^2 * w)
print(theta)
```

```
## [1] 1.461255
```

```
x = rnorm(N, sd = 2)
w = f(x)/dnorm(x, sd = 2)
theta <- mean(x = x^2 * w)
print(theta)
```

```
## [1] 2.013747
```

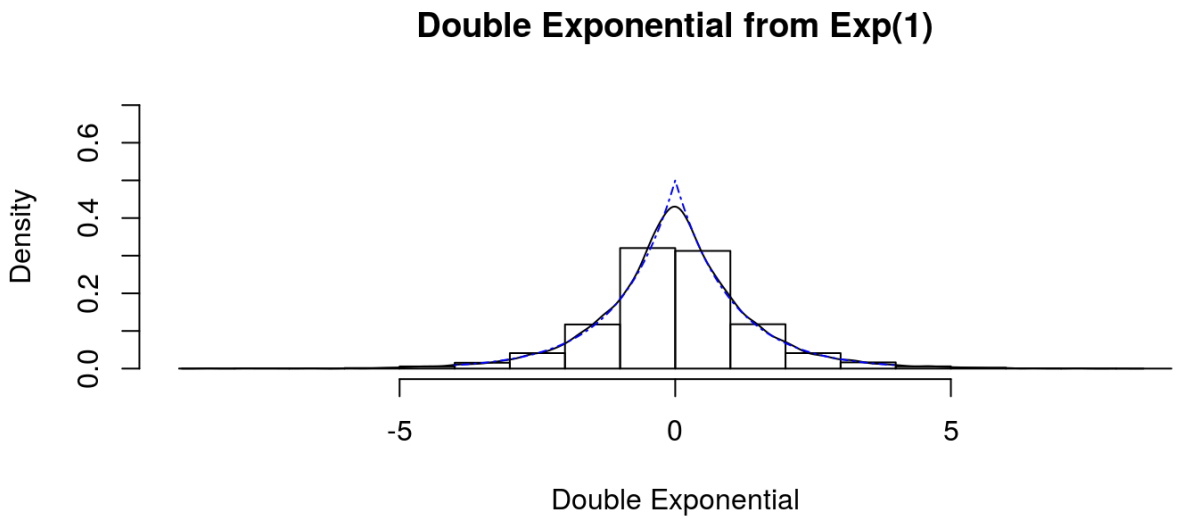
Calculating Expectation of Laplace Distribution

```
x = rnorm(N, sd = 4)
w = f(x)/dnorm(x, sd = 4)
theta <- mean(x = x^2 * w)
print(theta)
```

```
## [1] 2.01577
```

Calculating Expectation of Laplace Distribution

Now to compare this using samples generated from an $Exp(1)$ distribution. We generate a double exponential from an exponential distribution by generating N random variables from $Exp(1)$ then splitting up the distribution (randomly & evenly in approximately half) between the left and right sides of $\mu = 0$. Our importance weights in this case would just be $w_i = 1$. See reference Double Exponential (http://www.itl.nist.gov/div898/handbook/eda/section3/eda366c.htm)




```
## [1] "theta estimate:  1.97292481651938"
```

Calculating Expectation of Laplace Distribution

Code

```
N = 10^4
f <- function(x){
  out = (1/2) * exp(x = -1 * abs(x))
  return(out)
}
pos_neg <- sample(x = c(-1,1), size = N, replace = TRUE)
x = rexp(n = N)
double_exp = pos_neg * x

hist(double_exp, probability = TRUE, ylim = c(0,0.7),
     main = "Double Exponential from Exp(1)",
     xlab = "Double Exponential", ylab = "Density")
lines(density(double_exp))
curve(expr = f, from = -4, to = 4, add = TRUE, col = 4, lty = 4)


theta = mean(double_exp^2)
print(theta)
```

Importance Sampling Resampling

Sampling Importance Resampling (SIR) simulates random variables approximately from the target distribution. SIR is based upon the notion of importance sampling and nonparametric bootstrapping.

Briefly, importance sampling proceeds by sampling from an importance sampling distribution g, where g is the envelope or instrumental distribution. If needed, each point in the sample is weighted to correct the sampling porbabilities so that the weighted sample can be related to the target density f.

As seen previously, the weighted sample can be used to estimate expectations under f.

Importance Sampling Resampling

For the target density f, the weights used to correct sampling probabilities are called the *standardized importance weights*. Since the initial weights $w_i = f(x_i)/g(x_i)$ to not sum to 1, we have to normalize them and thus we obtain

$$\begin{aligned} w_i^* &= \frac{w_i}{\sum_{i=1}^n w_i} \\ &= \frac{f(x_i)/g(x_i)}{\sum_{i=1}^n f(x_i)/g(x_i)} \end{aligned}$$

for a sample of random variables $x_1, \ldots, x_n \sim g(x)$.

Importance Sampling Resampling

We may view importance sampling as approximating f by the discrete distribution having mass w_i on each observed sample point x_i. The SIR algorithm can be described as follows

1. Sample $x_1, \ldots, x_n \sim g(x)$
2. Calculate the standardized importance weights $w_i^* = \frac{f(x_i)/g(x_i)}{\sum_{i=1}^n f(x_i)/g(x_i)}$
3. Resample $y_1, \ldots, y_n$ with replacement with probability w_i (or w_i^*).

A random variable X drawn with the SIR algorithm has a distribution that converges to f as $m \to \infty$.

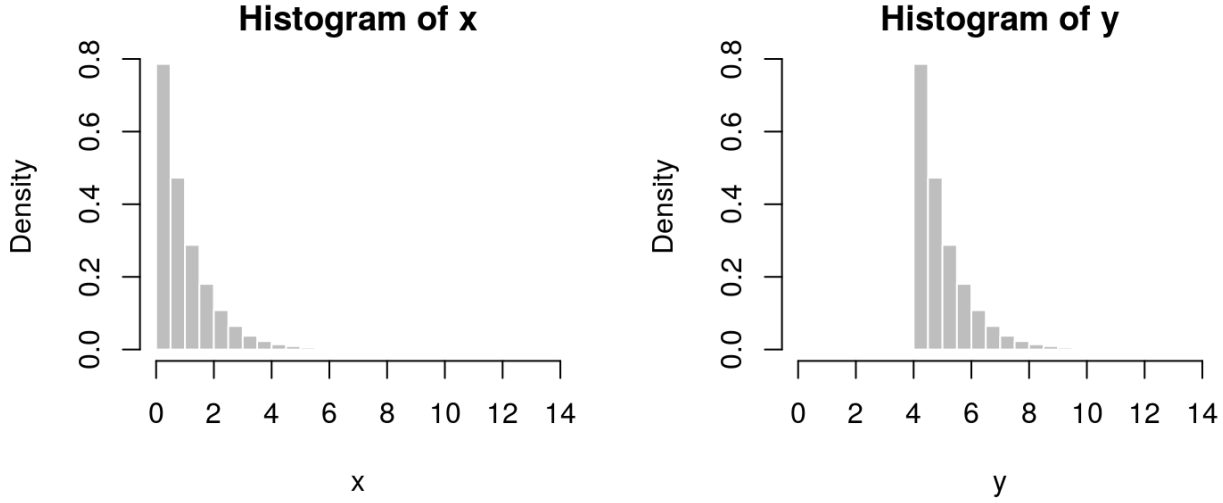
This method introduces some bias to the estimate we are trying to calculate, but for large samples, it is usually negligible.

Tail Probabilities of a *t*-distribution

A good candidate for sampling from a tail distribution is the $Exp(\theta)$ distribution. We can generate random variables from a left truncated exponential distribution, $Y \sim Exp^{+}(\alpha, \theta)$, by simply shifting $X \sim Exp(\theta)$ values by α. Thus, $Y = X + \alpha$

```
n = 5*10^4
x = rexp(n = n, rate = 1)
alpha = 4
y = x + alpha
par(mfrow = c(1,2), mar = c(4,4,2,2))

hist(x, probability = TRUE, xlim = c(0, 14), col = "gray", border = "white")
hist(y, probability = TRUE, xlim = c(0, 14), col = "gray", border = "white")
```



Tail Probabilities of a *t*-distribution

Using the truncated exponential distribution, we can use $Exp^{+}(\alpha, \theta)$ as our importance function for calculating the tail probabilities

```
x = rexp(n = n, rate = 1)
alpha = 10
y = x + alpha
w = dt(x = y, df = 5)/dexp(x = (y-alpha), rate = 1)
MC_p_val = mean(w)
deterministic_p_val = integrate(f = function(x) dt(x, df = 5), alpha, Inf)

print(MC_p_val)
```

```
## [1] 8.321677e-05
```

```
print(deterministic_p_val)
```

```
## 8.547406e-05 with absolute error < 2.1e-05
```

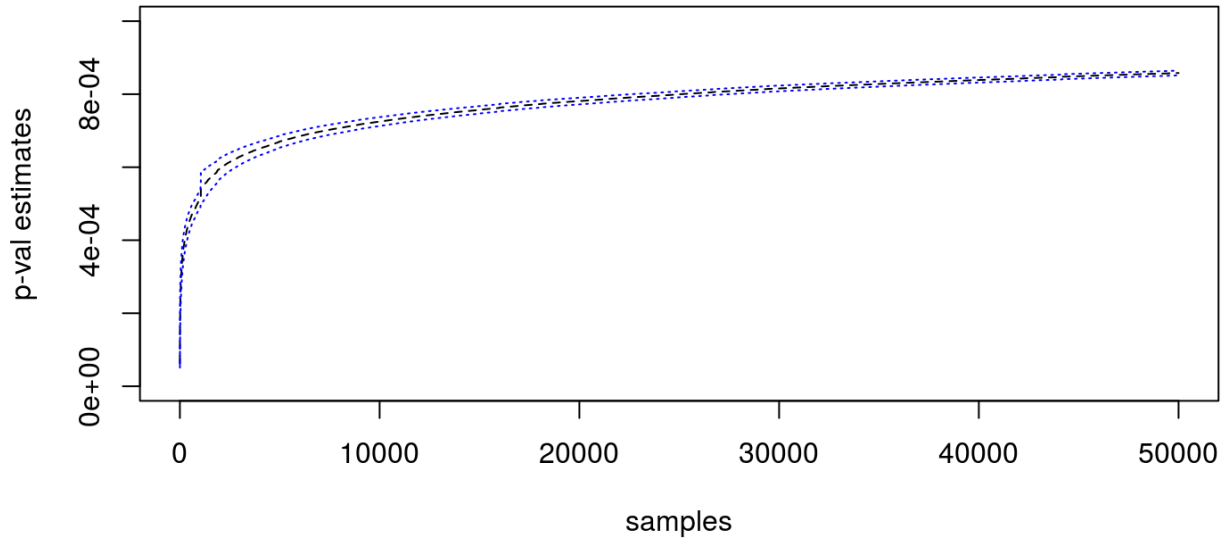
```
print(pt(q = 10, df = 5, lower.tail = FALSE))
```

```
## [1] 8.547379e-05
```

Tail Probabilities of a *t*-distribution

```
pvals = cumsum(w/1:n)
sterr = sqrt(cumsum((w - pvals)^2))/(1:n)

par(mar = c(4,4,2,2))
plot(x = 1:n, y = pvals, type = "l", lty = 2,
     xlab = "samples", ylab = "p-val estimates", ylim = c(0, 0.001))
lines(pvals + 1.96*sterr, col = "blue", lty = 3)
lines(pvals - 1.96*sterr, col ="blue", lty = 3)
```



Normal-Cauchy Bayes estimator

For a single Normal observation $x = 2.5$, where $x \sim N(\theta, 1)$ with a Cauchy prior on its mean, $\theta \sim C(0, 1)$, estimate the posterior distribution and mean.

The Bayes formula to obtain the posterior distribution is given by

$$\pi(\theta|x) = \frac{f(x|\theta) \times \pi(\theta)}{m(x)}$$

where $f(x|\theta)$ is the likelihood function, $\pi(\theta)$ is the prior distribution and $m(x)$ is the marginal distribution of x. We calculate $m(x) = \int_{-\infty}^{\infty} f(x|\theta) \times \pi(\theta) d\theta$. The standard Cauchy distribution has pdf

$$\pi(\theta) = \frac{1}{\pi \times 1 + \theta^2}$$

The likelihood is obtained by the following

$$f(x|\theta) = \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}(x - \theta)^2)$$

Normal-Cauchy Bayes estimator

Therefore,

$$\begin{aligned} f(x|\theta) \times \pi(\theta) &= \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}(x-\theta)^2) \times \frac{1}{\pi \times 1 + \theta^2} \\ &= (2\pi)^{-1/2}\pi^{-1} exp(-\frac{1}{2}(x-\theta)^2)(1+\theta^2)^{-1} \\ &\propto exp(-\frac{1}{2}(x-\theta)^2)(1+\theta^2)^{-1} \end{aligned}$$

Next, we obtain our marginal distribution $m(x)$

$$\begin{aligned} m(x) &= \int_{-\infty}^{\infty} f(x|\theta) \times \pi(\theta) d\theta \\ &= \int_{-\infty}^{\infty} exp(-\frac{1}{2}(x-\theta)^2) \times (1+\theta^2)^{-1} d\theta \end{aligned}$$

where the first component of that integral is proportional to $N(x, 1)$. Therefore, we can estimate the $m(x)$, and posterior distribution, by sampling $\theta \sim N(x, 1)$ and weighting the values through a Cauchy density function.

Normal-Cauchy Bayes estimator

$$p(\theta|\mathbf{x}) = \frac{1}{(1+\theta^2)} e^{-(x-\theta)^2/2} \Big/ \int_{-\infty}^{\infty} \frac{1}{(1+\theta^2)} e^{(x-\theta)^2)/2} d\theta$$

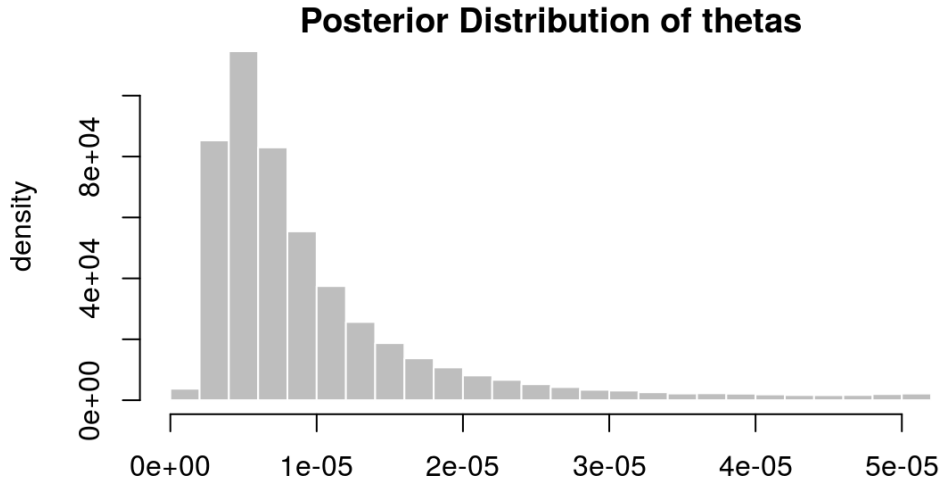
Note that the posterior can also be interpreted as a self-normalized importance sampling approximation with the importance function being the normal density and the importance ratio being equal to $\frac{1}{(1+\theta^2)}$

Normal-Cauchy Bayes estimator

```
N = 10^5
X = 2.5

thetas = rnorm(N, X, sd = 1)
weights = dcauchy(x = thetas, location = 0, scale = 1) ## 1/(pi * (1 + s^2))
sumOfWeights = sum(weights)

posteriorDist =  (weights )/(sumOfWeights)
hist(posteriorDist, probability = TRUE,
     col = "gray", border = "white",
    main = "Posterior Distribution of thetas",
    xlab = "theta", ylab = "density")
```



Normal-Cauchy Bayes estimator

The mean of the posterior distribution is calcaulated by

$$E(\theta|x) = \int_{-\infty}^{\infty} \theta \times \frac{exp(-\frac{1}{2}(x-\theta)^2) \times (1+\theta^2)^{-1}}{m(x)} d\theta$$

where

$$m(x) = \int_{-\infty}^{\infty} exp(-\frac{1}{2}(x-\theta)^2) \times (1+\theta^2)^{-1} d\theta$$

Normal-Cauchy Bayes estimator

Therefore the expectation of the mean of θ is given by

$$E(\theta|x) \approx \sum_{i=1}^{n} \frac{\theta_i}{(1+\theta_i^2)} e^{-(x-\theta_i)^2/2} \Big/ \sum_{i=1}^{n} \frac{1}{(1+\theta_i^2)} e^{(x-\theta_i)^2)/2}$$

Normal-Cauchy Bayes estimator

```
## The expectation of the posterior distribution
posteriorThetaEst = sum((thetas * weights)/ sumOfWeights)
print(posteriorThetaEst)
```

```
## [1] 1.754396
```

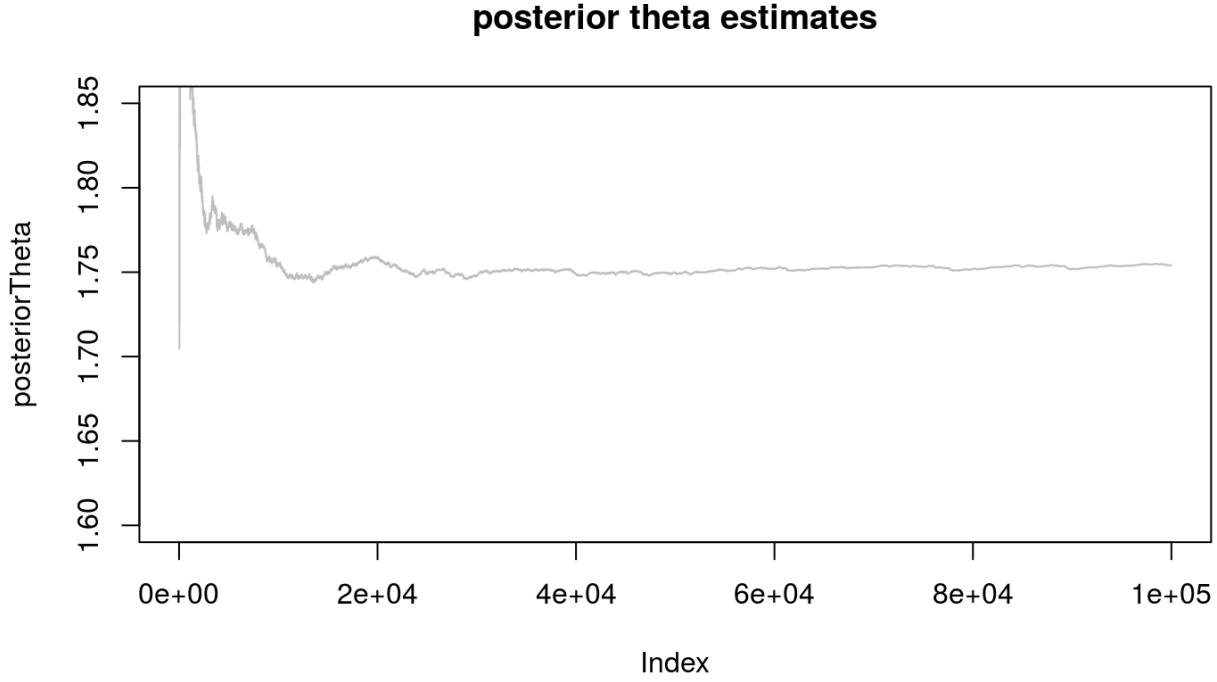
```
## Effective Sample Size
ESS = 1 / sum( (weights / sumOfWeights)^2) ## Effective Sample Size
print(ESS)
```

```
## [1] 56451.62
```

Normal-Cauchy Bayes estimator

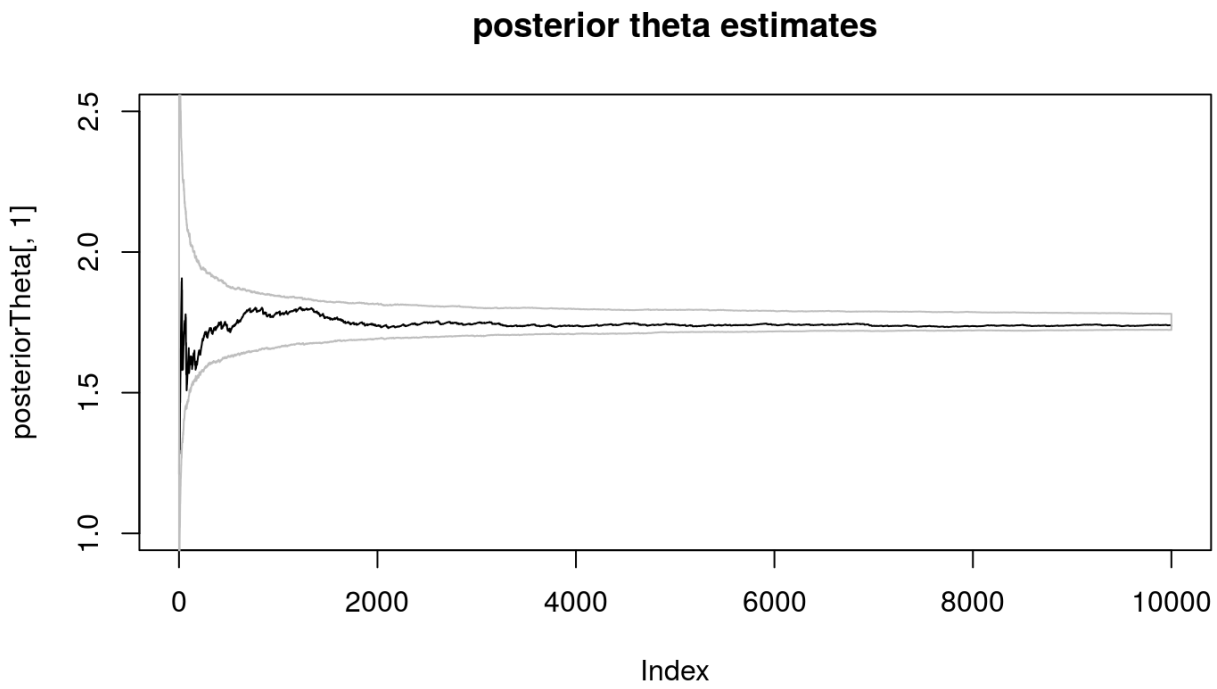
```
posteriorTheta = cumsum(thetas * weights) / cumsum(weights)

plot(posteriorTheta, type = "l", main = "posterior theta estimates",
     ylim = c(1.6, 1.85), col = "gray")
```



Normal-Cauchy Bayes estimator

```
X = 2.5
N = 10^4
M = 1000
thetas = matrix(data = rnorm(N*M), ncol = M) + X ## N(0,1) + 2.5
weights = dcauchy(x = thetas, location = 0, scale = 1)
posteriorTheta = apply(thetas * weights, 2, cumsum) / apply(weights, 2, cumsum)
```



Normal-Cauchy Bayes estimator

Code

```
## look at the first column estimates
plot(posteriorTheta[,1], main = "posterior theta estimates",
    ylim = c(1, 2.5), type = "l")
## grab the 2.5 and 97.5 quantiles for a 95% CI
confInts = apply(posteriorTheta, 1, quantile, c(0.025, 0.975))
polygon(x = c(1:N, N:1), y = c(confInts[1,], rev(confInts[2,])), border = "gray")
```