Monte Carlo Integration

Jonathan Navarrete

May 20, 2017

Introduction

Beyond 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
3. Importance Sampling Resampling

Monte Carlo Integration

Given a function $h(x)$ for which we wish to integrate over [a,b], $\int_a^b h(x)dx$, we can treat this deterministic problem as a stochastic one to find the solution. Treat X as if a random variable with density $f(x)$, then the mathimatical expecation of the random variable $Y = h(X)$ is

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

If a random sample $X_1, ..., X_n$ is generated from $f(x)$, an unbiased estimator of $E[h(X)]$ is the sample mean. Review: Sample mean

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

As an example, suppose we have a function $h(x) = 3x^2$ for which we wish to integrate over the interval 0 to 2. We could apply deterministic numerical approximation methods (see R's `integrate`) or we could treat x as a random variable from a $Unif(0, 2)$ whose pdf is simply $f(x) = \frac{1}{2-0} = \frac{1}{2}$. If we now generate some $N = 1{,}000$ random values from $f(x)$ and evaluate them at $h(x)$, then take the mean, we'd be calculating the expected value of $h(x)$,

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

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

```
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)

v = 2 * mean(h(X)) ## 2 * E[h(x)]

print(v) ## approximately 8
```

```
## [1] 8.057179
```

Now, some of you may ask, "why is it that we can use the empirical average to create an estimate?" Well, that's because the discrete expected value of $h(x)$ is $E[h(x)] = h(x_1)P(X = x_1) + ... + h(x_1)P(X = x_n)$ where $P(X = x_i) = \frac{1}{n}$ since $x_i \sim Unif(0, 2)$.

Now, to generalize the method above. Given a function $h(x)$ whose integral is well defined, for which 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 from.
2. Generate $x_1, ..., x_n \sim f(x)$
3. Compute $(b-a) \times \bar{g_n}$, where $\bar{g_n} = \frac{1}{n}\sum_{i=1}^n h(x_i)$

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

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

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

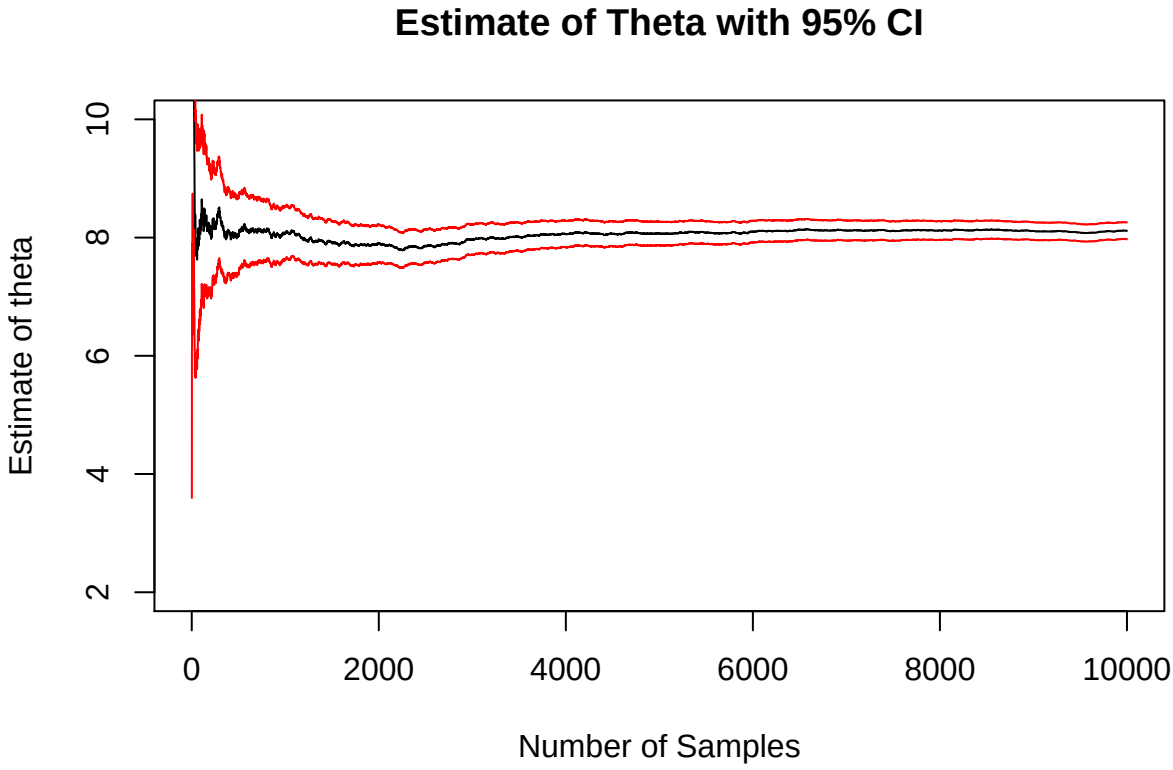
$$\frac{\sigma^2}{n} = \frac{1}{n^2}\sum_{i=1}^n (h(x_i) - \hat{\theta})^2$$

So, the standard error estimate is

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

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)
  denom = 1:n
  result = num/denom
  return(result)
}


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

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

plot(x = 1:N, y = thetas, type = "l", ylim = c(2, 10), 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 = "red") ## CI
lines(x = 1:N, y = thetas - 1.96*SE, col = "red") ## 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.

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

print("Theta estimate: ")
```

```
## [1] "Theta estimate: "
```

```
print(thetaHat)
```

```
## [1] 8.115206
```

```
print("Confidence Intervals")
```

```
## [1] "Confidence Intervals"
```

```
print(ci)
```

```
## [1] 7.973447 8.256965
```

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

Strong Law of Large Numbers

The SLLN says that for a given random sample $X_1, ..., X_n$ 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

Central Limit Theorem

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

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

as $n \to \infty$. See: Central Limit Theorem

Example: Normal p-value estimate for unbounded region

We will next explore how to generate $Normal(\mu, \sigma)$ 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 $\mu = 1$ and $\sigma = 1$, such that we are only dealing with the standard Normal CDF. We are interested in calculating a Monte Carlo estimate for $\Phi(x)$

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

Our first problem is dealing with bounds $(-\infty, x)$. We cannot direcly apply our MC integration algorithm directly because the integral range $(-\infty, x)$, but a way around this issue is by breaking 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 $P(X < 0) = 1/2$. Now, all we need to concern ourselves with is the region $(0, x)$.

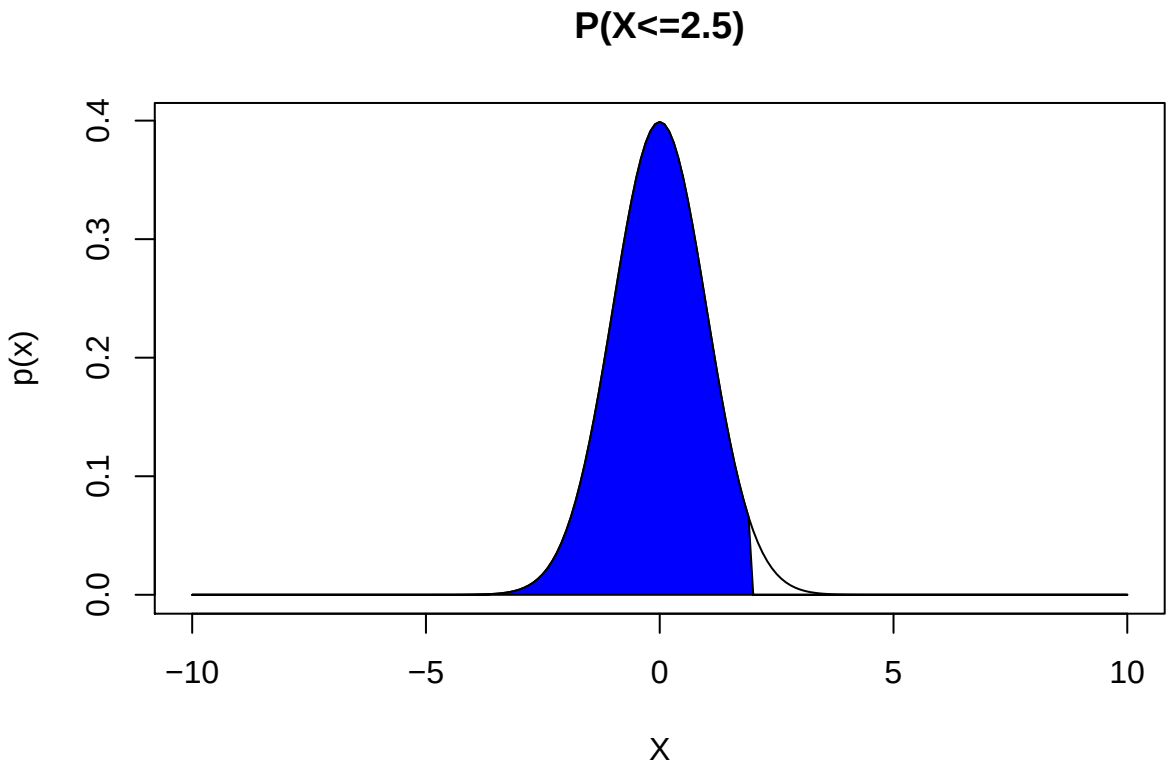
Now, assuming $x > 0$, then we use the Monte Carlo estimator as is

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

However, what if $x < 0$? Well, then, again exploiting the symmetry property of $N(\mu, \sigma)$, we esimate $\Phi(X \leq x)$ by $1 - \Phi(-x)$. I'll illustrate how this will be done.

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

```r
x <- seq(from = -10, to = 10, by = 0.1)

out = dnorm(x)
out[x >= 2] = 0

plot(x, dnorm(x), type = "l", main = "P(X<=2.5)",
     xlab = "X", ylab = "p(x)")
polygon(x, out, col = "blue")
```



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

```r
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] 0.9983411
```

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

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

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 $P(X \leq x) = \theta$,

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

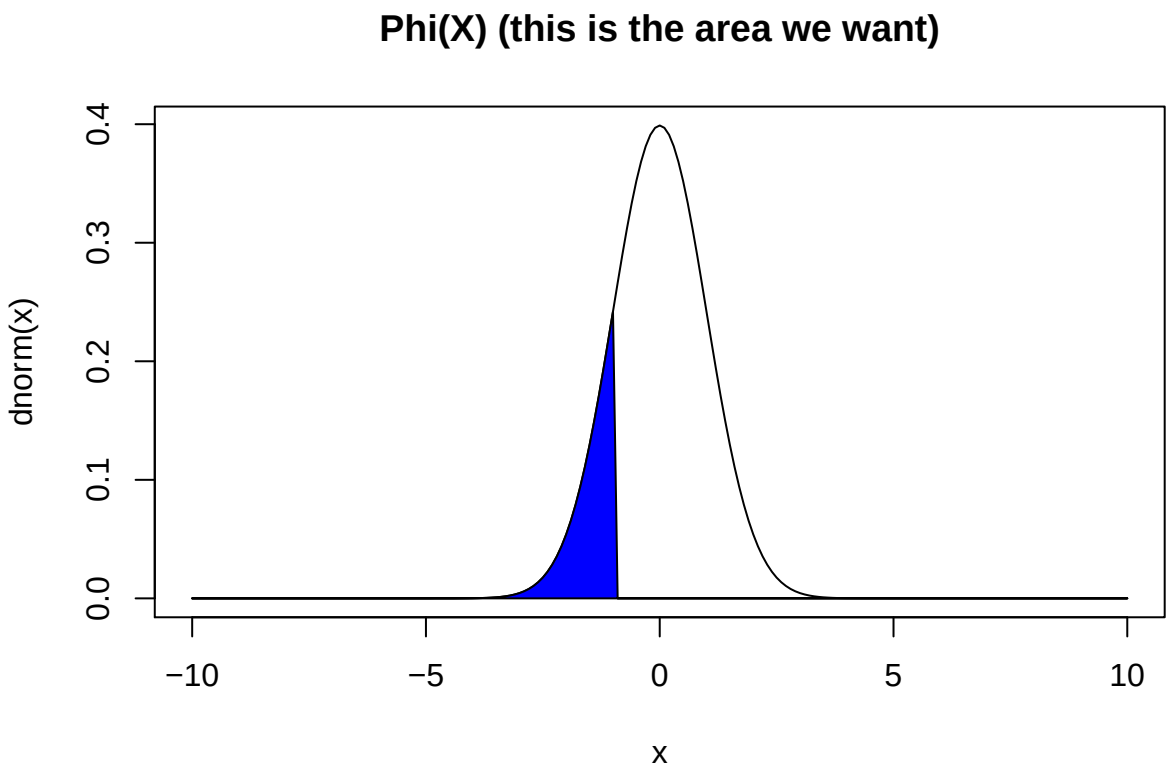
and let $Y = t/x \rightarrow t = xY$ such that

$$\begin{aligned}&= \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_{0}^{1} x e^{-(x \times Y)^2/2} \frac{1}{x} dy \\ &\approx \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \frac{1}{N} \sum_{i=1}^{N} x_i e^{-(x_i \times Y)^2/2} \\ &= \hat{\theta}\end{aligned}$$

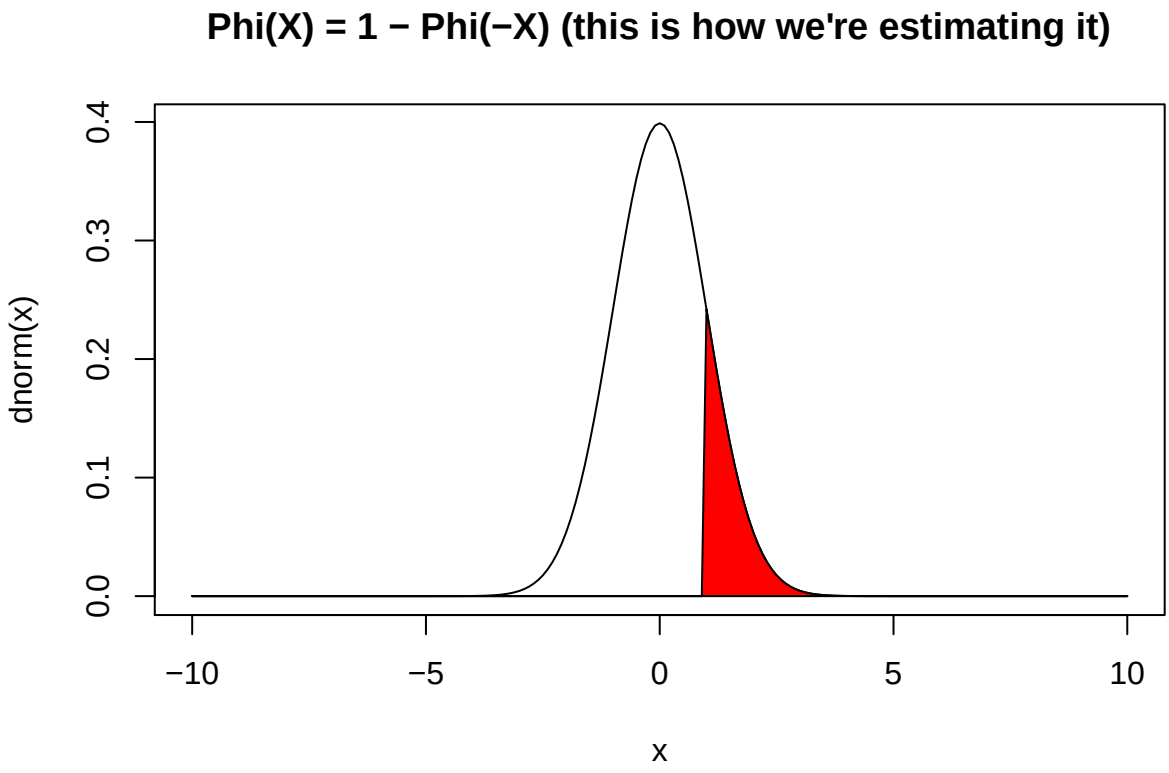
for $x > 0$ and $x_i \sim Unif(0, 1)$.

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

```
x <- seq(from = -10, to = 10, by = 0.1)

out = dnorm(x)
out[x > -1] = 0

plot(x, dnorm(x), type = "l", main = "Phi(X) (this is the area we want)")
polygon(x, out, col = "blue")
```



```
out = dnorm(x)
out[x < 1] = 0

plot(x, dnorm(x), type = "l", main = "Phi(X) = 1 - Phi(-X) (this is how we're estimating it)")
polygon(x, out, col = "red")
```



Calculating N(0,1) tail probabilities

Suppose $Z \sim N(0,1)$, then $\Phi(x) = P(Z \leq x) = E[I_{Z \leq x}] \approx \frac{1}{N} \sum_{i=1}^{N} I_{z_i \leq x}$ 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] 0.9996
```

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

```
## [1] 4
```

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. Suppose that we wish to weight samples x by density g such that this would improve results for tail probabilities.

We can rewrite our previous result as

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

Example: 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 $P(X \leq x)$ by

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

```
set.seed(9124555)
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.

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

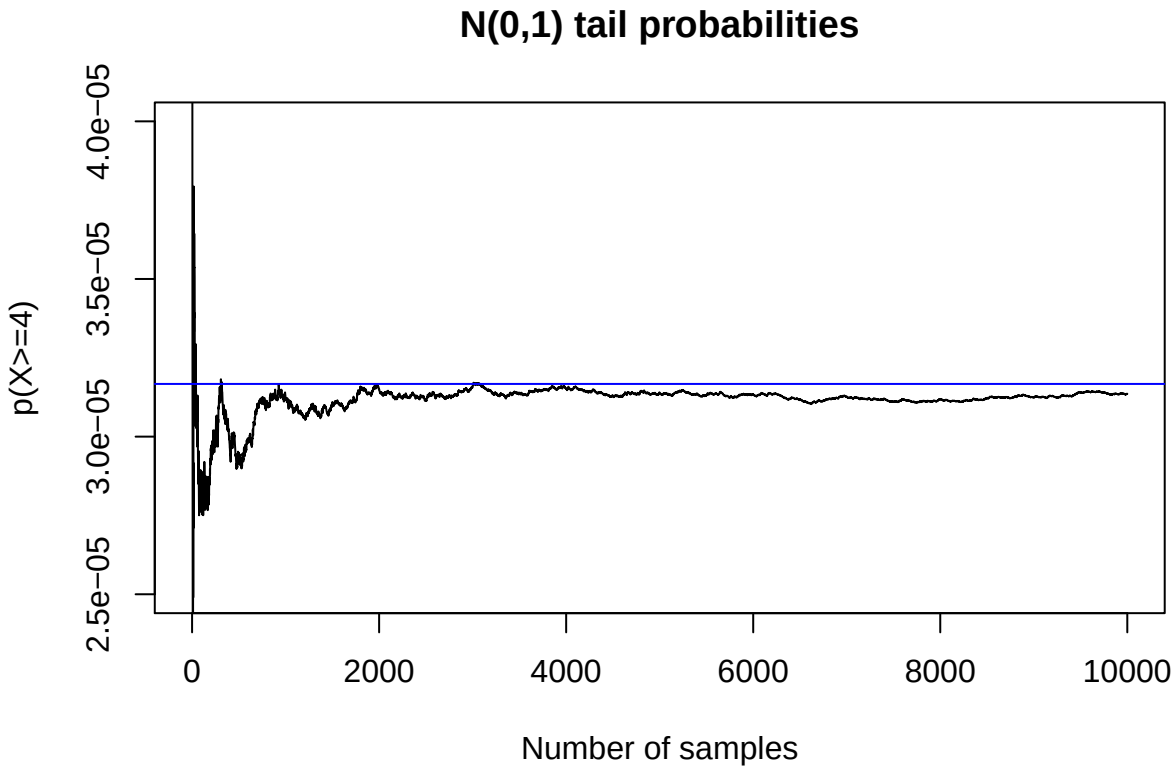
$$\begin{aligned} g(x) &= \frac{e^{-x}}{\int_4^\infty e^{-x}dx} \\ &= e^{-x}\frac{1}{-e^{-\infty} + e^{-4}} \\ &= e^{-x}\frac{1}{e^{-4}} \\ &= e^{-x}e^{4} \\ &= e^{-x+4} \\ &= e^{-(x-4)} \end{aligned}$$

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

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

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

```
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

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")
```



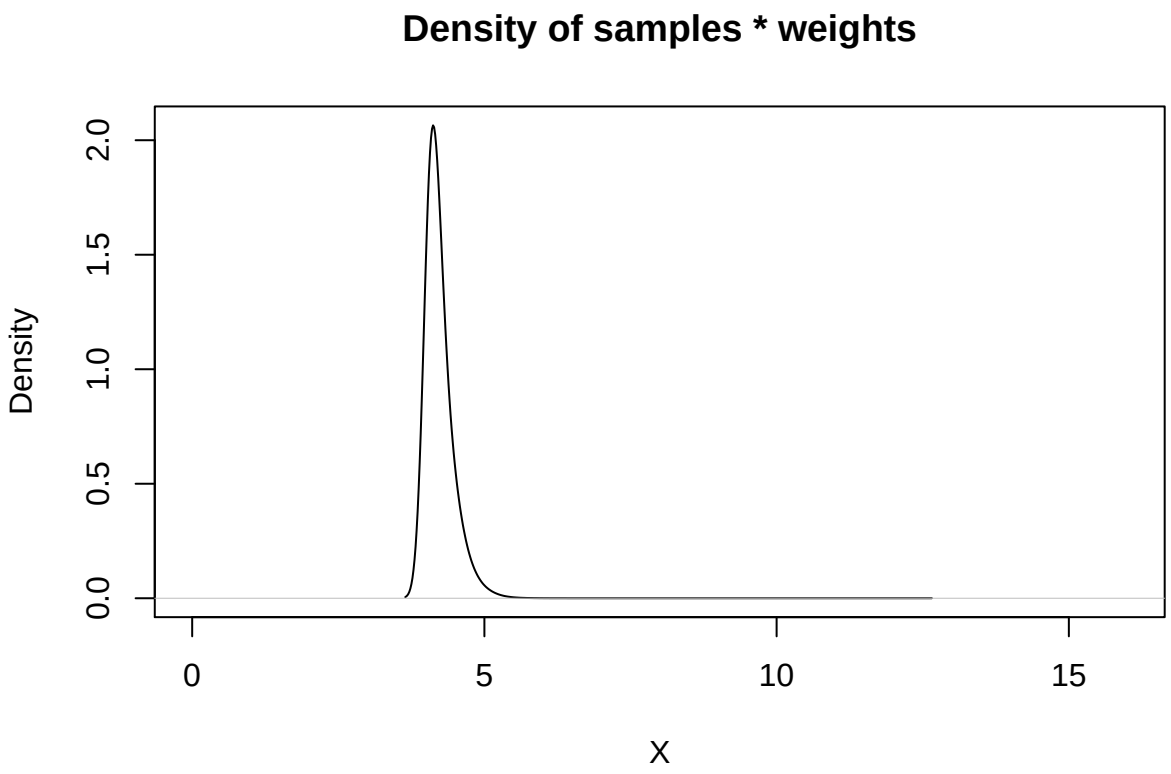

```
print(pnorm(t, lower.tail = FALSE))
```

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

```
print(theta) ## close results!
```

```
## [1] 3.135722e-05
```

```
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
```


Density of samples * weights
Density
0.0
0.5
1.0
1.5
2.0
0
5
10
15
X

Importance Sampling Resampling

Sampling Importance Resampling (SIR) simulates random variables approximately from 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. g will here be our envelope or instrumental importance function. 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.

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 $x_1, ..., x_n \sim g(x)$.

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 $x_1, ..., x_n \sim g(x)$
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 $y_1, ..., y_n$ 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 $m \to \infty$.

Example Beta(a,b) posterior

Suppose we have $x_i \sim Beta(a, b)$ with priors on both a and b. There exists a family of conjugate priors of the form

$$\pi(a, b) \propto \{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\}^\lambda x_0^a y_0^b$$

where λ, x_0, y_0 are hyperparameters; See: Does the beta distribution have a conjugate prior? . The posterior is then equal to

$$\pi(a, b|x) \propto \{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\}^{\lambda+1} [x x_0]^a [(1-x) y_0]^b$$

This family of distributions is intractable if only because of the difficulty of dealing with the gamma functions. Simulating directly from $\pi(a, b|x)$ is therefore impossible. We thus need to use a subsitute distribution $g(a, b)$, and we can get a preliminary idea by looking at an image representation of $\pi(a, b|x)$. If we take $\lambda = 1$, $x_0 = 0.5$, $y_0 = 0.5$, and $x = 0.6$.

$$\begin{aligned} \pi(a, b|x) &\propto \{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\}^2 [0.6 \times 0.5]^a [(1-0.6) \times 0.5]^b \\ &\propto \{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\}^2 [0.3]^a [0.2]^b \end{aligned}$$

Now, for computational purposes, we will use a log-transform then exponential transform.

$$\begin{aligned} &\propto \{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\}^2 [0.3]^a [0.2]^b \\ &\propto exp\{\{2 \times (\Gamma(a+b) - \Gamma(a) - \Gamma(b)) + (a \times 0.3) + (b \times 0.2)\} \end{aligned}$$

```r
posterior = function(a,b){
  x = exp(2*(lgamma(a+b) - lgamma(a) - lgamma(b)) +
            a*log(0.3) + b* log(0.2))
  return(x)
}

a = 1:150
b = 1:100
post = outer(a,b,posterior)
image(a, b, post,
      xlab = expression(alpha), ylab = expression(beta),
      main = paste("Controur plot of the ", expression(Pi), "(a,b|x)", sep = ""))
contour(a, b, post, add = TRUE)
```