STA2311: Advanced Computational Methods for Statistics I

Class 6: Simulation and Monte Carlo

Radu Craiu   Robert Zimmerman

University of Toronto

October 24, 2023

Section 1

Introduction

Introduction

- Simulation plays a crucial role in statistics
- In the frequentist realm, simulation can help us verify:
  - that a proposed model fits the observed data
  - the properties of a model
- In the Bayesian realm, simulation allows us to study posterior distributions
  - by sampling from the posterior
  - ...thereby allowing us to build a picture of the posterior
  - For example: if $\theta_1, \theta_2, \ldots, \theta_n$ are an iid sample from a posterior $f(\cdot \mid \boldsymbol{x})$, then

$$\mathbb{E}_f[h(\theta) \mid \boldsymbol{x}] = \int h(\theta) \cdot f(\theta \mid \boldsymbol{x}) \, \mathrm{d}\theta \approx \frac{1}{n} \sum_{i=1}^{n} h(\theta_i)$$

Simulation: The Basics

- Let f be a *distribution* of interest (posterior, predictive, etc)
- What does it mean to simulate $X \sim f$?
- Formally speaking, we mean generating an observed value (or "realization") x of a random variable X that is f-distributed.
- We have already used simulation in Class 1 and Classes 3–4, but these were settings in which the targets were so standard that we could just use R's built-in routines directly
  - e.g., to sample $X \sim \mathcal{N}(0, 1)$ we could simply use `x <- rnorm(n=1)`
- For the remainder of the course, we will study techniques for sampling from arbitrary distributions (either exactly or approximately)

Section 2

Basic Theory of Simulation

Random Number Generators

- Every non-constant simulated random variable starts with the generation of a (pseudo)-random number generated by a *random number generator (RNG)*
- Random number generation is a science in itself
- There has been much work over the years developing RNGs whose outputs satisfy desirable properties
- No RNG is truly random: their outputs can be reproduced exactly using the same *seed*
    - Exception: some generators use natural phenomena (e.g., radioactive decay)

Random Number Generation

- Usually it is enough to generate a random number in $(0, 1)$
  - If $u \in (0, 1)$ and $a < b$, then $u' = a + (b - a) \cdot u \in (a, b)$
- Good RNGs will produce a sequence of outputs which *appear* to be independent
- On a computer these will always be rational numbers (due to finite precision), but in practice this usually isn't an issue
- So from now on, we assume we have access to an RNG that will generate $U_1, U_2, \ldots \overset{iid}{\sim} \text{Unif}(0, 1)$

Random Variables

- How do we simulate random variables or random vectors from an arbitrary distribution?
- Generally speaking, there is no method that works for all distributions
- The most basic method (and arguably most fundamental) is called the *inverse cdf method*, which is based on the following theorem

The Inverse CDF Method

Theorem (Probability Integral Transform)

Let f be the density of a continuous random variable with cdf F and inverse cdf F^{-1}. If $U \sim Unif(0,1)$ and $X = F^{-1}(U)$, then $X \sim f$.

- In other words, to simulate a continuous random variable, it suffices that we know how to compute its inverse cdf
- More generally, for *any* cdf F (continuous or not), we have that $X = F^{-}(U)$ has cdf F, where $U \sim \text{Unif}(0,1)$ and

$$F^{-}(p) := \inf\{x \in \mathbb{R} : F(x) \geq p\}$$

  - When F is continuous, $F^{-} = F^{-1}$

The Inverse CDF Method (Continued)

- If we can compute F^{-}, the *inverse cdf method* provides a way to simulate $X \sim f$:
  1. Simulate $U \sim \text{Unif}(0, 1)$
  2. Take $X = F^{-}(U)$

The Inverse CDF Method: Example

```r
set.seed(2311)

n <- 5000

lambda <- 3

u <- runif(n=n)
x <- -(1/lambda)*log(1-u)
y <- rexp(n=n, rate=lambda)

hist(x)
hist(y)

ks.test(x,y)
```

When Doesn't It Work?

- The inverse cdf method cannot be used when...
- ... F is the cdf of a random vector
  - Because then F lacks an inverse (since $\text{dom}F \neq \text{ran}F$)
- ... F^{-1} does not have a closed form
- But other methods are available!

Simulating Discrete Random Variables

- If f is discrete, then $X \sim f$ is supported on $\mathcal{X} = \{x_1, x_2, \ldots\}$ (which may be finite) where WLOG $x_i < x_{i+1}$ for all i
- Then the intervals $I_i = (F(x_{i-1}), F(x_i)]$ partition $(0, 1]$, where $F(x_0) := 0$ and $F(x_i) = \sum_{j=1}^{i} f(x_j)$
- Then

$$f(x_i) = F(x_i) - F(x_{i-1}) = \mathbb{P}(F(x_{i-1}) < U \leq F(x_i)) = \mathbb{P}(U \in I_i)$$

  where $U \sim \text{Unif}(0, 1)$
- So $\sum_{i \geq 1} x_i \cdot \mathbb{1}_{U \in A_i} \sim f$

Simulating Discrete Random Variables: Example

```r
set.seed(2311)

n <- 5000

lambda <- 5
X <- 0:100
F.p <- c(0, cumsum(lambda^X*exp(-lambda)/factorial(X)))

u <- runif(n=n)
x <- findInterval(u, F.p) - 1

y <- rpois(n=n, lambda=lambda)

hist(x)
hist(y)

ks.test(x,y)
```

Simulating Discrete Random Variables: Example

- Sometimes, we can sometimes write $X \sim f$ as a function of other random variables which are easier to simulate from
- For example, if $U_1, U_2, \ldots \overset{iid}{\sim} \text{Unif}(0,1)$, then $\min\{j \in \mathbb{N} : \prod_{i=1}^{j} U_i < e^{-\lambda}\} \sim \text{Poisson}(\lambda)$

```r
set.seed(2311)

n <- 5000
N <- 100
x <- rep(0, n)

lambda <- 5

for (i in 1:n) { x[i] <- which(cumprod(runif(n=N)) < exp(-lambda))[1] - 1}

y <- rpois(n=n, lambda=lambda)

hist(x)
hist(y)
ks.test(x,y)
```

Stochastic Representations: Mixture Models

- Techniques like these are very handy when the inverse cdf method fails
- For example, suppose $X \sim f = \sum_{i \geq 1} p_i \cdot f_i$ where each $p_i > 0$ with $\sum_{i \geq 1} p_i = 1$ and the f_i's are distributions which are easy to sample from
  - The f_i's should all have the same support!
- Let Z be discrete with $\mathbb{P}(Z = i) = p_i$ and let $X \mid Z \sim f_Z$
- Then it is easily shown that $X \sim f$
- So to simulate from f, first simulate Z, and then simulate from f_Z
- The same procedure is valid when $f = \int p(z) \cdot f_z \, \mathrm{d}z$ and $p(z)$ is a density supported on $(0, 1)$

Stochastic Representations: Random Vectors

- Simulating a random vector $\boldsymbol{X}$ with dependent components is (generally) very difficult
- In theory, the cdf F of any random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ can be written as

$$F(\boldsymbol{x}) = C(F_1(x_1), \ldots, F_d(x_d)),$$

  where F_h is the marginal cdf of X_h and C is some copula

  - Eschewing technicalities, a *copula* is the cdf of some random vector with uniform marginals

- In principle, if the copula C of $\boldsymbol{X}$ is known, one can sample $\boldsymbol{X}$ by first sampling $U_1, \ldots, U_d$ from C (which imposes the correct dependence structure) and then taking $X_h = F_h(U_h)$ (which imposes the correct marginal distributions)
- In practice, unless C is specified, this rarely works because identifying C from F is usually very difficult, if not impossible

Stochastic Representations: Bivariate Normal

- Sometimes the components of $\boldsymbol{X}$ have convenient stochastic representations
- For example, if $U_1, U_2 \overset{iid}{\sim} \text{Unif}(0,1)$ and

$$\begin{aligned}
Z_1 &= \sqrt{-2\log(U_1)}\cos(2\pi U_2) \\
Z_2 &= \sqrt{-2\log(U_1)}\sin(2\pi U_2) \\
X_1 &= \mu_1 + \sigma_1 Z_1 \\
X_2 &= \mu_2 + \sigma_2\left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right),
\end{aligned}$$

then

$$(X_1, X_2) \sim \mathcal{N}_2\left(\begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & \sigma_1\sigma_1\rho \\ \sigma_1\sigma_1\rho & \sigma_2^2\end{bmatrix}\right)$$

- This is the *Box-Muller transform*

Box-Muller Transform: Example

```r
set.seed(2311)

n <- 5000

mu <- c(-2, 2)
rho <- 0.8
sigma <- c(2, 1)

u1 <- runif(n=n)
u2 <- runif(n=n)

z1 <- sqrt(-2*log(u1))*cos(2*pi*u2)
z2 <- sqrt(-2*log(u1))*sin(2*pi*u2)

x1 <- mu[1] + sigma[1]*z1
x2 <- mu[2] + sigma[2]*(rho*z1 + sqrt(1-rho^2)*z2)

library(ggplot2)
ggplot(data.frame(x=x1, y=x2), aes(x,y)) + stat_bin_hex()
```

Section 3

Monte Carlo Methods

The Basics

- The last two examples were cases where the target random variable had a convenient stochastic representation in terms of other random variables that were easy to sample from
- Such examples are quite rare, and each method is specific to the target distribution
- However, each requires a pre-determined number of samples (usually 1)
- For more general distributions, one can instead turn to *Monte Carlo methods*
- Monte Carlo methods are a large class of sampling algorithms that rely on *repeated* sampling to obtain either exact or approximate samples from a target distribution

A Classical Example: Estimating π

- If $U_1, U_2 \overset{iid}{\sim} \text{Unif}(0,1)$, check that $\mathbb{P}(U_1^2 + U_2^2 < 1) = \pi/4$
- So for $U_{1,1}, \ldots, U_{1,n}, U_{2,1}, \ldots, U_{2,n} \overset{iid}{\sim} \text{Unif}(0,1)$, the LLN gives

$$\pi = 4 \cdot \mathbb{P}(U_1^2 + U_2^2 < 1) = 4 \cdot \mathbb{E}[\mathbb{1}_{U_1^2+U_2^2<1}] \approx \frac{4}{n}\sum_{i=1}^{n} \mathbb{1}_{U_{1,i}^2+U_{2,i}^2<1}$$

```
set.seed(2311)

n <- 10000

u1 <- runif(n=n)
u2 <- runif(n=n)

incirc <- sqrt(u1^2 + u2^2) < 1

pi.est <- mean(incirc)*4
```

Rejection Sampling

- What if, instead of simply estimating the mean, we want a sample $X \sim f$ but none of the previous methods work?
- Again, suppose also that g is easier to sample from, and this time there exists some constant $c > 1$ such that $f(x) \leq c \cdot g(x)$ for all $x \in \operatorname{Supp} f$
- Suppose we sample $Y \sim g$ and *accept* it (i.e., set $X = Y$) with probability $\frac{f(Y)}{c \cdot g(Y)}$
- We can do this by independently drawing $U \sim \text{Unif}(0, 1)$ and setting $X = Y$ if $U < \frac{f(Y)}{c \cdot g(Y)}$

Rejection Sampling (Continued)

- Then...

$$
\begin{aligned}
&\mathbb{P}(X \le x) \\
&= \mathbb{P}\left(Y \le x \mid U < \frac{f(Y)}{c \cdot g(Y)}\right) \\
&= \frac{\mathbb{P}\left(Y \le x \wedge U < \frac{f(Y)}{c \cdot g(Y)}\right)}{\mathbb{P}\left(U < \frac{f(Y)}{c \cdot g(Y)}\right)} \\
&= \int_{-\infty}^{x} \left(\int_0^{f(y)/cg(y)} \mathrm{d}u\right) g(y)\,\mathrm{d}y \Bigg/ \int_{-\infty}^{\infty} \left(\int_0^{f(y)/cg(y)} \mathrm{d}u\right) g(y)\,\mathrm{d}y \\
&= \int_{-\infty}^{x} f(y)\,\mathrm{d}y \Big/ \int_{-\infty}^{\infty} f(y)\,\mathrm{d}y \\
&= F_X(x)
\end{aligned}
$$

Rejection Sampling (Continued)

- It is easy to see that the probability of accepting a draw is exactly $1/c$
- So for an efficient rejection sampler, we want c to be as small as possible
- Of course, this requires a careful choice of g
- In theory, the optimal choice of g is f, but of course this is unattainable
- But the “closer” g is to f, the better

Rejection Sampling: Example

```r
set.seed(2311)

n <- 1000

f <- function(x) {2*dnorm(x)*(x >= 0)}
g <- function(x) {dexp(x)}
c <- sqrt(2*exp(1)/pi)

Y <- rexp(n=n)
u <- runif(n=n)

X <- Y*(u < f(Y)/(c*g(Y)))
X <- X[X > 0]

Z <- ifelse(runif(n=length(X)) < 0.5, 1, -1)*X

hist(Z)

ks.test(Z, "pnorm")
```

Monte Carlo Integration

- Monte Carlo can be used to estimate definite integrals $\int_a^b g(x)\,\mathrm{d}x$ that may be difficult or impossible to evaluate analytically
- Idea: manipulate the integral so that it is equal to $\int_c^d h(x)\cdot f(x)\,\mathrm{d}x$ for some easily-sampled density f supported on (c,d)
- Then the LLN gives

$$\frac{1}{n}\sum_{i=1}^{n} h(X_i) \approx \mathbb{E}[h(X)] = \int_c^d h(x)\cdot f(x)\,\mathrm{d}x = \int_a^b g(x)\,\mathrm{d}x$$

  where $X, X_1, \ldots, X_n \overset{iid}{\sim} f$

- It's okay if the original integral and/or the transformed one is improper!

Monte Carlo Integration: Example

- Consider approximating

$$I = \int_0^\infty \frac{\log(x)}{4x^2+1}\,\mathrm{d}x$$

- We can view $\frac{1}{4x^2+1}$ as an unnormalized density on $(0, \infty)$
- Basic calculus gives $\int \frac{1}{4x^2+1}\,\mathrm{d}x = \frac{\tan^{-1}(2x)}{2} + c$, so $\int_0^\infty \frac{1}{4x^2+1}\,\mathrm{d}x = \frac{\pi}{4}$ and $f(x) = \frac{4}{\pi(4x^2+1)}$ is a density on $(0, \infty)$
- So

$$I = \frac{\pi}{4} \cdot \mathbb{E}[h(X)] \approx \frac{\pi}{4n} \sum_{i=1}^{n} h(X_i)$$

where $h(x) = \log(x)$ and $X, X_1, \ldots, X_n \overset{iid}{\sim} f$

Monte Carlo Integration: Example (Continued)

- How do we sample from f? We can use the inverse cdf method!
- The antiderivative from before gives $F(x) = \frac{2\tan^{-1}(2x)}{\pi} \cdot \mathbb{1}_{x \geq 0}$, and easy algebra gives $F^{-1}(y) = \frac{1}{2}\tan\left(\frac{\pi y}{2}\right)$, which is all we need:

Monte Carlo Integration: Example (Continued)

```r
set.seed(2311)
M <- 1000
I=rep(0,M)
for(i in 1:M){
u <- runif(n=10000)
x <- tan(pi*u/2)/2

I[i] <- (pi/4)*mean(log(x))
}
round(mean(I),3)
```

```
## [1] -0.544
```

```r
round(sqrt(var(I)),3)
```

```
## [1] 0.012
```

- In fact, one can show using contour integration that

$$I = -\frac{\pi \cdot \log(2)}{4} = -0.5443965\ldots$$

Section 4

Variance Reduction and Swindles

Monte Carlo Variance

- In all of the examples above, we have used the simulated X_i's to construct an estimator $\tilde{I}_n = \tilde{I}_n(\boldsymbol{X})$ of the integral $I = \mathbb{E}_f[h]$ we sought to estimate
- As with any statistical estimator, $\tilde{I}_n$ is a random variable with its own variance
- We call this the *Monte Carlo variance* and, naturally, we would like it to be small for fixed n
- This is especially true when $\tilde{I}_n$ is an unbiased estimator of I

Monte Carlo Variance (Continued)

- When $\tilde{I}_n = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$ for $X_1, \ldots, X_n \overset{iid}{\sim} f$, we get that

$$\begin{aligned}\mathrm{Var}(\tilde{I}_n) &= \frac{1}{n}\mathrm{Var}(h(X_i)) \\ &= \frac{1}{n}\left(\mathbb{E}\left[h(X_i)^2\right] - \mathbb{E}[h(X_i)]^2\right) \\ &= \frac{1}{n}\left(\int h(x)^2 \cdot f(x)\,\mathrm{d}x - I^2\right)\end{aligned}$$

- What if we could replace $\tilde{I}_n$ with some other unbiased estimator $\frac{1}{n}\sum_{i=1}^{n} g(X_i)$ with a lower variance?
  - That is, we want $\int g(x)^2 \cdot f(x)\,\mathrm{d}x < \int h(x)^2 \cdot f(x)\,\mathrm{d}x$
- It turns out that we can!
- Such techniques are sometimes known as *Monte Carlo swindles*

Control Variates

- Suppose we know some non-constant function k such that $k(X_i)$ is an unbiased estimator of 0
  - Clearly it's enough to know that $\mathbb{E}[k(X_i)] = c$ for some k and some c, for then we can replace $k(x)$ by $k(x) - c$

- Then for any $\lambda \in \mathbb{R}$, the estimator

$$\tilde{I}_{\text{c.v.}} = \frac{1}{n}\sum_{i=1}^{n}(h(X_i) - \lambda \cdot k(X_i))$$

is still unbiased for I, and its variance is given by

$$\mathrm{Var}(\tilde{I}_{\text{c.v.}}) = \frac{1}{n}\Big(\mathrm{Var}(h(X_i)) + \lambda^2 \cdot \mathrm{Var}(k(X_i)) - 2\lambda \cdot \mathrm{Cov}(h(X_i), k(X_i))\Big)$$

Control Variates (Continued)

- Which λ minimizes the variance? Easy calculus gives the optimal value as
$$\lambda^* = \frac{\mathrm{Cov}(h(X_i), k(X_i))}{\mathrm{Var}(k(X_i))}$$

- Plugging this in gives
$$\begin{aligned}\mathrm{Var}(\tilde{I}_{\text{c.v.}}) &= \frac{1}{n}\mathrm{Var}(h(X_i)) \cdot \left(1 - \mathrm{Corr}(h(X_i), k(X_i))^2\right) \\ &= \mathrm{Var}(\tilde{I}_n) \cdot \left(1 - \mathrm{Corr}(h(X_i), k(X_i))^2\right) \\ &\leq \mathrm{Var}(\tilde{I}_n)\end{aligned}$$

- In theory, a k such that $\mathrm{Corr}(h(X_i), k(X_i))^2 = 1$ would give the "perfect" estimator, but that can only happen when $k(x) = \pm(h(x) - I)$
    - But I is the very thing we're trying to estimate!

Control Variates (Continued)

- Fortunately, we can use Monte Carlo to estimate λ^* via

$$\lambda^* \approx \frac{\frac{1}{n(n-1)} \sum_{i=1}^{n} \left(h(X_i) - \overline{h(X)}\right) \cdot \left(k(X_i) - \overline{k(X)}\right)}{\frac{1}{n(n-1)} \sum_{i=1}^{n} \left(k(X_i) - \overline{k(X)}\right)^2}$$

$$= \frac{\sum_{i=1}^{n} \left(h(X_i) - \overline{h(X)}\right) \cdot \left(k(X_i) - \overline{k(X)}\right)}{\sum_{i=1}^{n} \left(k(X_i) - \overline{k(X)}\right)^2}$$

- In fact, this is equivalent to the OLS estimator for β_1 if $h(X_i) = \beta_0 + \beta_1 k(X_i) + \varepsilon_i$

Control Variates: Example

- Again consider approximating

$$I = \int_0^\infty \frac{\log(x)}{4x^2+1}\,\mathrm{d}x = \frac{\pi}{4}\cdot\mathbb{E}[h(X)]$$

where $X \sim f(x) = \frac{4}{\pi(4x^2+1)}$ on $(0,\infty)$ and $h(x) = \log(x)$

- It is easy to show that $\int_0^\infty \frac{4}{\pi(4x^2+1)^2}\,\mathrm{d}x = \frac{1}{2}$, so we take $k(x) = \frac{1}{4x^2+1} - \frac{1}{2}$ and

$$\tilde{I}_{\text{c.v.}} = \frac{\pi}{4n}\sum_{i=1}^{n}(h(X_i) - \lambda^* \cdot k(X_i))$$

Control Variates: Example (Continued)

```r
set.seed(2311)
M=1000
I.CV=rep(0,M)
for(i in 1:M){
u <- runif(n=10000)
x <- tan(pi*u/2)/2
kx <- 1/(4*x^2 + 1) - 1/2
hx <- log(x)

lambda.star <- lm(hx ~ kx)$coefficients[2]

I.CV[i] <- (pi/4)*mean(hx - lambda.star*kx)}
round(mean(I.CV),3)
```

```
## [1] -0.544
```

```r
round(sqrt(var(I.CV)),3)
```

```
## [1] 0.006
```

Antithetic Variates

- Suppose that the function $h : \mathbb{R} \to \mathbb{R}$ in $I = \mathbb{E}[h(X)]$ is monotone
- Furthermore, suppose that we have *two* unbiased estimators $\tilde{I}^{(1)}$ and $\tilde{I}^{(2)}$ of I which are identically distributed but negatively correlated, with $\mathrm{Var}(\tilde{I}^{(1)}) = \mathrm{Var}(\tilde{I}^{(2)}) \leq \mathrm{Var}(\tilde{I}_n)$
- If we form the estimator
$$\tilde{I}_{\text{a.v.}} = \frac{\tilde{I}^{(1)} + \tilde{I}^{(2)}}{2},$$
then $\tilde{I}_{\text{a.v.}}$ is clearly unbiased for I
- Moreover,
$$\begin{aligned}\mathrm{Var}(\tilde{I}_{\text{a.v.}}) &= \frac{\mathrm{Var}(\tilde{I}^{(1)}) + \mathrm{Var}(\tilde{I}^{(2)})}{4} + \frac{\mathrm{Cov}(\tilde{I}^{(1)}, \tilde{I}^{(2)})}{2} \\ &= \frac{(1 + \mathrm{Corr}(\tilde{I}^{(1)}, \tilde{I}^{(2)}))}{2} \cdot \mathrm{Var}(\tilde{I}_n) \\ &< \mathrm{Var}(\tilde{I}_n)\end{aligned}$$

Antithetic Variates (Continued)

- How do we choose $\tilde{I}^{(1)}$ and $\tilde{I}^{(2)}$ such that they both have the same distribution as $\tilde{I}_n = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$?
- According to the probability integral transform, X_i has the same distribution as $F^{-1}(U_i)$, where $U_i \sim \text{Unif}(0,1)$
- But $1 - U_i \sim \text{Unif}(0,1)$ too! So X_i *also* has the same distribution as $F^{-1}(1 - U_i)$
- The rest relies on a basic theorem from probability

Theorem

If $g_1, g_2 : \mathbb{R} \to \mathbb{R}$ are monotone functions, then $\mathrm{Cov}(g_1(X), g_2(X)) \geq 0$ for any random variable X.

Antithetic Variates (Continued)

- An immediate corollary is that

$$\mathrm{Corr}(h(F^{-1}(U_i)), h(F^{-1}(1-U_i))) \leq 0$$

- Thus, we take

$$\begin{aligned}\tilde{I}_{\text{a.v.}} &= \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n} h(F^{-1}(U_i)) + \frac{1}{n}\sum_{i=1}^{n} h(F^{-1}(1-U_i))\right) \\ &= \frac{1}{2n}\sum_{i=1}^{n}\Big(h(F^{-1}(U_i)) + h(F^{-1}(1-U_i))\Big)\end{aligned}$$

- In fact, the antithetic variates estimator is a special case of the control variates estimator

Antithetic Variates: Example

- Consider once again approximating

$$I = \int_0^\infty \frac{\log(x)}{4x^2+1}\,\mathrm{d}x = \frac{\pi}{4}\cdot\mathbb{E}[h(X)]$$

where $X \sim f(x) = \frac{4}{\pi(4x^2+1)}$ on $(0,\infty)$ and $h(x) = \log(x)$

- We previously found that $F^{-1}(y) = \frac{1}{2}\tan\left(\frac{\pi y}{2}\right)$
- Since h is monotone on $(0,\infty)$, we have all we need to construct the antithetic variates estimator:

$$\tilde{I}_{\text{a.v.}} = \frac{\pi}{8n}\sum_{i=1}^{n}\left(\log\left(\frac{1}{2}\tan\left(\frac{\pi U_i}{2}\right)\right) + \log\left(\frac{1}{2}\tan\left(\frac{\pi(1-U_i)}{2}\right)\right)\right),$$

where $U_1,\ldots,U_n \overset{iid}{\sim} \mathrm{Unif}(0,1)$

Antithetic Variates: Example (Continued)

```
set.seed(2311)
M=1000
I.AV=rep(0,M)

for(i in 1:M){
u <- runif(n=10000)
Finv <- function(y) {0.5*tan(pi*y/2)}

I.AV[i] <- (pi/8)*mean(log(Finv(u)) + log(Finv(1-u)))
}
round(mean(I.AV),3)
```

```
## [1] -0.544
```

```
round(sqrt(var(I.AV)),7)
```

```
## [1] 0
```

Comparing Swindles

- If we have several variance reduction techniques at our disposal, it is of interest to decide which produces an estimator with the lowest variance for a fixed Monte Carlo sample size n
- In theory, two estimators $I^{(1)}$ and $I^{(2)}$ of the same quantity I can be compared by their *relative efficiency* $\mathrm{Var}(I^{(1)})/\mathrm{Var}(I^{(1)})$
- However, a closed form for the relative efficiency is rarely available
- It is usually much easier to estimate $\mathrm{Var}(I^{(1)})$ and $\mathrm{Var}(I^{(2)})$ individually by running the simulation processes multiple times

Comparing Swindles (Continued)

```r
set.seed(2311)

M <- 1000 # number of simulations

I.vanilla <- rep(0, times=M)
I.CV <- rep(0, times=M)
I.AV <- rep(0, times=M)

for (m in 1:M) {
  u1 <- runif(n=10000)
  x1 <- tan(pi*u1/2)/2
  I.vanilla[m] <- (pi/4)*mean(log(x1))

  u2 <- runif(n=10000)
  x2 <- tan(pi*u2/2)/2
  kx <- 1/(4*x2^2 + 1) - 1/2
  hx <- log(x2)
  lambda.star <- lm(hx ~ kx)$coefficients[2]
  I.CV[m] <- (pi/4)*mean(hx - lambda.star*kx)
```

Comparing Swindles (Continued)

```r
  u3 <- runif(n=10000)
  Finv <- function(y) {0.5*tan(pi*y/2)}
  I.AV[m] <- (pi/8)*mean(log(Finv(u3)) + log(Finv(1-u3)))
}

cat("Vanilla Monte Carlo... MC mean: ", round(mean(I.vanilla),5),
    "; MC SE: ", round(sqrt(var(I.vanilla)),5), sep="")
cat("Control variates... MC mean: ",round(mean(I.CV),5),
    "; MC SE: ",round(sqrt(var(I.CV)),5), sep="")
cat("Antithetic variates... MC mean: ", round(mean(I.AV),5),
    "; MC SE: ",round(sqrt(var(I.AV)),5), sep="")
```

```
## Vanilla Monte Carlo... MC mean: -0.54402; MC SE: 0.01216

## Control variates... MC mean: -0.54445; MC SE: 0.00514

## Antithetic variates... MC mean: -0.5444; MC SE: 0
```

$$I = -\frac{\pi \cdot \log(2)}{4} = -0.5443965\ldots$$

Importance Sampling

- Suppose the goal is to estimate $\mathbb{E}_f[h] := \mathbb{E}[h(X)] = \int h(x) \cdot f(x) \, \mathrm{d}x$ but f is hard to sample from directly
- Suppose also that g is easier to sample from and $\operatorname{Supp} f \subseteq \operatorname{Supp} g$
- If $Y_1, Y_2, \ldots, Y_n \overset{iid}{\sim} g$, then

$$\frac{1}{n} \sum_{i=1}^{n} h(Y_i) \cdot \frac{f(Y_i)}{g(Y_i)} \to \mathbb{E}_g \left[h(Y) \cdot \frac{f(Y)}{g(Y)} \right] = \mathbb{E}_f[h].$$

- Observe that if $f = g$, we get back the standard estimator of the mean
- The more f and g "differ", the more variance we introduce into the estimator
    - We will make this precise later

Importance Sampling: Example

```r
set.seed(2311)

n <- 10000

f <- function(x) {1/(4*cosh((x-5)/2)^2)}
g <- function(x) {dnorm(x, mean=5)}

hY <- sin(rnorm(n=n, mean=5)-5)^3

mu.est <- mean(hY*f(Y)/g(Y))

# actual value: 0 (prove!)
```

What About the Normalizing Constant?

- Often, we can only evaluate the target density f up to a constant (see Bayesian posteriors)
  - i.e., $f(x) = c \cdot q(x)$ where $q(x)$ is known but $c = \left(\int q(x)\,\mathrm{d}x\right)^{-1}$ is not
- Importance sampling can be adapted to work in this situation
- If $Y_1, \ldots, Y_n \overset{iid}{\sim} g$, let $w^*(Y_i) = q(Y_i)/g(Y_i)$ and define the weights $w(Y_i) = w^*(Y_i)/\sum_{i=1}^n w^*(Y_i)$
- Then two importance sampling approximations yield

$$\frac{1}{n}\sum_{i=1}^n h(Y_i)\cdot w(Y_i) = \frac{\frac{1}{n}\sum_{i=1}^n h(Y_i)\cdot q(Y_i)/g(Y_i)}{\frac{1}{n}\sum_{i=1}^n q(Y_i)/g(Y_i)} \approx \frac{\int h(y)\cdot q(y)\,\mathrm{d}y}{\int q(y)\,\mathrm{d}y}$$

- The last term is simply $\int h(y)\cdot c\cdot q(y)\,\mathrm{d}y = \int h(y)\cdot f(y)\,\mathrm{d}y = \mathbb{E}_f[h]$

Other Methods

- There are other simple methods for reducing variance of Monte Carlo estimators
- For example, Rao-Blackwellizing an estimator automatically decreases its variance, and in certain situations one can apply the same principle to Monte Carlo
  - See Rob's STA261 slides for more info on Rao-Blackwellization
- Importance sampling itself can be thought of a variance reduction method if one chooses the "easier" sampling density g such that $\mathbb{E}_f[h(X_i)] = \mathbb{E}_g\left[h(Y_i) \cdot \frac{f(Y_i)}{g(Y_i)}\right]$ but $\mathrm{Var}_g\left[h(Y_i) \cdot \frac{f(Y_i)}{g(Y_i)}\right] < \mathrm{Var}_f[h(X_i)]$