Lab 11

November 30, 2016

This is an R Markdown Notebook.

More on numerical stability of Importance Sampling

When calculating the likelihood for a large sample you, the likelihood will often take on astronomically small values, so numerical considerations must be taken. For example, R would conclude the quantity

$$\frac{e^{-1000}}{e^{-1000}+e^{-1001}}$$

is `NaN`, because both the numerator and denominator are both 0, as far as R is concerned.

```
exp(-1000)/(exp(-1000)+exp(-1001))
```

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

However, we know this quantity is equal to $1/(1+e^{-1}) = .731$.

```
1/(1+exp(-1))
```

```
## [1] 0.7310586
```

The importance function has a similar issue, since it is a ratio of two densities. It is numerically more stable to work with

$$\log(\tilde{w}(x)) = \log(\tilde{\pi}(x)) - \log(g(x))$$

This is necessary when the sample size is relatively large. Let

$$M = \max_i \log(\tilde{w}(Xi))$$

we can derive the new estimator:

$$\begin{aligned}
\overline{\pi_{n,IS}}(h) &= \frac{\sum_{k=1}^n h(X_k)\tilde{\omega}(X_k)}{\sum_{k=1}^n \tilde{\omega}(X_k)} \\
&= \frac{\sum_{k=1}^n h(X_k)\exp(\log(\tilde{\omega}(X_k)))}{\sum_{k=1}^n \exp(\log(\tilde{\omega}(X_k)))} \\
&= \frac{e^M \sum_{k=1}^n h(X_k)\exp(\log(\tilde{\omega}(X_k)) - M)}{e^M \sum_{k=1}^n \exp(\log(\tilde{\omega}(X_k)) - M)} \\
&= \frac{\sum_{k=1}^n h(X_k)\exp(\log(\tilde{\omega}(X_k)) - M)}{\sum_{k=1}^n \exp(\log(\tilde{\omega}(X_k)) - M)}
\end{aligned}$$

The final line is a more numerically stable version of the importance sampling estimator.

The famed Bayes Theorem:

Bayes theorem and old western

The following is an abridged version of the same story detailed on page 155 of **The Cult of Statistical Significance**:

Dolores arrives at Sweetwater town trying to find out how Arnold died. Without a body to inspect, classical inference is of little help to her:

Classical hypothesis testing evaluates the hypothesis "Arnold was hanged" based on the likelihood of the outcome of the hypothesis. Usually it is impossible to escape death if you are hanged (unless you are in an old western), so the hypothesis is not implausible.

Then Dolores remembers he could also have caught pneumonia, or a bullet; each of those hypotheses cannot be rejected either. With a millions ways to die in the west, she is left scatching her head.

Fisher's test evaluates the probability of Arnold is dead, given that he was hanged ($P(\mathrm{D}|\mathrm{H})$). What she wants to know is the opposite: the probability of he hanged, given that he is now dead ($P(\mathrm{H}|\mathrm{D})$).

Bayes to the rescue

If she understands Bayes theorem, and some context of the world they live in, things can become easier.

In a world where people die from dehydration, cattle stampedes, bullet wounds, flu, and cardiac arrest, being dead is very weak evidence that the person was hanged. When the probability of being hanged is low compared to other ailments to start with, the probability of him dying from hanging, after knowing the outcome, can still be small.

But wait, there's more

It is not coincidental that the author of the book chose to name the hypothesis "hanging", and data "death". But we don't have to. We can in fact calculate the probability of each hypothesis.

$$P(\mathrm{H}=i|\mathrm{D}) = \frac{P(\mathrm{D}|\mathrm{H}=i)P(\mathrm{H}=i)}{P(\mathrm{D})}$$

Hence we know the ratio of the probability of any two hypothesis, ex post facto, quite easily:

$$\frac{P(\mathrm{H}=i|\mathrm{D})}{P(\mathrm{H}=j|\mathrm{D})} = \frac{P(\mathrm{D}|\mathrm{H}=i)P(\mathrm{H}=i)}{P(\mathrm{D})}\frac{P(\mathrm{D})}{P(\mathrm{D}|\mathrm{H}=j)P(\mathrm{H}=j)} = \frac{P(\mathrm{D}|\mathrm{H}=i)P(\mathrm{H}=i)}{P(\mathrm{D}|\mathrm{H}=j)P(\mathrm{H}=j)}$$

That is,

$$P(\mathrm{H}=i|\mathrm{D}) \propto P(\mathrm{D}|\mathrm{H}=i)P(\mathrm{H}=i)$$

and $P(\mathrm{D})$ is a normalizing constant that does not vary with your hypothesis.

Knowing the probability sums to 1,

$$P(\mathrm{D}) = \sum_i P(\mathrm{D}|\mathrm{H}=i)P(\mathrm{H}=i)$$

Naming conventions

We call the $P(\mathrm{H})$ the prior (prior belief, before observing the data), $P(\mathrm{H}|\mathrm{D})$ the posterior, and $P(\mathrm{D}|\mathrm{H})$ the likelihood.

Posterior is proportional to the likelihood times the prior.

Common notations:

- Θ for parameter of interest
- Y for random event observed
- $\pi(\theta)$ the prior
- $f(y|\theta)$ the likelihood
- $\pi(\theta|y)$ the posterior

Bayesian Inference

The primary examples where you want to determine properties of a distribution given only the unnormalized density come from Bayesian inference. Suppose you observe data $y_1, ..., y_n$ with density $f(y|\theta)$ indexed by a parameter θ. A typical approach is to view θ as a fixed unknown truth you want to determine and and estimate by maximum likelihood

$$\max_{\theta} f_\theta(y_1, ..., y_n)$$

Instead, a bayesian approach views the parameter as having a distribution when conditioned on the data, and look to make inference about the posterior density, $\pi(\theta|y_1, ..., y_n)$. Noting that $f_\theta(y) = f(y|\theta)$ and applying bayes rule twice:

$$\pi(\theta|y_1, ..., y_n) = \frac{p(\theta, y_1, ..., y_n)}{p(y_1, ..., y_n)} = \frac{f(y_1, ..., y_n|\theta)\pi(\theta)}{p(y_1, ..., y_n)}$$

In the case of y_i's being iid samples, as the sample size increases, the shape of the posterior is increasingly dominated by the likelihood.

The denominator, i.e., the marginal probability of the data

$$p(y_1, ..., y_n) = \int p(\theta, y_1, ..., y_n)\mathrm{d}\theta = \int f(y_1, ..., y_n|\theta)\pi(\theta)\mathrm{d}\theta$$

is the normalizing constant for the posterior distribution and is often an intractable integral. We can apply importance sampling to the unnormalized posterior density

$$f(y_1, ..., y_n|\theta)\pi(\theta)$$

Example 1

Suppose $X_1, ..., X_n \sim \text{Binomial}(10, \theta)$ where $\theta \in (0, 1)$ has a $\text{Beta}(5, 3)$ prior density:

$$\pi(\theta) = \frac{\Gamma(8)}{\Gamma(5)\Gamma(3)}\theta^4(1-\theta)^2$$

We want to estimate the mean of the posterior distribution: $\int \theta\pi(\theta|x_1, ..., x_n)\mathrm{d}\theta$. Take the (proposal) density function g in importance sampling to be $\text{Beta}(\alpha, \beta)$ density, where

$$\alpha = c\bar{X}\beta = c(10 - \bar{X})$$

where $\bar{X}$ is the sample mean, and c is a positive constant. This will ensure that g is peaked near the MLE $\bar{X}/10$, where the posterior distribution should have a lot of mass. The larger c is, the more sharply peaked around $\bar{X}/10$ will be. We will tune c to minimize the variance of the estimate.

Likelihood:

$$
\begin{aligned}
f(x_1, ..., x_n|\theta) &= \prod_{i=1}^{n} f(x_i|\theta) \\
&= \prod_{i=1}^{n} \binom{10}{x_i} \theta^{x_i}(1-\theta)^{10-x_i} \\
&\propto \theta^{\sum x_i}(1-\theta)^{10-\sum x_i} \\
&= \theta^{n\bar{x}}(1-\theta)^{n(10-\bar{x})}
\end{aligned}
$$

Posterior:

$$
\begin{aligned}
\pi(\theta|x_1, ..., x_n) &\propto f(x_1, ..., x_n|\theta)\pi(\theta) \\
&\propto \theta^{n\bar{x}}(1-\theta)^{n(10-\bar{x})}\pi(\theta) \\
&= \theta^{n\bar{x}}(1-\theta)^{n(10-\bar{x})}\frac{\Gamma(8)}{\Gamma(5)\Gamma(3)}\theta^4(1-\theta)^2 \\
&\propto \theta^{n\bar{x}}(1-\theta)^{n(10-\bar{x})}\theta^4(1-\theta)^2 \\
&= \theta^{n\bar{x}+4}(1-\theta)^{n(10-\bar{x})+2}
\end{aligned}
$$

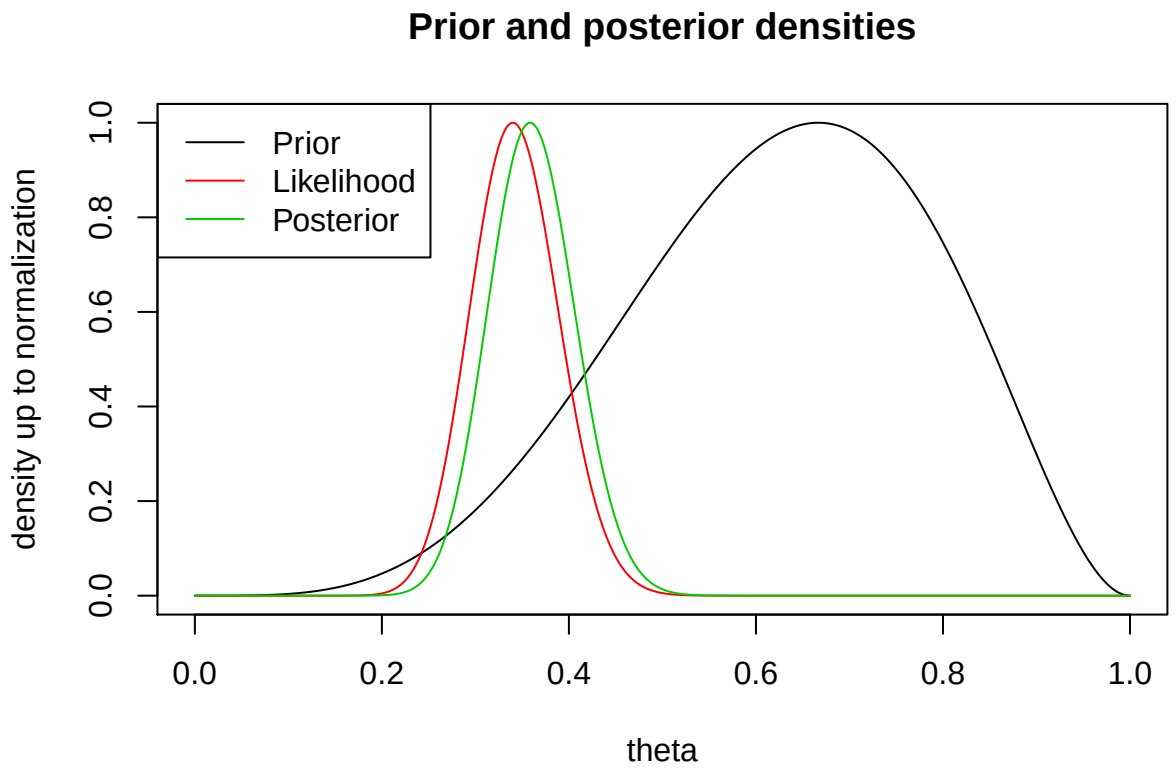
The log of this quantity is

$$(n\bar{x}+4)\log(\theta) + (n(10-\bar{x})+2)\log(1-\theta)$$

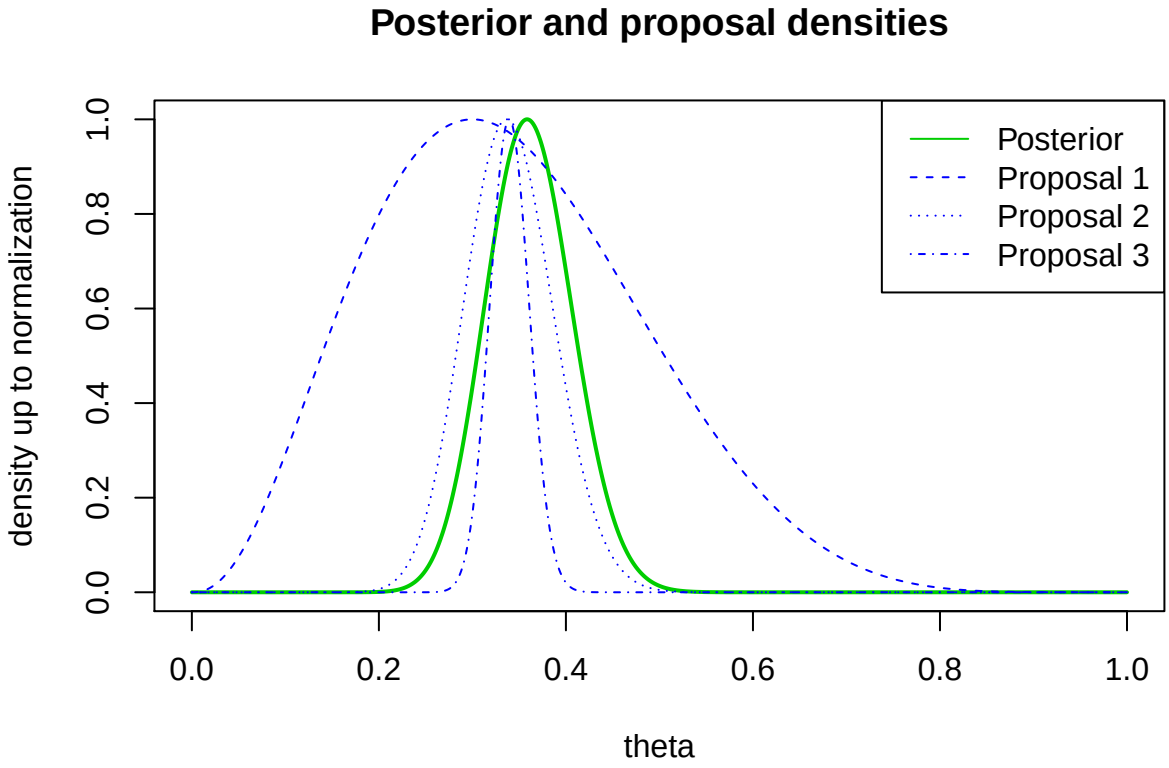
The following code illuistrate choices of different trial/proposal densities to approximate posterior.

```
log.prior <-  function(theta) {
  4 * log(theta) + 2 * log(1 - theta)
}

log.likelihood <-  function(data, theta) {
  n <- length(data)
  sum(data) * log(theta) + n * (10 - mean(data)) * log(1 - theta)
}

log.posterior <-  function(data, theta) {
  log.prior(theta) + log.likelihood(data, theta)
}

log.g <- function(data, theta, C) {
   dbeta(theta, C * mean(data), C * (10 - mean(data)), log = TRUE)
}

data = rbinom(10, 10, .4)

theta.seq <- seq(0,1,0.001)
log.prior.eval <- log.prior(theta.seq)
log.likelihood.eval <- log.likelihood(data, theta.seq)
log.posterior.eval <- log.posterior(data, theta.seq)

plot(theta.seq, exp(log.prior.eval-max(log.prior.eval)),
     type = 'l', xlab = "theta", ylab = "density up to normalization",
     main = "Prior and posterior densities")
lines(theta.seq, exp(log.likelihood.eval-max(log.likelihood.eval)), col = 2)
lines(theta.seq, exp(log.posterior.eval-max(log.posterior.eval)), col = 3)
```

```r
legend("topleft", legend = c("Prior", "Likelihood", "Posterior"),
       lty = 1, col = c(1:3))
```




```r
C.choice1 <- 1
C.choice2 <- 10
C.choice3 <- 50
log.g1.eval <- log.g(data, theta.seq, C.choice1)
log.g2.eval <- log.g(data, theta.seq, C.choice2)
log.g3.eval <- log.g(data, theta.seq, C.choice3)

plot(theta.seq, exp(log.posterior.eval-max(log.posterior.eval)),
     type = 'l', xlab = "theta", ylab = "density up to normalization",
     main = "Posterior and proposal densities", col = 3, lwd = 2)
lines(theta.seq, exp(log.g1.eval-max(log.g1.eval)), col = 4, lty = 2)
lines(theta.seq, exp(log.g2.eval-max(log.g2.eval)), col = 4, lty = 3)
lines(theta.seq, exp(log.g3.eval-max(log.g3.eval)), col = 4, lty = 4)

legend("topright", legend = c("Posterior", "Proposal 1", "Proposal 2", "Proposal 3"),
       lty = 1:4, col = c(3,4,4,4))
```

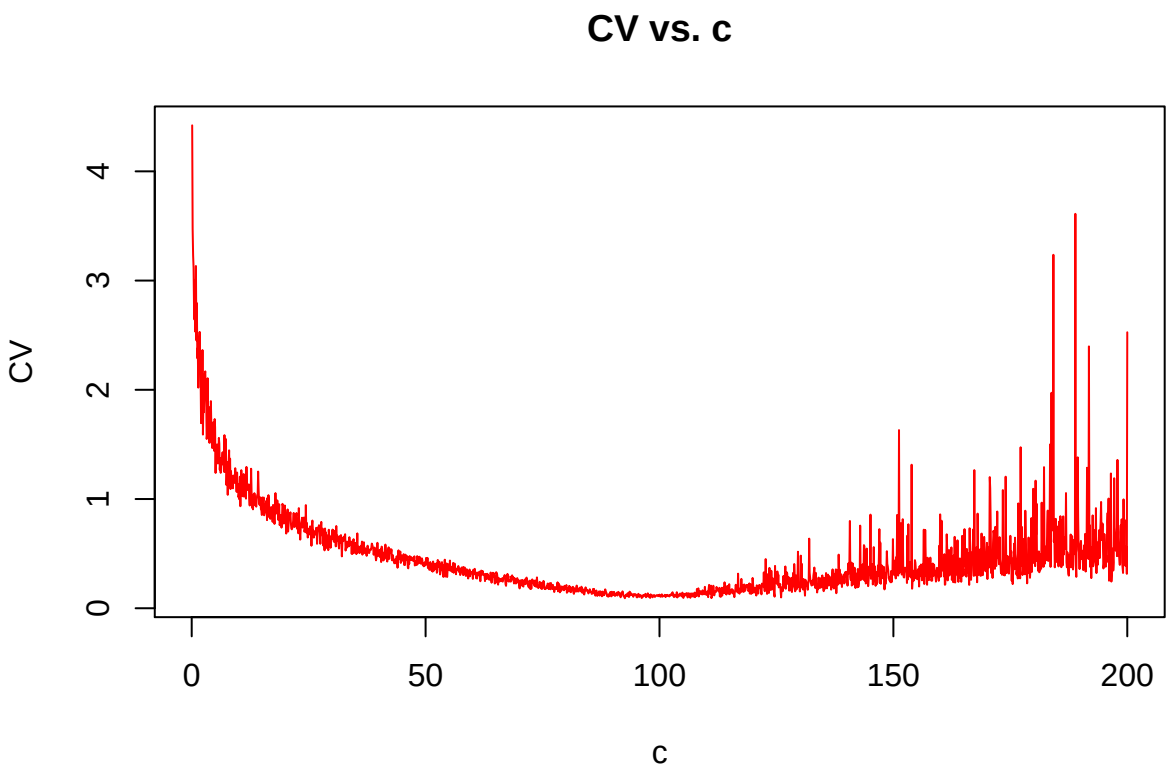


The bettter our proposal density approximates the posterior density, the less variable the importance weight $\pi(\theta|data)/g(\theta)$, hence the more accurate our importance sampling estimates will be.

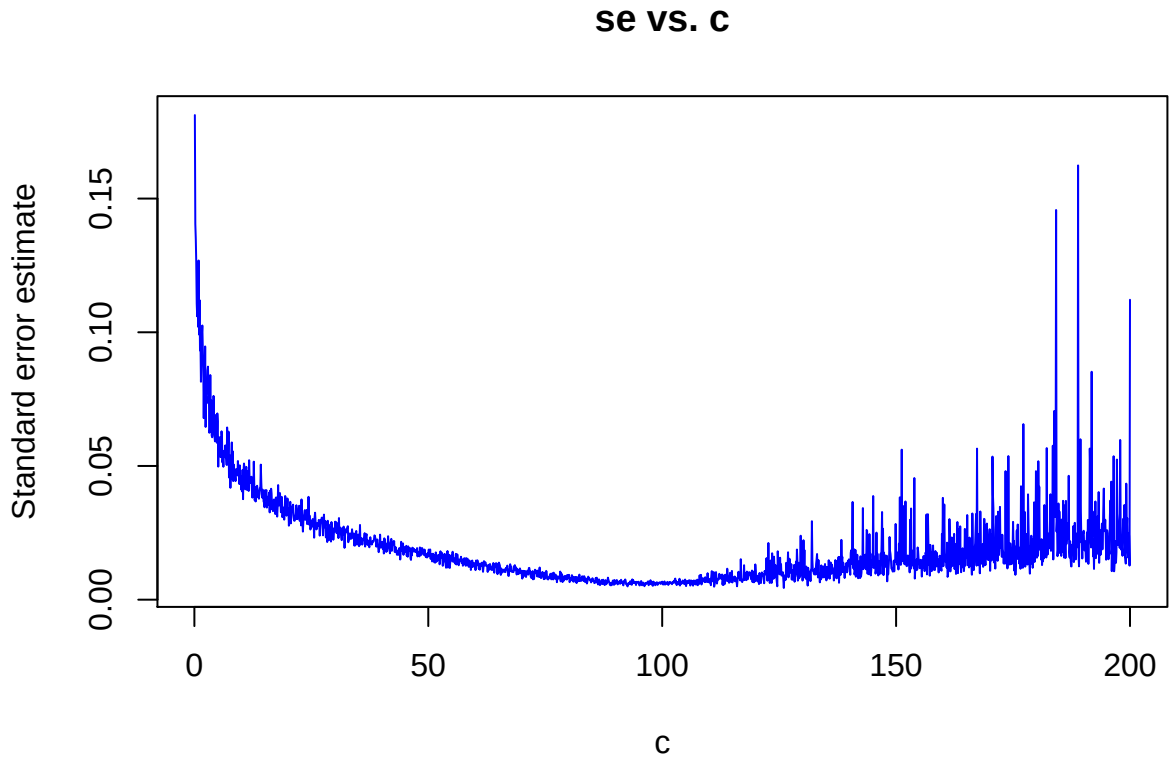
The following R code executes the importance sampling:

```
# X is the data, res is the number of monte carlo samples,
# C is the tuning parameter for the g distribution
IS <- function(X, C, n)
{
  # sample size
  n <- length(X)
  # log posterior derived above
  log.posterior <-  function(theta) {
    (sum(X) + 4) * log(theta) + (n * (10 - mean(X)) + 2) * log(1 - theta)
  }
  # log trial (proposal) density, g
  log.g <- function(theta) {
    dbeta(theta, C * mean(X), C * (10 - mean(X)), log = TRUE)
  }

  # log importance function
  log.w <- function(theta) {
    log.posterior(theta) - log.g(theta)
  }

  # generate from the trial (proposal) distribution
  U <- rbeta(n, C * mean(X), C * (10 - mean(X)))

  # calculate the list of log.w values
  LP <- log.w(U)
```

```
  # factor out the largest value to prevent numerical underflow
  w <- max(LP)
  LP <- LP - w

  # importance sampling estimate
  I <- mean(exp(LP) * U) / mean(exp(LP))

  # calculate CV
  CV <- sqrt( var( exp(LP) / mean(exp(LP) ) ) )

  # calculate s.e. of the IS estimate
  Z = exp(LP) * U / mean(exp(LP))
  sig.sq <- var( Z )
  se <- sqrt( sig.sq / n )

  # return the 95 percent confidence interval for the estimate
  return(list(lower = I - 1.96 * se, upper = I + 1.96 * se, CV = CV))
}

# Generate 100 binomial(10,.4) random variables
X = rbinom(100, 10, .4)

## calculate CV and var(E(theta|X))
## over a grid of values for alpha
C <- seq(.1, 200, length = 2000)
A <- matrix(0, 2000, 2)

for (j in 1:2000) {
  output <- IS(X, C[j], 1000)
  CV <- output$CV
  V <- (output$upper - output$lower) / 3.92
  A[j, ] <- c(CV, V)
}

# see where CV is low enough
plot(C, A[, 1], ylab = "CV", xlab = "c",
     main = "CV vs. c", col = 2, type = "l")
abline(h = 5)
```




```
# standard error estimates
plot(C, A[, 2], xlab = "c", ylab = "Standard error estimate",
     main = "se vs. c", col = 4, type = "l")
```




```
# The final confidence interval
IS(X, C[which.min(A[, 2])], 1000)[1:2]
```

```
## $lower
## [1] 0.3850031
##
## $upper
## [1] 0.4252301
```

Bootstrap

Suppose we want to estimate a parameter θ of the population distribution $\mathcal{F}$. We propose an estimator $\hat{\theta}$ based on a collected samples $\mathcal{X} = X_1, ..., X_n$.

We want know how good $\hat{\theta}$ is by evaluating MSE($\hat{\theta}$).

If we know $\mathcal{F}$, then in principle we can calculate MSE($\hat{\theta}$). But unless the distribution is easy to handle (e.g. normal), analytical formulation is intractable. Besides, we do not know $\mathcal{F}$ in practice.

Suppose we can sample from F, how to compute MSE($\hat{\theta}$)?

1. Draw samples $\mathcal{X}_1, ..., \mathcal{X}_K$, each of size n, from $\mathcal{F}_\theta$.
2. Compute sample statistic $\hat{\theta}_k$ for each sample $\mathcal{X}_k$.
3. Approximate MSE of θ by $\sum_{k=1}^{K}(\hat{\theta}_k - \theta)^2/n$.

Bootstrap: Now we don't know F, and have only one sample $\mathcal{X}$, we can mimic the above procedure

1. Draw bootstrap samples $\mathcal{X}_1^*, ..., \mathcal{X}_K^*$, each of size n, from the original sample $\mathcal{X}$.
2. Compute sample statistic $\hat{\theta}_k^*$ for each sample $\mathcal{X}_k^*$.
3. Aprroximate MSE of θ by $\sum_{k=1}^{K}(\hat{\theta}_k^* - \hat{\theta})^2/n$, where $\hat{\theta}$ is the estimate from original sample $\mathcal{X}$.

Example 1: Parametric bootstrap

Consider the following model. Given the data tuples $(y_i, x_i), i = 1, 2, ..., n$, we model the distribution of Y_i's given x_i's as as exponential distribution with parameter λx_i.

$$Y_i | x_i, \lambda \sim \text{Exponential}(\lambda x_i)$$

Notice that we treat x_i's as fixed input data values (as one does for linear regressions in usual case which is called "fixed design"") without modeling them as realizations of a random variables.

Given the maximum likelihood estimator

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} x_i y_i}$$

We are interested in the bias, variance and MSE of the maximum likelihood estimator $\hat{\lambda}$.

Now please download lab11b.Rdata and . The name of the dataframe is `sampledata`.

We first perform prametric bootstrap. Every simulation of $\hat{\lambda}_k^*$ simulate y's on all the deisgn location x's.

```
x = sampledata[,1]
y = sampledata[,2]
n = length(x)

## calculate lambda-hat
lambda.hat = n / sum( x * y )

## specify bootstrap sample size
B = 10000
```

```r
# storage for bootstrapped lambda.hat
lambda.hat.p.boot = c()

## use parametric bootstrap for y
for (i in 1:B){
    y.sample = sapply(lambda.hat*x, FUN = function(x) rexp(1,x))
    lambda.hat.p.boot[i] = n / sum( x * y.sample )
}
# bias
mean(lambda.hat.p.boot)-lambda.hat
```

```
## [1] 0.01611539
```

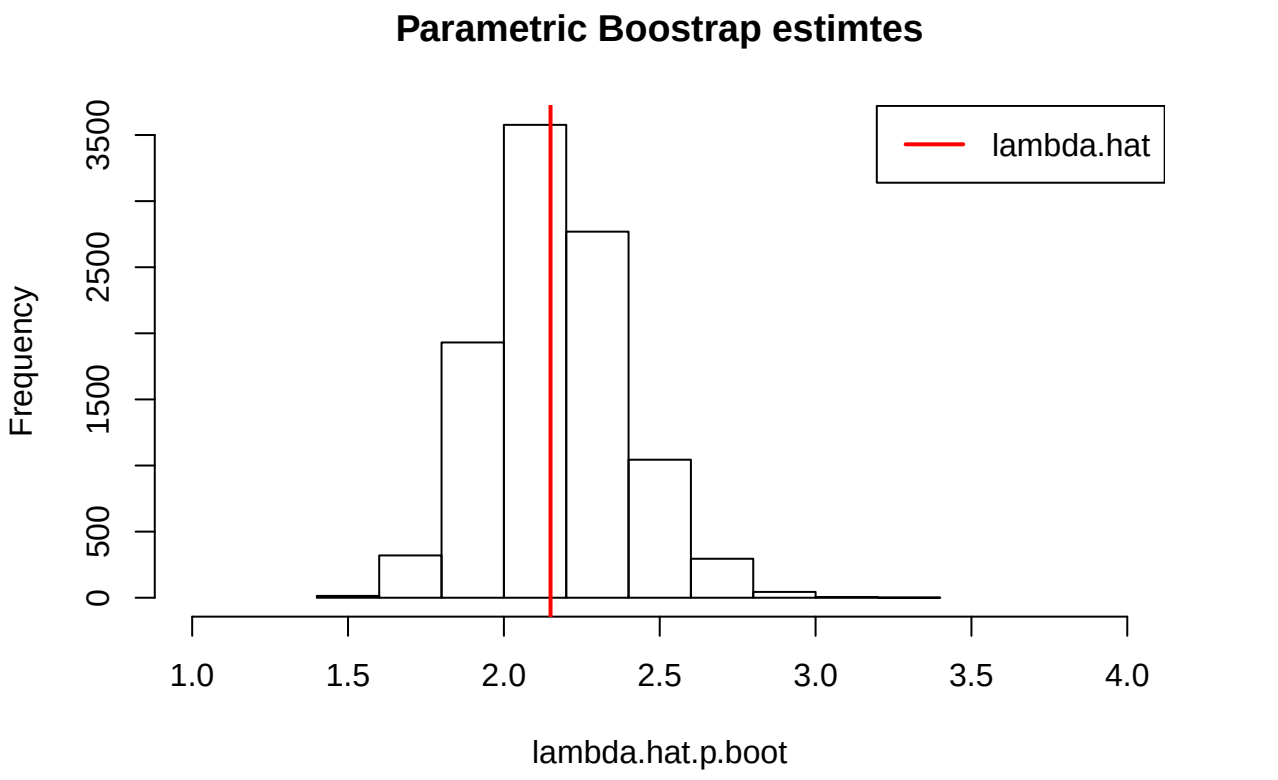
```r
# variance
var(lambda.hat.p.boot)
```

```
## [1] 0.04759474
```

```r
# MSE
mean((lambda.hat.p.boot - lambda.hat)^2)
```

```
## [1] 0.04784968
```

```r
hist(lambda.hat.p.boot, breaks = 10, xlim = c(1, 4), main = "Parametric Boostrap estimtes")
abline(v = lambda.hat, lwd = 2, col = 2)
legend("topright", legend = "lambda.hat", col = 2, lwd = 2, border = F)
```



Bias is no cause for worry; variance is large at this sample size.

Example 2: Non-parametric bootstrap

We now perform the non-parametric version of bootsreap. No new data points are simulated in this case.

```
lambda.hat.np.boot = c()
for (i in 1:B){
    sample.id = sample(n,size=n, replace=TRUE)
    lambda.hat.np.boot[i] = n/sum(x[sample.id]*y[sample.id])
}
# bias
mean(lambda.hat.np.boot)-lambda.hat
```

```
## [1] 0.01777213
```

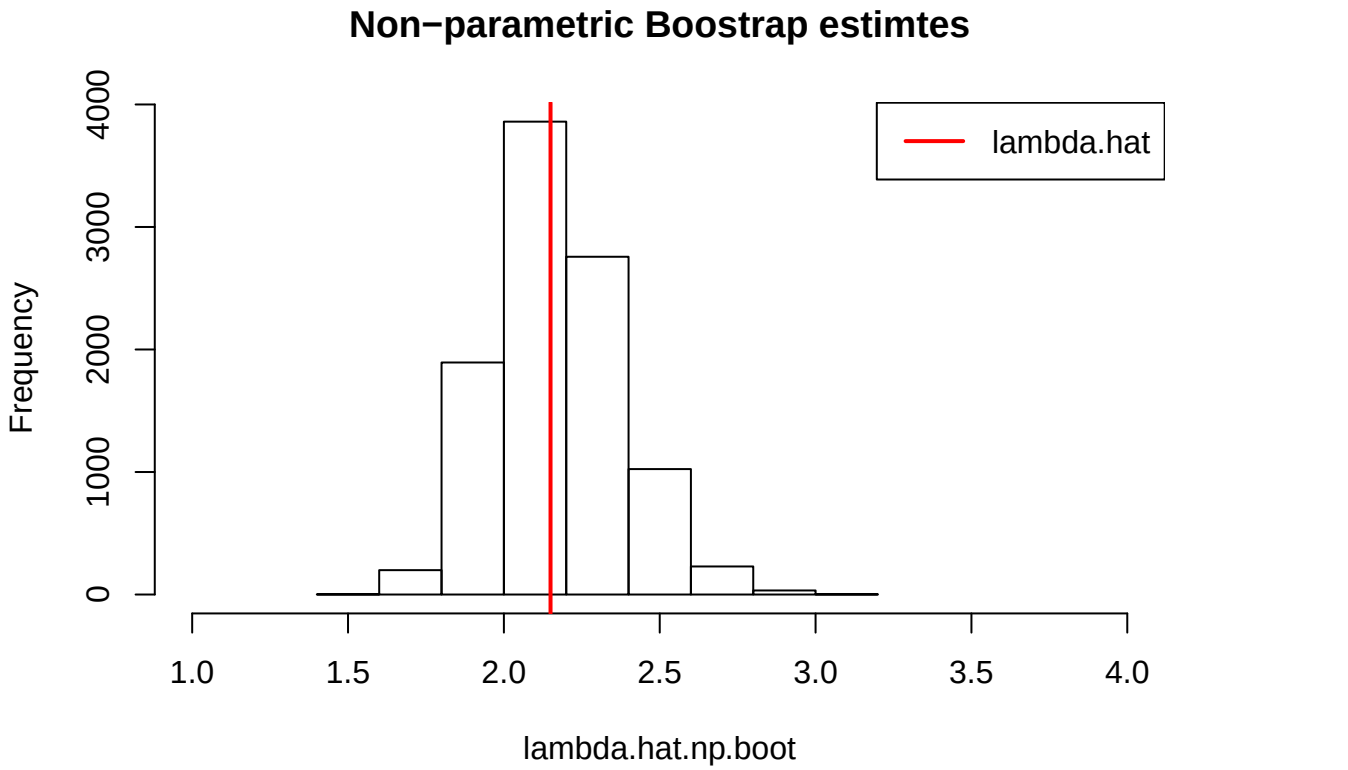
```
# variance
var(lambda.hat.np.boot)
```

```
## [1] 0.04115168
```

```
# MSE
mean((lambda.hat.np.boot - lambda.hat)^2)
```

```
## [1] 0.04146341
```

```
hist(lambda.hat.np.boot, breaks = 10, xlim = c(1, 4), main = "Non-parametric Boostrap estimtes")
abline(v = lambda.hat, lwd = 2, col = 2)
legend("topright", legend = "lambda.hat", col = 2, lwd = 2, border = F)
```



Example 3 (optional): Treating design as random

If we treat x input as realizations of random variable - i.e., random design, we *do* model x. To perform *parametric* bootstrap of $\hat{\lambda}$ using y_i condition on x_i, i = 1, ..., n, we have to use nonparametric bootstrap to sample for x's (there's no way to perform parametric bootstrap to sample x since the distribution of x is completely unspecified).

```
lambda.hat.p.np.boot = c()
for (i in 1:B){
    y.sample = c()
    x.sample = sample(x, size=n, replace=TRUE)
    y.sample = sapply(lambda.hat*x.sample, FUN = function(x) rexp(1,x))
    lambda.hat.p.np.boot[i] = n/sum(x.sample*y.sample)
}
# bias
mean(lambda.hat.p.np.boot)-lambda.hat
```

```
## [1] 0.0235726
```

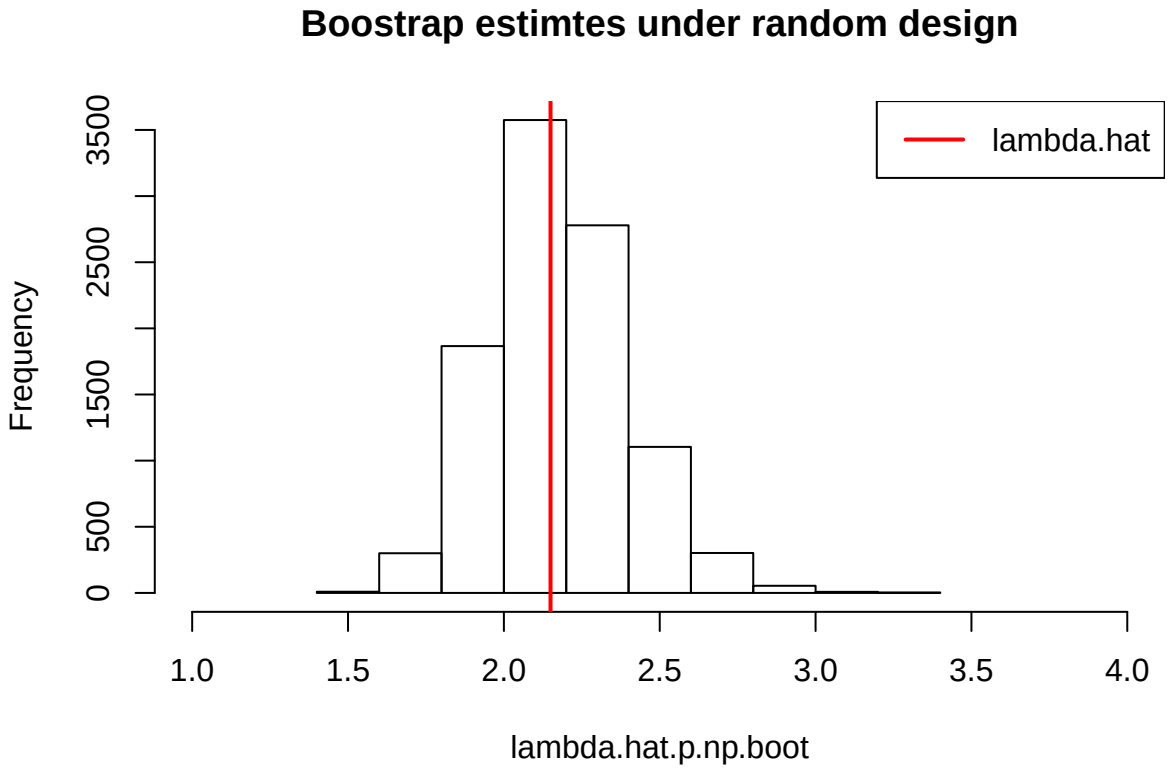
```
# variance
var(lambda.hat.p.np.boot)
```

```
## [1] 0.04827631
```

```
# MSE
mean((lambda.hat.p.np.boot - lambda.hat)^2)
```

```
## [1] 0.04882715
```

```
hist(lambda.hat.p.np.boot, breaks = 10, xlim = c(1, 4), main = "Boostrap estimtes under random design")
abline(v = lambda.hat, lwd = 2, col = 2)
legend("topright", legend = "lambda.hat", col = 2, lwd = 2, border = F)
```



All three bootstrap estimates are pretty similar in this example.

Extras

When does bootstrap work? Bootstrap works in most applications and perhaps all examples you will see in this course. The sufficient and necesary condition for bootstrap consistency is that a central limit property holds for $\hat{\theta}$ (Mamman, 1992).

See more comprehensive treatment here http://www.unc.edu/~saraswat/teaching/econ870/fall11/JH_01.pdf.