Stat 243: Building an adaptive rejection sampler

James Bladen, Lisa Felberg, Siwei Tu, Hsin-Wei Tsao

December 13, 2013

1 Introduction

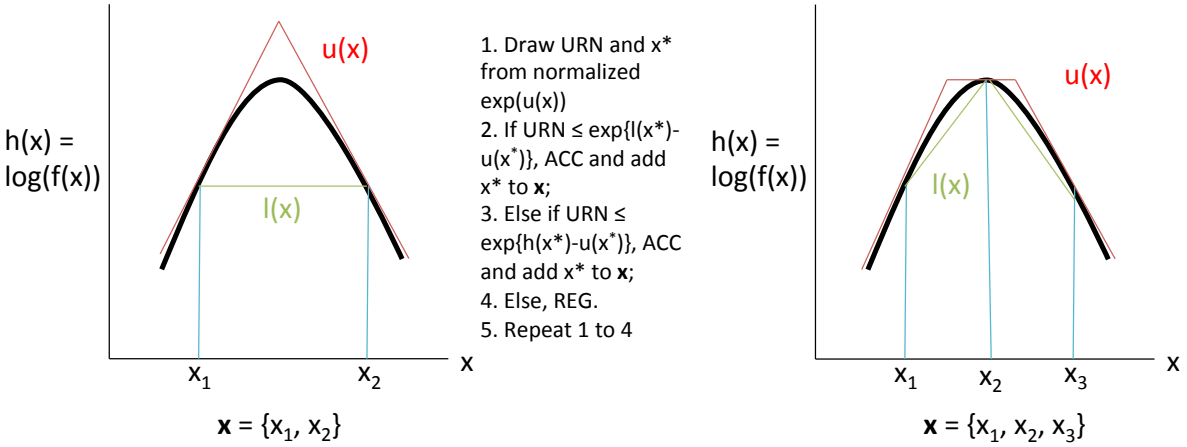
(i)

Adaptive Rejection sampling was first introduced in 1992 by Gilks and Wild. It was proposed as an alternative to vanilla rejection sampling for functions that are difficult to evaluate multiple times. The objective of this method is to sample from a "difficult" distribution by representing it with tangent lines and evaluating the actual function as few times as possible. The basic algorithm is as follows:

- Take the log of the function to sample from, $h(x) = log(f(x))$
- Select two starting points, x_1 and x_2 that are to the right and to the left of the function's maximum, respectively
- Sample two random numbers, one from the uniform random number distribution, URN, and one from the normalized exponential of the upper bounded line, x^*
- If $URN \leq exp(l(x*) - u(x*))$, accept and add x^* to vector of x's
- Else if $URN \leq exp(h(x*) - u(x*))$, accept and add x^* to vector of x's
- Else reject
- Continue to sample numbers and testing procedure until a desired number of samples is obtained

The image below is a graphical representation of the algorithm.

Figure 1: Algorithm of ARS method



2 Code Structure

We decided to utilize the S4 class as a format for our code. It was chosen because of the formal structure and modularity. These properties also made it easy to test and create in a very piecewise manner. The general structure to the code is a follows: A wrapper function *a_r_s* runs the sampling method. The program is run as follows:

$$sample \leftarrow ars(\mathbf{n}, \mathbf{f\_x}, \mathbf{bounds}, \mathbf{guess\_of\_mode})$$

where
$\mathbf{n} = \{\#\, of\ samps\, desired\}, \mathbf{f\_x} = \{f(x)\, to\, sample\}$,
$\mathbf{bounds} = \{bds\, of\ f(x)\}, \mathbf{guess\_of\_mode} = \{user's\, guess\, of\ function\, mode\}$

This will return a *Cadapt_reject_sample* class which contains sampled points in *ars@samples*. The class itself contains all the methods and variables required to perform adaptive rejection sampling.

First, two points are chosen as starting x_1 and x_2 values. Next, x^* values are drawn from the corresponding $u(x)$ distribution and the sampling criteria is tested. This is repeated until the desired number of samples is obtained. Brief descriptions of the most important methods are given below. We've included more detailed source documentation as part of our solution as well (see *a_r_sManual.pdf*).

gen_x method

In this method, we generate two starting points i.e. x_1, x_2 according to user's input of bounds and guess of mode.
There are 4 possible cases:

- If the target function is bounded on both sides, then we basically take 2 bounds as x_1, x_2.

If the target function is unbounded on either one or both sides, we make use of user's guess of mode to find the maximum of target function $f(x)$ (In theory, it should be the real mode.) by applying $optimize(\,)$ and use $(mode - 100, mode + 100)$ as bounds in the optimize function. The step is to find the appropriate starting points more quickly.

- If the target function is unbounded on both sides, we set $x_1 = maxf(x) - 0.1$. If $h'(x_1) > 0$ then find an x_2 on the right side of x_1 s.t. $h'(x_2) < 0$. Similary, if $h'(x_1) < 0$ then find an x_2 on the left side of x_1 s.t. $h'(x_2) > 0$. In addition, if $h'(x_1) = 0$ which means $h(x)$ is symmetric to x_10, then we set $(x_1 - \frac{1}{2}, x_1 + \frac{1}{2})$ to be 2 starting points.
- If the target function is bounded only on left side, and if the bound is less than $maxf(x)$, then we use the $maxf(x)$ as x_1, and find an x_2 on the right side of $maxf(x)$ s.t. $h'(x_1)h'(x_2) < 0$. Or, if the bound is greater than $maxf(x)$, we just use the bound as as x_1, and find an x_2 on the right side of $maxf(x)$ s.t. $h'(x_1)h'(x_2) < 0$.
- If the target function is bounded only on right side, and if the bound is greater than $maxf(x)$, then we use the $maxf(x)$ as x_1, and find an x_2 on the right side of $maxf(x)$ s.t. $h'(x_1)h'(x_2) < 0$. Or, if the bound is smaller than $maxf(x)$, we just use the bound as as x_1, and find an x_2 on the right side of $maxf(x)$ s.t. $h'(x_1)h'(x_2) < 0$.

Note that $h(x) = log(f(x))$ which is concave. So for any x_1, x_2, if $h'(x_1)h'(x_2) < 0$, x_1 and x_2 would be on the different side of maximum of $h(x)$. And with the options of 2 to 4, we save x, $h(x)$ and $h'(x)$ for all x's we computed in the process for later use.

ev_h method

In this method, we calculate the values of $h(x)$ and $h'(x)$ for any x with $genD(\,)$ in package $numDeriv$.

s_x method

In this method, we calculate $s_k(x)$ by normalizing $e^{u_k(x)}$. To normalize the $e^{u_k(x)}$, we integrate each piece of $u(x)$ with following algorithm:
First note that $u(x)$ is a piecewise defined function. It is linear in each piece of interval. Actually for $x \in [z_{j-1}, z_j]$, $u(x) = ax + b$, where $a = h'(x_j)$ and $b = h(x_j) - x_j h'(x_j)$.
So for $x \in [z_{j-1}, z_j]$,

$$\int_{z_{j-1}}^{z_j} e^{u(x)} dx = \frac{1}{h'(x_j)} exp(h'(x_j)x + h(x_j) - x_j h'(x_j))\Big|_{z_{j-1}}^{z_j},$$

and

$$\int_D e^{u_k(x')} dx' = \sum_j \frac{1}{h'(x_j)} exp(h'(x_j)x + h(x_j) - x_j h'(x_j))\Big|_{z_{j-1}}^{z_j}.$$

In the special case of $a = 0$ which means that for $x \in [z_{j-1}, z_j]$, $u(x) = b$ where $b = h(x_j)$.
Then

$$\int_{z_{j-1}}^{z_j} e^{u(x)} dx = e^b x_j\Big|_{z_{j-1}}^{z_j},$$

and,

$$\int_D e^{u_k(x')} dx' = \sum_j e^b x_j\Big|_{z_{j-1}}^{z_j}.$$

Moreover, if $f(x)$ is not a log-concave distribution, there will be some negative piecewise intergations in certain intervals. In this case, we just break the program and tell user that "function is not log concave".

sampling method

In this method, we sample a new x^* from the $s_k(x)$ by using inverse CDF of $s_k(x)$ with a random value $u \sim unif(0, 1)$. Moreover, the inverse CDF algorithm we use is as follow:
Once we decided which interval x^* falls in, we want to sample from $\frac{e^{u(x)}}{\int e^{u(x)} dx}$. Suppose $x^* \in [z_{j-1}, z_j]$, then $u(x) = ax + b$, where $a = h'(x_j)(\neq 0)$ and $b = h(x_j) - x_j h'(x_j)$.
Let

$$C = \int_{z_{j-1}}^{z_j} e^{u(x)} dx,$$

then the pdf is

$$\frac{e^{ax+b}}{C}.$$

So the CDF

$$F'(x) = \int_{z_{j-1}}^{x'} \frac{e^{ax+b}}{C} dx = \frac{1}{ca} e^{ax+b}\Big|_{z_{j-1}}^{x'} = \frac{ca}{e^b}(e^{ax'} - e^{az_{j-1}}).$$

Then the inverse of $F(x')$ will be

$$\frac{\log\left(\frac{ac}{e^b} + e^{az_{j-1}}\right)}{a}.$$

In the special case of $a = 0$ which means that for $x^* \in [z_{j-1}, z_j]$, $u(x) = b$ where $b = h(x_j)$.
Then the pdf will be be $\frac{e^b}{c}x$ which is an uniform distribution in $[z_{j-1}, z_j]$. In this situation, we just sample x^* from $unif \sim (z_{j-1}, z_j)$.

upper method

In this method, we calculate the $u_k(x)$ for x^* which we sample from the sampling method.

lower method

In this method, we calculate the $l_k(x)$ for x^* which we sample from the sampling method.

update method

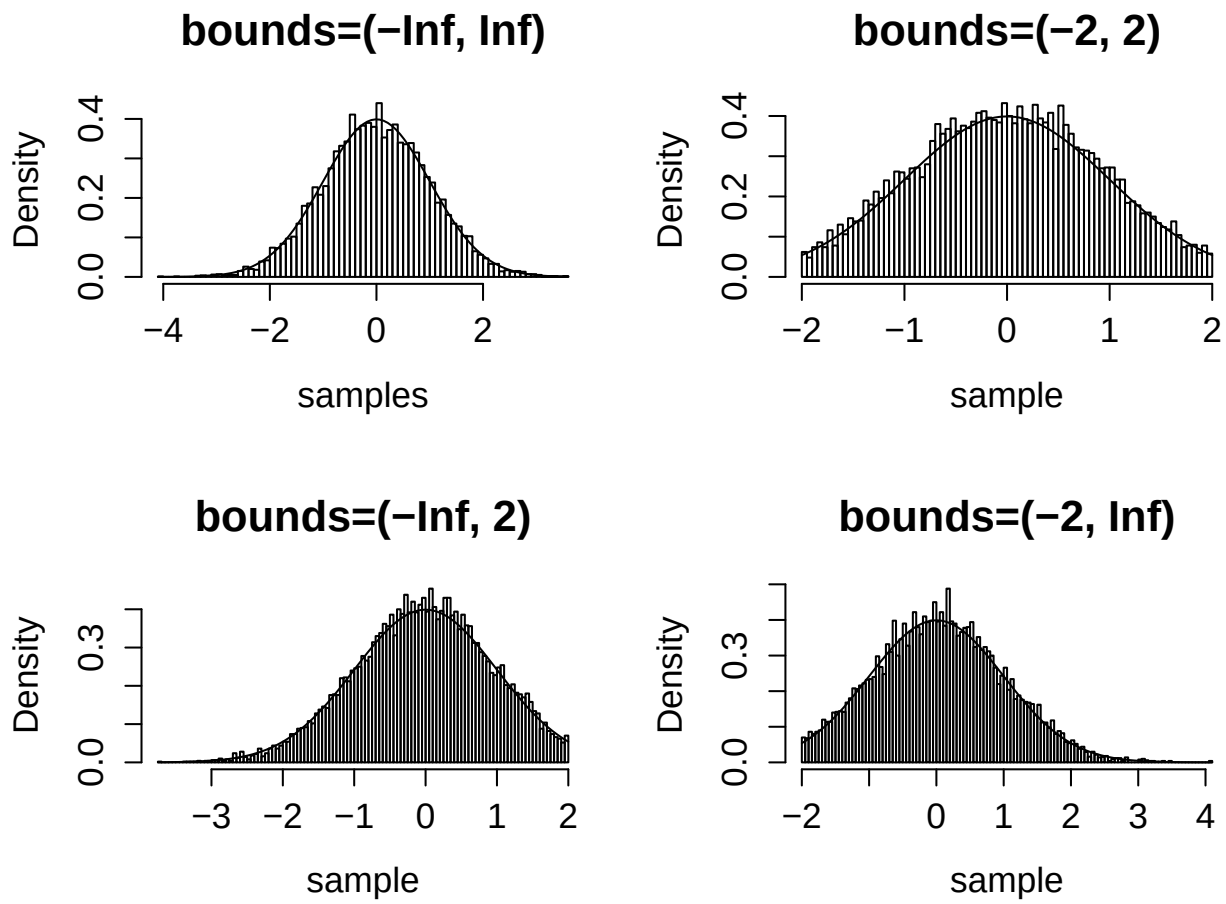
In this method, we test the sample element x^* from sampling method. Once it is accepted, we add the x^* into the set of outputs.

3 Testing

For testing, we rigorously tested each method to ensure it was functioning properly. To test the overall function, we sampled over a few well known distributions with different bounds, including: normal, gamma, truncated normal. The results are shown below.
First we test the standard normal with varying bounds , the code to do so and the histogram of the resulting samples are given below:

```
samples <- ars(n=10000,fx=function(x){(1/sqrt(2*pi)*exp((-(x-0)^2)/2))},
    bounds=c(-Inf, Inf) )
sample<-ars(10000,function(x){(1/sqrt(2*pi)*exp((-(x-0)^2)/2))},c(-2, 2))
sample<-ars(10000,function(x){(1/sqrt(2*pi)*exp((-(x-0)^2)/2))},c(-Inf,
    2))
sample<-ars(10000,function(x){(1/sqrt(2*pi)*exp((-(x-0)^2)/2))},c(-2, Inf
    ))
```

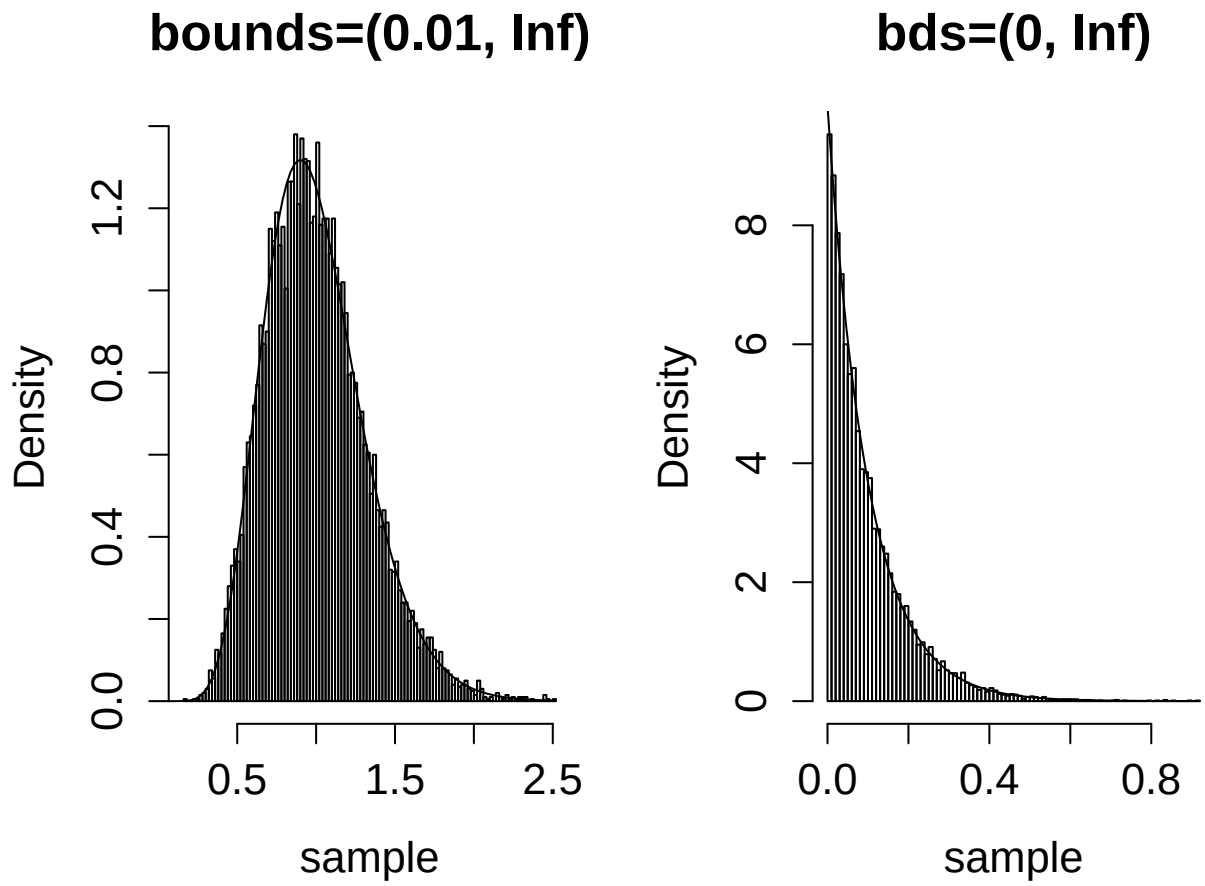
Figure 2: Histogram of samples from $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ with different bounds



Second we test the gamma(10,10) with a right bound 0.01 and the exponential(10) with a right bound 0, the code to do so and the histogram of the resulting samples are given below:

```
sample<-ars(10000,function(x){(10^10)/gamma(10)*x^9*exp(-10*x)},c(0.01,
    Inf))
sample<-ars(10000,function(x){10*exp(-10*x)},c(0, Inf))
```

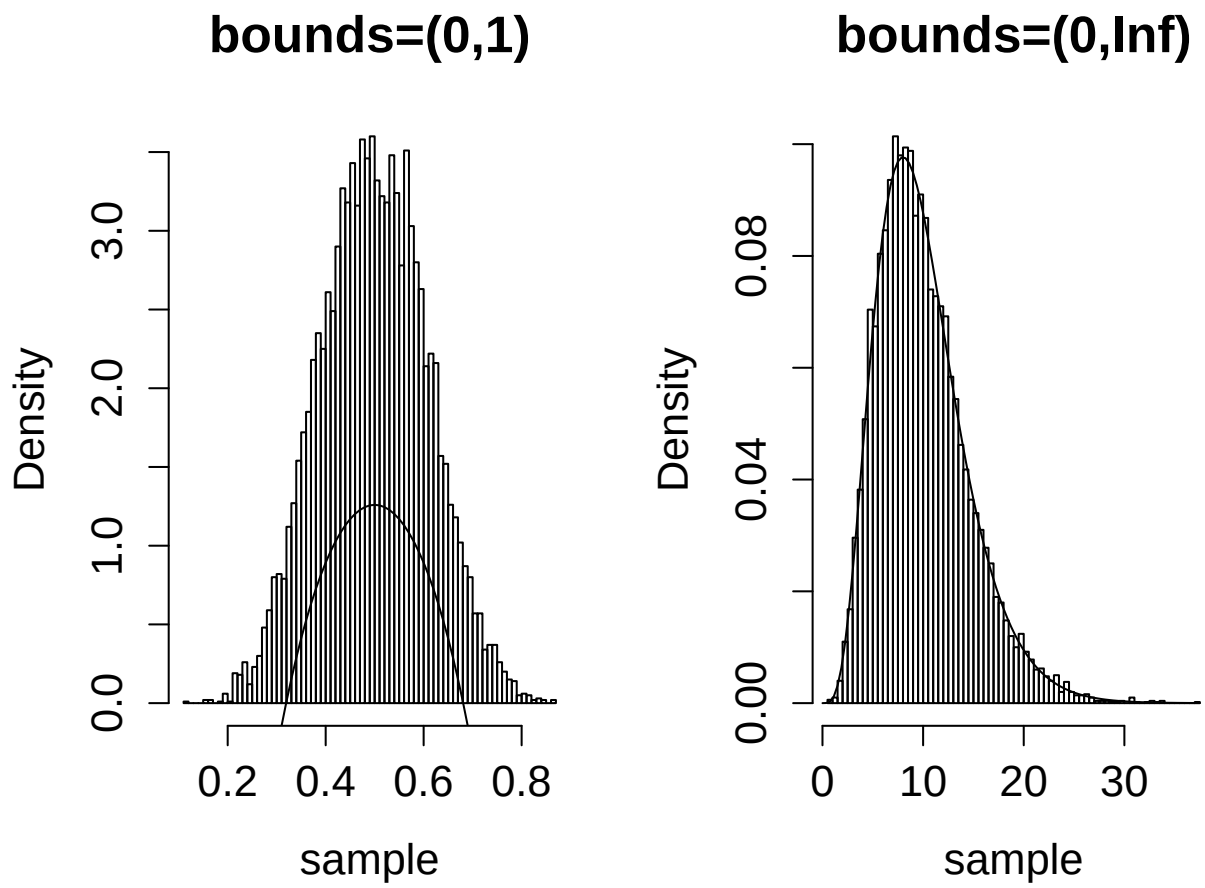
Figure 3: left: gamma(10,10), right: exponential(10)



Third we test the beta(10,10) with bounds= $(0, 1)$ and the chi-square (df=10) with a right bound 0, the code to do so and the histogram of the resulting samples are given below:

```
sample<-ars(10000,function(x){x^9*(1-x)^9/beta(10,10)},c(0,1))
sample<-ars(10000,function(x){1/(2^5*gamma(5))*x^4*exp(-x/2)},c(0,Inf))
```

Figure 4: left: beta(10,10), right: chi-square(df=10)



For the beta distribution, you may note that the plotted distribution curve falls below the sampled histogram, this is simply because the density is not normalized. But it is immediately apparent that the samples follow the same trend as the curve.

For methods testing, we have included in our final solution some simple lines of code to run, testing methods individually. Please see our source code for more details.

4 Contributions

Everyone in the group contributed relatively equally to each aspect of the project. Some efforts were more focused as follows:

Code writing

General structure: Lisa

Methods: James, Siwei, Hsin-Wei

Code testing

Methods testing: James, Siwei

Overall function tests: Lisa, Hsin-Wei

Documentation

Manual creation: Lisa

Project write-up: Hsin-Wei, Lisa