STAT 243 Final Project
Implementation of an Adaptive-Rejection Sampling Method

Ye Zhi, Boying Gong, Max Gardner, Alexander Brandt

December 17, 2015

1 Theory

Our final project was to implement an adaptive-rejection sampling method in R. Our method largely implements the methods described by Gilks and Wild. ARS attempts to efficiently generate random values according to a user-defined distribution (and a user-defined domain). The method follows a few general steps, exploiting the tendency of many important and useful distributions to be log-concave (even if the distributions themselves are not concave). For some distribution $g(x)$, let us define $h(x) = \log(g(x))$. Note, this point, if $h(x)$ is found to be not log-concave, then our program will reject the values.

Now, in order to efficiently sample from values (and not waste time generating values that will ultimately be rejected), we construct an upper envelope function that bounds $h(x)$. This is performed by taking two or more starting points from within the domain such that they lie within our users defined bounds. At points we will calculate the tangent's slope at x_i. We then calculate the x-intercept where these slopes meet with:

$$z_j = \frac{h(x_{j+1}) - h(x_j) - x_{j+1}h'(x_{j+1}) + x_j h'(x_j)}{h'(x_j) - h'(x_{j+1})}$$

Where $j \in [1, k-1]$. Note that z_0 and z_k are the upper and lower bounds of the function, respectively, even if those bounds are $-\infty$ or ∞.

Now that we can define the envelope for $x \in [z_{j-1}, z_j]$:

$$u_k(x) = h(x_j) + (x - x_j)h'(x_j)$$

By exponentiating and normalizing this $u_k(x)$, we now have a density from which to sample our random values:

$$s_k(x) = \frac{\exp(u_k(x))}{N}$$

Where N is simply the all-space integral value of $s_k(x)$ (i.e. $N = \int_D s_k(x)dx$).

Furthermore, we exploit a computationally inexpensive procedure to compute a lower hull, which is defined as:

$$l_k(x) = \frac{(x_{j+1} - x)h(x_j) + (x - x_j)h(x_{j+1})}{x_{j+1} - x_j}$$

$l_k(x)$ can allow us to perform a squeezing test to quickly accept or reject proposed sampling values. This is performed by selecting some value x^t from $s_k(x)$, and drawing a value w from the uniform distribution defined from [0,1]. We can accept x^t if:

$$w \leq \exp(l_k(x^t) - u_k(x^t))$$

Or if:

$$w \leq \exp(h(x^t) - u_k(x^t))$$

These are the squeezing test and rejection test, respectively. As we continue to compute $h(x^t$ and $h'(x^t)$, we can update the points that construct our upper and lower hulls, thereby increasing the chance of new values drawn from $s_k(x)$ being accepted. This is the mathematical and statistical intuition behind the implementation discussed in the next section.

2 Implementation

In order to create a modular solution, we have created several helper functions in various files which are called by ars in ars.R in an order described above.

1. dh.R

   The derivative of h is of critical importance throughout this procedure. To implement, we use a dh method that takes advantage of numerical derivation (via numericDeriv in the stats package) to find our tangents.

2. find_init.R

   Though it would always be best for the user, who has a better understanding of the distribution, to supply their initial points, it was our intention that the program perform well even in cases when the user did not. To this end, ars.R begins by calling find_init.R

3. get_zj.R

   To compute the intersection between our tangents, we use get_zj.R. It computes all the intersections given various x_i's from the formula given above.

4. upper_piecewise.R

   This uses a vectorized formulation to compute $h(x)$ by using a boolean operation based on x's range, the range of z_i, to compute the correct piece (i.e. only the relevant line segment for a given range will be used to compete $h(x)$.

3 Results/Examples

We provide some basic results in addition to the extensive testing of each function individually, which can be seen in more detail in the test suite defined in the package.

3.1 Normal Distribution

3.2 Chi-Squared Distribution

3.3 Truncated Normal Distribution

4 Roll the Credits

Ye Zhi

Boying Gong

Max Gardner developed auxillary functions for ARS, lead the documentation effort, and rolled everything up into a nice and neat R package. He is a first-year MS/PhD student in Civil Systems.

Alexander Brandt wrote an initial implementation of the ARS and LaTeX'd up the final report. He is a computational biologist interested in worms, humans, and what they have in common (and what they don't!).

5 Citations