# Final Project 2018 - STAT243

*Bunker, JD; Bui, Anh; Lavrentiadis, Grigorios*

*December 10, 2018*

## I. Purpose

This project aims to apply the adaptive rejection sampling method described in Gilks et al. (1992). The main function ars generates sample from any univariate log-concave probability density function. In adaptive sampling, the envelop and squeezing functions are not determined in advance, but are updated according to the new element that are included in the sample.

## II. Contributions of members

Responsibility to write auxiliary functions

Bunker, JD: Generate the initial function to sample starting points. Make an improvement for the efficiency of the Update_accept function and test the log-concavity of the given function.

Lavrentiadis, Grigorios: Piecewise exponential sampling to generate value x* based on calculated probabilities. Assemble auxiliary functions to generate the comprehensive ars function.

Bui, Anh: Update_accept function to decide whether x* is accepted or rejected in the final sample. Update the envelop and squeezing functions accordingly. Write the report and the formal testing.

## III. Theoretical background for rejection sampling

Assume f(x) is an univariate log-concave probability density function. We sample from g(x), which is a scaling of function f(x). Assuming that we have the envelop function $g_u(x)$ and the squeezing function $g_l(x)$ of g(x).

Let Y ~ Unif(0,$g_u(x)$) and where w ~ Unif(0,1). We reject $x*$ if

$$Y < g(x*)$$
$$\frac{Y}{g_u(x)} < \frac{g(x)}{g_u(x)}$$
$$w < \frac{g(x)}{g_u(x)}$$

The paper gives the algorithm of adaptive rejection sampling when working with the log of g(x), $g_l(x)$, and $g_u$, which will be discussed more in details in the auxiliary functions section.

# IV. Auxiliary functions

## 1. Initial function

## 2. Generate x* from piecewise exponential probabilities

The `SamplePieceExp` function draws samples $x^*$ out of a piece-wise exponential distribution using the inverse sampling approach. Initially a $Pinv$ sample is drawn from a 0 to 1 uniform distribution that corresponds to the cumulative probability of the random sample $x^*$ To find the bin at which $x^*$ belongs, $Pinv$ is compared with the cumulative probability of each bin. $x^*$ belongs to the bin whose cumulative probability ($Pcum_i$) is the smallest out of all bins that have $Pcum$ larger than $Pinv$. $x^*$ is estimated by solving the following cumulative probability equation, where $z_0$ is the left bound of the distribution and $P_j$ is the probability of bin $j$.

$$P_{inv} = \int_{z_0}^{x^*} s(x)dx = \sum_{j=1}^{i} P_j + \int_{z_i}^{x^*} s(x)dx$$

$DP$ equals to the probability $P(z_i > x > x^*)$ where $z_i$ is the lower bound of the bin $i$ where $x^*$ belongs

$$DP = \int_{z_i}^{x^*} e^{h(x_j)+(x-x_j)h'(x_j)}dx$$

To simplify the equation we define: $a = h(x_j) - x_j h'(x_j)$ and $b = h'(x_j)$ From this equation, $x^*$ equals to:

$$x^* = \frac{1}{b}log(DP\ b\ e^{-a} + e^{b\ z_i})$$

## 3. Update accept function

### Algorithm

Inputs: w ~ Unif(0,1)

$$l_k(x^*) = log(g_l(x^*))$$
$$u_k(x^*) = log(g_u(x^*))$$
$$h(x^*) = log(g(x^*))$$
$$s_k(x) = exp(u_k(x))/\left(\int_D u_k(x')\ dx'\right) = g_u(x)/\left(\int_D g_u(x')\ dx'\right)$$

The lower bound of h(x) is $l_k(x)$, which connects the values of function h on abscissaes. The function of $l_k(x)$ between two consecutive abscissaes $x_j$ and $x_{j+1}$ is

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

Let X be the domain of abscissaes, H be the domain of the realized function H at abscissaes, H_prime be the domain of the realized first derivative of function H at abscissaes, Z be the domain of intersection of tangent lines at abscissaes.

$$h'(x) = \frac{dlog(g(x))}{dx} = \frac{g'(x)}{g(x)}$$

The intersection of the tangents at $x_j$ and $x_{j+1}$ is

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

Then for x between $z_{j-1}$ and $z_j$

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

**Step 1**: If $w < exp(l_k(x^*) - u_k(x^*))$
- Accept $x^*$ when the condition is satisfied. Draw another $x^*$ from $s_k(x)$
- Reject $x^*$ when the condition is not satisfied.

**Step 2**: These two procedures can be done in parallel.
- Evaluate $h(x^*), h'(x^*)$. Update $l_k(x), u_k(x), s_k(x)$. X includes $x^*$ as an element.
- Accept $x^*$ if $w < exp(h(x^*) - u_k(x^*))$. Otherwise, reject.

Since the h(x), $l_k(x)$, $u_k(x)$ can be generated from vectors H, H_prime, and Z, we improve the efficiency of the calculation by efficiently updating H, H_prime, and Z. We append the vectors associated with the new abscissae x* and append to the existing vectors.

Multiple testing are generated based on values that $x^*$ can take. For example, if $x^*$ is out of the domain of X, $l_k(x^*) = -Inf$. If $x^*$ is at the minimum value X[1] and maximum value X[n] in the domain of X, $l_k(x^*)$ will take the values on the lines connecting X[1] and X[2], and connecting X[n-1] and X[n] respectively.