STAT 243 Final Project: Adaptive Rejection Sampling

Yuqi Yang, Shuyao Wang and Nolan David Gunter

12/15/2022

Github Package

- The Github page can be accessed at: https://github.berkeley.edu/yuqiyang/ars
- The .tar.gz file for the package can be accessed at: https://github.berkeley.edu/yuqiyang/ars/blob/main/ars_0.1.0.tar.gz

Description

Our implementation of adaptive rejection sampling is designed to encourage vectorized and modularized code for efficiency.

The `utility.R` file includes a variety of utility functions used to conduct a single task, most of which to called in `ars.R` where the `ars()` function available to users is defined.

- `zfindj()` is a helper function that finds z_{j-1} (stored as `z[j]`) and z_j (stored as `z[j+1]`) such that $x \in [z_{j-1}, z_j]$, and returns `j`
- `u_fun()` uses the definition of the $u_k(x)$ function given by equation (2) from Gilks and Wild.

Next, in order to draw random samples, we derive the inverse CDF from the equation (3) given in Gilks and Wilk, which is:

$$s_k(x) = \frac{\exp(u_k(x))}{\int_D \exp u_k(x')dx'}$$

We can calculate the integral in the denominator analytically:

$$\int_D \exp u_k(x')dx' = \sum_{j=1}^{k} \int_{z_{j-1}}^{z_j} \exp u_k(x')dx'$$

$$= \sum_{j=1}^{k} \int_{z_{j-1}}^{z_j} \exp(x'h'(x_j) + h(x_j) - x_j h'(x_j))dx'$$

$$= \sum_{j=1}^{k} [\frac{1}{h'(x_j)} \exp u_k(x')]_{z_{j-1}}^{z_j}$$

$$= \sum_{j=1}^{k} \frac{1}{h'(x_j)} (\exp u_k(z_j) - \exp u_k(z_{j-1}))$$

Note that the denominator is constant, so we may define it as C. Thus

$$s_k(x) = \frac{exp(u_k(x))}{C} = \frac{exp(xh'(x_j) + h(x_j) - x_j h'(x_j))}{C}$$

Note that $s_k(x)$ is the pdf of the probability distribution. To find the CDF ($Pr(X < x)$), we may take the integral $\int_{z_0}^{x} s_k(t)dt$, where z_0 is the lower bound.

Since $s_k(x)$ is piece-wise exponential, we can calculate the integral in a piece-wise way:

$$p = Pr(X < x) = \int_{z_0}^{x} s_k(t)dt = \sum_{i=1}^{j-1} \int_{z_{i-1}}^{z_i} s_k(t)dt + \int_{z_{j-1}}^{x} s_k(t)dt$$

$$= \sum_{i=1}^{j-1} \int_{z_{i-1}}^{z_i} \frac{1}{C} \exp(th'(x_i) + h(x_i) - x_i h'(x_i))dt + \int_{z_{j-1}}^{x} \frac{1}{C} \exp(th'(x_j) + h(x_j) - x_j h'(x_j))dt$$

$$= \sum_{i=1}^{j-1} \frac{1}{Ch'(x_i)} (\exp u_k(z_i) - \exp u_k(z_{i-1})) + \frac{1}{Ch'(x_j)} (\exp u_k(x) - \exp u_k(z_{j-1}))$$

In order to draw samples, we may draw p from a uniform distribution of $[0, 1]$ and use the inverse CDF to find the corresponding x. The inverse CDF is simply solving the above equation for x given p:

$$\frac{1}{Ch'(x_j)} (\exp u_k(x) - \exp u_k(z_{j-1}))$$

$$= p - \sum_{i=1}^{j-1} \frac{1}{Ch'(x_i)} (\exp u_k(z_i) - \exp u_k(z_{i-1})) \exp u_k(x) - \exp u_k(z_{j-1})$$

$$= Ch'(x_j)(p - \sum_{i=1}^{j-1} \frac{1}{Ch'(x_i)} (\exp u_k(z_i) - \exp u_k(z_{i-1})))$$

$$\exp u_k(x) = \exp u_k(z_{j-1}) + Ch'(x_j)(p - \sum_{i=1}^{j-1} \frac{1}{Ch'(x_i)} (\exp u_k(z_i) - \exp u_k(z_{i-1})))u_k(x)$$

$$= \log(\exp u_k(z_{j-1}) + Ch'(x_j)(p - \sum_{i=1}^{j-1} \frac{1}{Ch'(x_i)} (\exp u_k(z_i) - \exp u_k(z_{i-1}))))x$$

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

$$\int_{z_0}^{x} s_k(t)dt = \sum_{i=1}^{j-1} \int_{z_{i-1}}^{z_i} s_k(t)dt + \int_{z_{j-1}}^{x} s_k(t)dt$$

$$= \sum_{i=1}^{j-1} \int_{z_{i-1}}^{z_i} \frac{exp(xh'(x_i) + h(x_i) - x_i h'(x_i))}{C} dt + \int_{z_{j-1}}^{x} \frac{exp(xh'(x_j) + h(x_j) - x_j)h'(x_j)}{C} dt$$

$$= \sum_{i=1}^{j-1} \frac{exp(u_k(z_i)) - exp(u_k(z_{i-1}))}{Ch'(x_i)} + \frac{exp(u_k(x)) - exp(u_k(z_{j-1}))}{Ch'(x_i)}$$

Since the above process is complicated, we break it down into several functions: - `calculateC()` calculates the denominator C for $s_k(x)$ as defined above. - `pfindj()` finds j such that the cumulative probability up to $z__{j-1} <= p < $ cumulative probability up to z_j (denoted `z[j]` and `z[j+1]` respectively in code) and returns a list of `j` and the cumulative probability up to z_{j-1} (`z[j]`) - `sinvcdf_fun()` takes in the probability p and returns the corresponding x such that $Pr(X < x) = p$ given the distribution of $s_k(x)$

- `xfindj()` is a helper function for `l_fun()` that finds x_j and x_{j+1} such that $x_j <= x < x_{j+1}$. If $x < x_1$ or $x > x_k$, the function returns 0
- `l_fun()` is the squeezing function defined by equation (4) in Gilks and Wild.
- `isConcave()` tests if a function is concave according to the following definition, in steps of 0.01 size for λ:

A function f is concave in an interval $[x_1, x_2]$ for all $\lambda \in [0, 1]$ if:

$$f(\lambda x_1 + (1 - \lambda)x_2) - \lambda f(x_1) - (1 - \lambda)f(x_2) \geq 0$$

For example, we know that the log of a normal density should be concave, but the log of a cauchy density will not be.

```
log.norm <- function(x) log(dnorm(x))

isConcave(log.norm, 0, 10)
```

```
## [1] TRUE
```

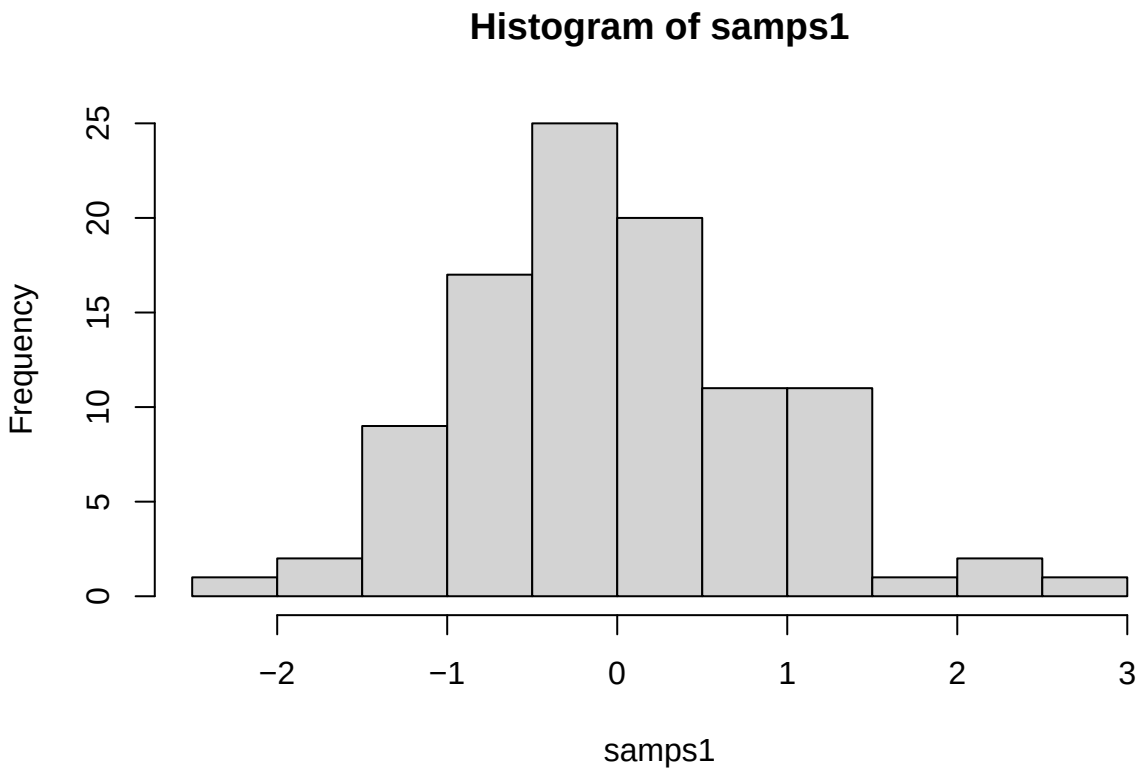
```
log.cauchy <- function(x) log(dcauchy(x))

isConcave(log.cauchy, 0, 10)
```

```
## [1] FALSE
```

Function Examples

Let's consider conducting adaptive rejection sampling in the following scenarios. First, we sample from a truncated normal distribution from -3 to 5.

```
set.seed(1)
samps1 <- ars(g = dnorm, n = 1e2, lower = -3, upper = 5)
hist(samps1)
```

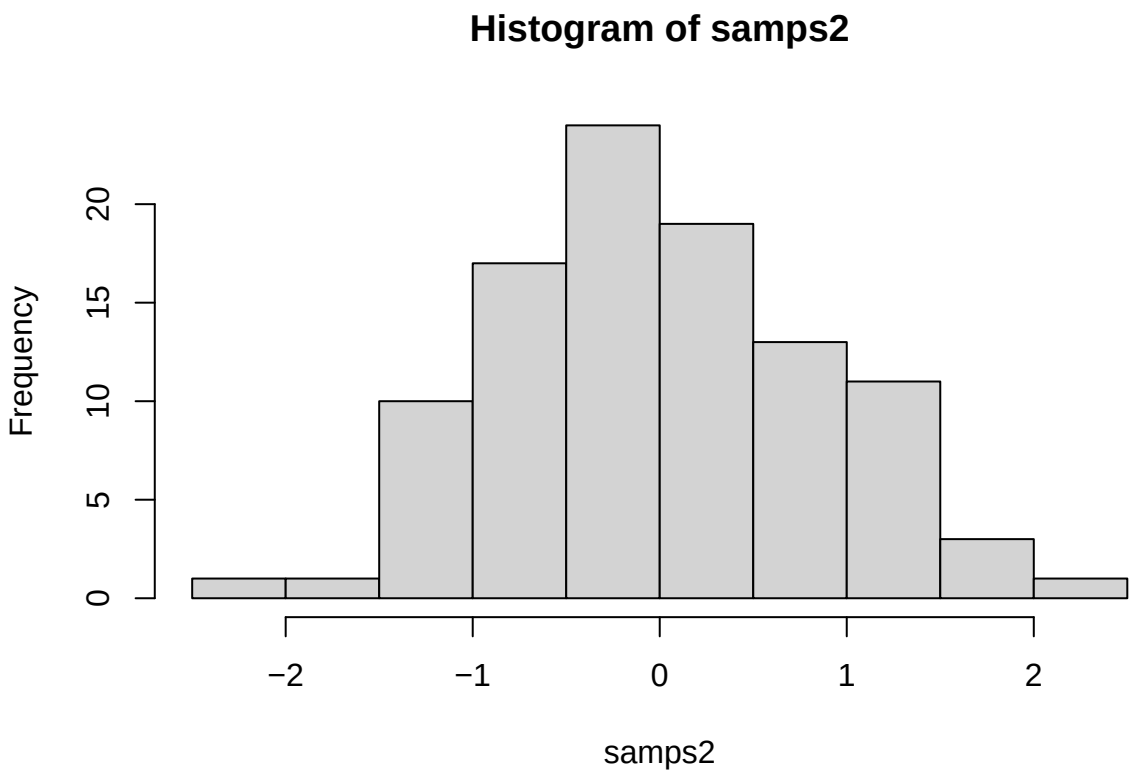


In this example, we don't supply lower or upper limits and see if the function is still able to set default bounds.

```
set.seed(1)
samps2 <- ars(g = dnorm, n = 1e2)
```

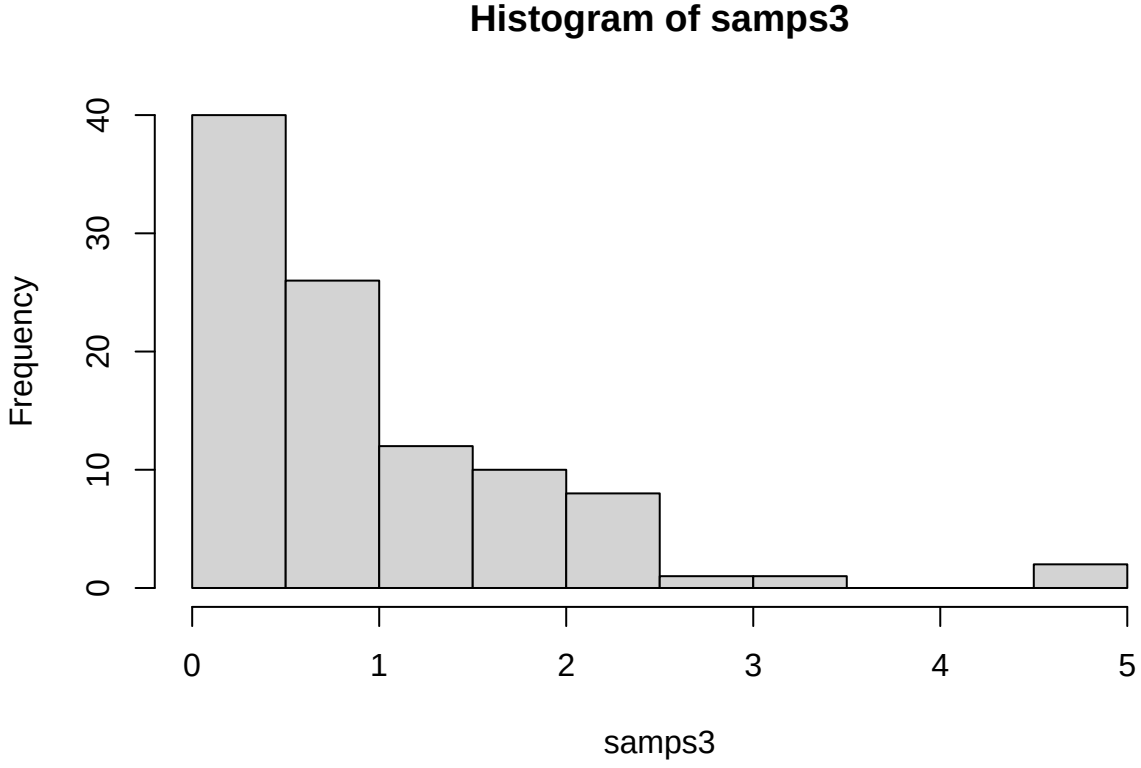
```
## Warning in ars(g = dnorm, n = 100): Setting bounds by default.
```

```
hist(samps2)
```



In this example, we sample from a truncated exponential distribution from 0 to 10.

```
set.seed(1)
samps3 <- ars(g = dexp, n = 1e2, lower = 0, upper = 10)
hist(samps3)
```



Contributions

All group members read the paper and wrote their own pseudo-code to try to understand the adaptive rejection sampling algorithm, and met multiple times over zoom.

- Nolan: wrote the initial utility functions in utility.R; troubleshooted errors in utility.R and ars.R; wrote descriptions and examples portions of report, and ars.Rd.
- Shuyao: wrote the initial test.R for test cases in test folder; troubleshooted errors in ars.R, utility.R, and report; and help finalize and connect different parts.
- Yuqi: wrote the initial main functions in ars.R; troubleshooted errors in utility functions and testing cases; and helped the group understand the piece wise.

Sources

Gilks, W.R. and Wild, P. Adaptive Rejection Sampling for Gibbs Sampling. *Journal of the Royal Statistical Society.* 41,2, 337:248. 1992.