STAT243: Group Project

Guillaume Baquiast, Mingyung Kim, Andreas Strand, Lu Zhang
(Github user name: baquiast/STAT_243_Project)

December 17 2015

1 Instructions

Sometimes the BCE cannot install pacakges that are required in order to run our *ars* package. Please follow these instructions for installation.

```
## Install the packages that are required in order to run our package.
install.packages("numDeriv")
install.packages("testthat")
install.packages("MASS")
install.packages("smoothmest")
install.packages("RUnit")
library(numDeriv); library(testthat)
library(MASS); library(smoothmest); library(RUnit)

## Install the ars package
install.packages("ars_0.1.7.tar.gz", repos = NULL, type="source")
library(ars)

## Run the ars function
par(mfrow=c(2,2))
mysample = ars(100, dnorm, x.start=c(-4,0,4), plot.type="bounds")
mysample = ars(100, dnorm, x.start=c(-4,0,4), plot.type="acceptance")
mysample = ars(100, dnorm, plot.type="bounds")
mysample = ars(100, dnorm, plot.type="acceptance")

# Run the main tests and unit tests
test_package('ars','main')
```

2 Comments on Functions

The objective of the adaptive rejection sampling is to efficiently generate samples from computationally expensive functions. The function follows the theoretical approach that Gilks and Wild (1992) introduced. We modularize the adaptive rejection sampler into sub functions: (1) checking log-concavity of the function, (2) finding piecewise linear lower bound and upper bound, (3) generating valid starting points, (4) performing initialization, sampling and updating steps, and (5) plotting lower bounds and upper bounds.

2.1 Numerical checks for log-concavity

This subfunction checks the log-concavity by evaluating derivatives of log densities, $h(x)$, at x that are rejected from the squeezing step. The log-concavity of the function is rejected only when the derivatives

increase as x increase. The function can thus catch the non-log-concave cases (i.e. where upper bounds and lower bounds are not bounding the log density) as calculations proceed.

2.2 Computing lower bound and upper bound

The function can generate samples efficiently because it uses a squeezing function, made from the upper bound, as sampling probabilities. It determines which samples to accept or to reject by evaluating the lower bound and the upper bound.

2.3 Generating starting points

The function generates valid starting points when these are not given by the user. First, our function finds two points of which $h(x)$ is numerically computational. When the $h(x)$ or $h'(x)$ cannot be computed at the domain boundaries in R, this function first sets starting points at the points located close to the boundaries and give numerical values. If the mode of the density is very close to one of them (x_1), our function replaces the other one (x_2) with a point of which log-density is the average of $h(x_2)$ and $\max(h(x))$.

2.4 Initialization, sampling and updating steps

The objective of this function is to extract the requested number of samples from the distribution. It carries out the three key steps in a vectorized way.

- The initialization step evaluates $h(x)$ at starting points.
- The sampling step generates a vector of samples based on the upper bound. We keep points that pass the squeezing test, and evaluate $h(x)$ at the remaining points. The procedure continues until it has generated the requested number of points. Vectorization makes the function fast.
- The updating step computes $h'(x)$ at the points that are rejected in the sampling step.

2.5 Plotting

This function uses our final results and plots (1) upper bounds and lower bounds or (2) rejection regions and acceptance regions.

3 Tests

3.1 Main tests

We tested that our main function returns appropriate samples for the eight generic distributions: Normal, Gamma (shape parameter ≥ 1), Beta (both parameters ≥ 1), Exponential, Chi Square (df ≥ 2), Logistic, Double Exponential, and Uniform. Each distribution is tested for several sets of parameters. The tests are based on the Kolmogorov–Smirnov Test and the Wilcoxon Rank Sum Test on a 5% significance level. Those tests compare if two samples (in our case, the one obtained from the built-in R function and one obtained from our *ars* function) follow the same distribution. We also plot bounds, acceptance regions and histograms of the samples, in order to visually check the output of our function. Examples of such plots are shown on the next page.

3.2 Unit tests

A unit test may discover bugs in each module. It should test various types of cases, also extreme ones. We tested our is.local.concave() function for several distributions that we know whether are log-concave. The chosen distributions are Normal, Gamma, Beta, Exponential, Chi Square, Logitstic, Double Exponential, Student's t, F, and Cauchy. Our function gives the expected results. The other subfunctions were compared with the hand calculations to see whether functionality is as intended.

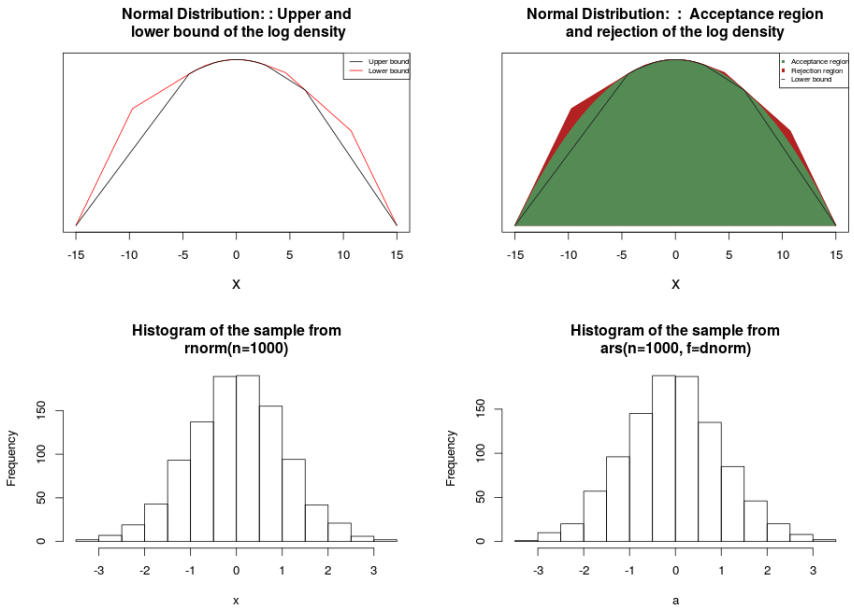

Figure 1: The main test for the Normal distribution: The top two panels show the bounds and the acceptance region generated from our *ars* function. The histograms compare the sample from the rnorm(0,1) function to the one from our *ars* function.

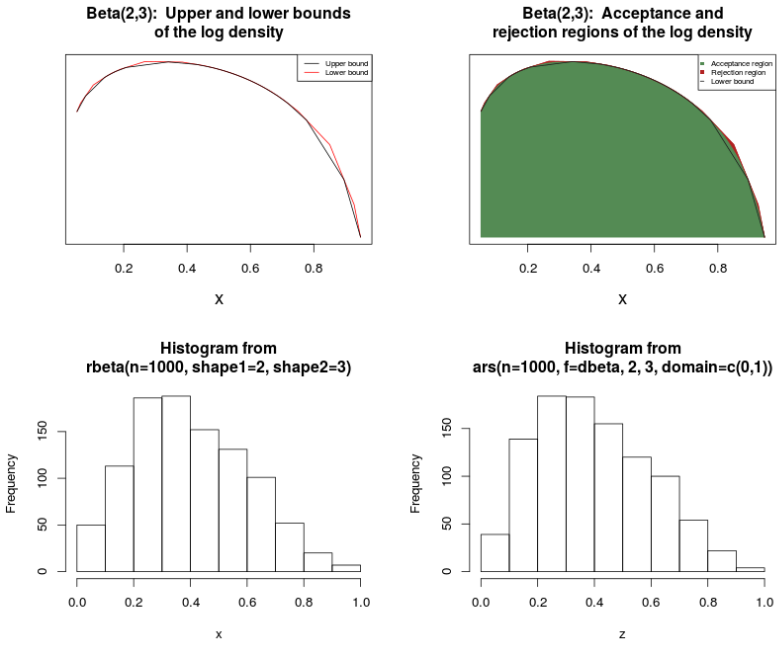

Figure 2: The main test for the Beta distribution: The top two panels show that our *ars* function works properly for the unsymmetric distribution. We can guess from the histogreams that the sample from the rbeta(2,3) function and the one from our *ars* function follow the same distribution.

4 Comments on the ARS Process

Bounds of the standard normal distribution as made by the ARS are shown in figure 3. Occasionally as the sample size n increases, a new value of the density is computed. Each new point provides a tighter bound of the distribution. With tighter bounds it is less likely that approving the next sample element requires calculating a distribution value. Another thing to note is that the bounds will lie closer to the distribution for values with high density since more values are chosen from dense parts.

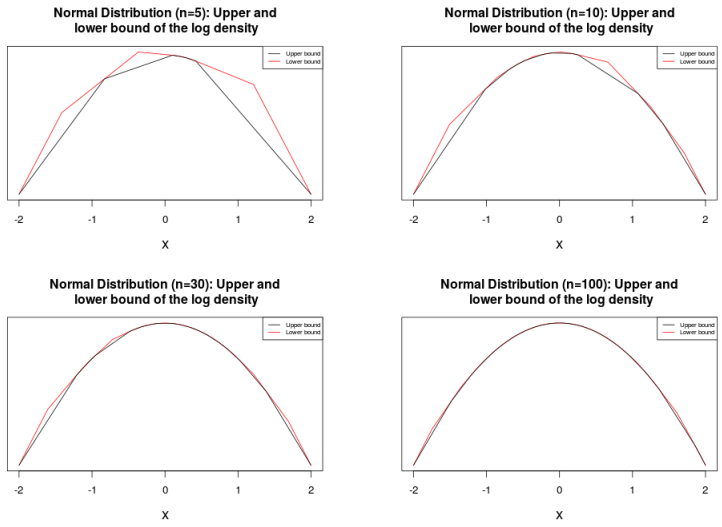

Figure 3: Development of ARS bounds by increasing sample size n. The distribution is computed when points in the sample are rejected from the squeezing steps. As more density values are known the bounds will move closer to the distribution.

5 Contributions

Mingyung Kim and Andreas Strand wrote the ARS function together. Guillaume Baquiast performed the main tests and Luna Zhang performed the unit tests. We wrote packages and documents together.