**UOL ID: 220460044**

**ST2195 Programming for Data Science**

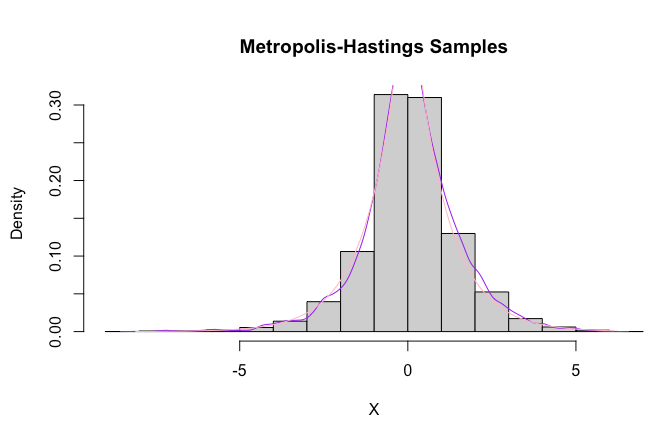
**Part 1 Report**

We work with the Markov Chain Monte Carlo, Metropolis-Hasting algorithm to simulate random numbers for the distribution with the probability density function:

We start by defining the probability density function and setting up the initial x value (x0), sample size (N) and standard deviation (s). We will allocate x0 as 0, N as 10000 and s as 1.

When setting up the Metropolis-Hasting algorithm, we create a function with parameters N, x0 and s into a numeric vector as the data structure to keep track of the generated x values. x takes the value, x0, x1, x2, … xN. We will refer to x0 as x\_current in this report.

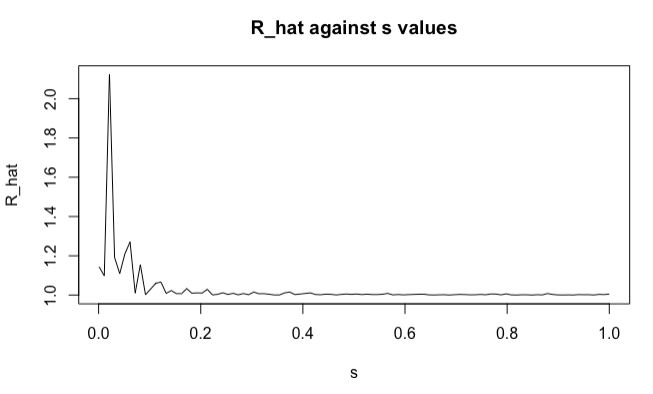
We simulate 1 random number for x\* from Normal distribution with mean = x\_current and compute the acceptance ratio lg\_r using f(x\*) and f(x\_current), taking log to ensure linearity. Thereafter we generate a random number u from uniform distribution [0,1] and use it to determine whether to reject lg\_u. If lg\_u < lg\_r, x\* is accepted else x\_current is accepted. The x\_current is then updated based on the criterion and the process is repeated until i=N and store all the x values into the vector.



We run the Metropolis-Hasting algorithm and plot a histogram and kernel density line graph (purple) with X values and density of X=x to provide a estimate of f(x). We also plot f(x) (pink) to see the quality of estimates. Using the Monte Carlo estimates, the sample mean is 0.07122055 and the sample standard deviation is 1.406888.

We follow the steps provided to obtain the convergence diagnostics . First we need to set up a matrix to store the chain sequence. We repeat what was done for the x value vector in the previous spart for the chain and generate the chains with the parameters N=2000, s=0.01 and J=4 in order to compute . We calculate the step-by-step from the instructions and we will Mj, Vj for j= 1 to 4, W, M, B and . In this case, R is 1 indicating a convergence.

We will change the set up to calculate , keeping N and J fixed with s values in the interval between 0.001 and 1 by generating 100 evenly spaced s\_values from the interval. We will then establish the chain sequence with s\_values and put it through the metropolis-Hasting algorithm.



We plot the against s\_values graph and observe that decreases when s increases from 0 to 1. The initial is highest at almost 2.2 initially when s = 0, quickly decreasing to 1.2 at s = 0.15, thereafter stabilises near 1. We note that some values are randomly generated so we setseed to ensure reproducibility. As different sets of numbers are generated in python and r, there is a variance in results.