IE531: Algorithms for Data Analytics
Spring, 2018
Programming Assignment 1: Working in an Unfair-World!
Due Date: February 16, 2018

1 Introduction

We have an unknown, possibly unfair, three-sided dice (cf. figure 1). You get one of three outcomes (say, $x_i \in \{1, 2, 3\}$) when you roll this dice. We have no clue as to what $Prob(x_i = 1), Prob(x_i = 2)$, and $Prob(x_i = 3)$ are. All we have to go on is that $Prob(x_i = 1) + Prob(x_i = 2) + Prob(x_i = 3) = 1$. We also know that by repeated tosses of this dice we can generate a string of i.i.d. outcomes $\{x_i\}_{i=1}^{\infty}$. Empirically estimating these probabilities are ruled-out, as there is no way to estimate the individual probabilities with *certainty* using a finite-set of outcomes.

We are asked to use this unknown, possibly unfair, three-sided dice as the only available source of "randomness," and to (a) generate an i.i.d. sequence of exponential RVs with rate λ (to be input by the user), and (b) to generate continuous-RVs that are from a unit-normal distribution. That is, we have to generate a set of Univariate Gaussian RVs $\{y_i\}_{i=1}^{\infty}$ such that $y_i \sim N(0, 1)$.

Figure 1: Three-sided-dice.

2 Discussion

We know that if we have a way of generating u.i.i.d. RVs we can (a) generate the Exponential RVs using the *Inverse Transform Method*, and (b) generate the Unit-Normal RVs using the *Box-Muller Transform*. **We are not permitted to use** `get_uniform()` **for this exercise. All we have is the unknown, possibly unfair, three-sided dice.** Essentially, we have to find a way of "converting" the outcomes of repeated-tosses of this three-sided dice into u.i.i.d. RVs.

I will go over the process of simulating a fair-coin (i.e. simulating a coin whose probability of "heads" and "tails" being equal) using the unknown, possibly unfair, three-sided dice, in class. This method is an illustration of a large-class of "accept/reject" algorithms that find use in different forms-and-shapes in data analytics. The resulting simulated-fair-coin is used to generate u.i.i.d. RVs. Following this, the rest of the programming exercise should be straightforward.

3 Requirements

What I need from you:

1. *.cpp and *.h files that uses a stream of i.i.d. discrete RVs $\{x_i\}_{i=1}^{\infty}$, where $x_i \in \{1, 2, 3\}$, where $Prob(x_i = 1) = p_1, Prob(x_i = 2) = p_2$, and $Prob(x_i = 3) = (1 - p_1 - p_2)$, where p_1 and p_2 are not known (the values of p_1 and p_2 are assigned randomly). Your code should produce

   (a) an exponential RVs generator with a user defined rate (i.e. λ; FYI, the mean of exponential-RV distribution is $\frac{1}{\lambda}$).

   (b) an unit-normal RV generator.

   This code should take the following inputs at the command line – #trials (for CDF generation); the rate of the exponential process, the name of the output file for the unit-normal CDF, and the name of the output file for the exponential CDF. The intent to compare the theoretical- and empirical-CDF plots to verify the correctness of your code. If you are lost, watch the video instructions on Compass.

These files can be submitted on Compass directly. Please do not e-mail them to the TA or me.

I have provided a hint on Compass for anyone that needs it. If you are working off the hint, you essentially have to fill-in the requisite C++ code in the commented spots where it says

"// write code here."