CECS 552 Programming Assignment 1, Fall 2016

Dr. Todd Ebert

Due Date: 6:30pm on October 11th

Presenting Your Work.

Submit via email an R script named `assign1-StudentFirstName.R`. For example, if your first name is Nitesh, then you would submit the file `assign1-Nitesh.R`. Line 1 of the file should include your full name as a comment. Also, for each block of code that pertains to some exercise, say Exercise 1, preface the code with the comment (all in caps)

```
#EXERCISE 1
```

Each function should be named exactly as it appears in the exercise, and have an input signature that follows the order specified in the exercise. Your email submission should be postmarked no later than the time shown on the due date. Submissions received after that time, but on the same day will lose one letter grade. Submissions emailed on a later day will either not be accepted, or will lose two letter grades.

Plagiarism

Plagiarism is defined for this assignment as the practice of taking someone else's code and passing it off as one's own. Although it is OK to discuss this assignment with other students, in the end each student must implement his or her own source code based on his or her own coding style, and way of seeing the solution. Students suspected of plagiarism will receive an automatic course grade of "F".

Exercises

1. The Craps dice game is played by rolling a pair of dice. A player wins if she rolls a 7 or 11 on the first roll, and loses if her first roll is 2,3, or 12. On the other hand suppose $X \in \{4, 5, 6, 8, 9, 10\}$ is the first roll. Then the player continues to roll until either i) she rolls a 7 and loses, or ii) she rolls X and wins. Implement an R function called `craps` which takes zero inputs, and returns an output of 1 (win) or 0 (loss).

2. Implement an R function called `estimate_bernoulli` that takes as inputs i) a function f (that takes zero inputs and randomly outputs either 0 or 1) ii) $\delta > 0$, and iii) $\epsilon > 0$, and returns an estimate of the probability that $f()$ returns 1. The estimate is accurate to the degree that the δ-confidence interval has a width that does not exceed 2ϵ. To implement the function, sample $f()$ in batches of 10,000 samples. After taking a batch of samples, check if

$$(\text{s.e.})\Phi^{-1}(\frac{1+\delta}{2}) \leq \epsilon.$$

   If this is the case, then return estimator $\hat{\lambda}$. Otherwise, proceed with the next batch of 10,000 samples.

3. Use `estimate_bernoulli` to estimate the probability of winning a game of craps. With 90% confidence the estimate should be within 0.005 of the actual value. Add a comment to your R scipt showing the call made to `estimate_bernoulli` with the appropriate inputs, along with the returned estimate.

4. Implement the R function `network_reliability` (havng zero inputs) which has the effect of estimating the reliability of the network that was pictured in the Monte Carlo lecture, assuming that each edge is reliable with probability 0.999. Do this by repeatedly sampling network configurations, where each configuration is obtained by sampling seven (one for each edge) independent Be(0.999) random variables. For a single sample configuration, assign $X = 1$ if there is no path from s to t, and $X = 0$ otherwise. In other words, the estimator $\hat{\lambda}$ estimates the probability of a broken configuration. Continue sampling until 50 broken configurations have been recorded. Then `network_reliability` will have the effect of printing i) $\hat{\lambda}$, ii) sample variance $\hat{\sigma}^2$, and iii) the number of samples n needed to reach 50 broken configurations.

5. Repeat the previous exercise by implementing the R function `network_reliability2` that now uses the variance reduction technique described in pages 15 and 16 of the Monte Carlo lecture. Now, when sampling the network Be(0.999) is replaced with Be(q), where $q = 1 - r/m = 1 - 2/7 = 5/7$. Moreover, for a broken configuration $\overline{x}$, $X = 1$ is replaced with $\pi(\overline{x}, 0.999)/\pi(\overline{x}, 5/7)$.