Homework 5 [75 points] / Updates in magenta

Questions related to the homework can be posted on the Piazza forum site, but do not post answers to homework problems. We will try to respond promptly to posted questions. You may send questions after 9pm the day before the assignment is due, but there is no guarantee they will be answered (on time).

You may discuss the homework problems and computing issues with other students in the class. However, you must write up your homework solution on your own. In particular, do not share your code or homework files with other students. If you collaborate with other students list their names at the beginning of your writeup.

This homework will build upon the material presented in lectures, and will be focused on Monte Carlo integration. All programming should be done in `R`. If there are any difficulties starting the homework, please contact the TA.

Points will be allocated to good code style and (mathematical) clarity. Points *may* be deducted if code and homework is unreadable, so take note of this.

Files to submit on CMS

Do check to see that you have submitted all these files. Each part will be graded as a whole. The writeups should either be TeXed up or done using R Markdown. Writeups are independent of each other.

1. Part One
   - `Finv.r`
   - `writeup_one.html` or `writeup_one.pdf`
2. Part Two
   - `writeup_two.html` or `writeup_two.pdf`
3. Part Three
   - `inverse_gen.r`
   - `proposed_gen.r`
   - `writeup_three.html` or `writeup_three.pdf`
4. Part Four
   - `rej_sampler.r`
   - `writeup_four.html` or `writeup_four.pdf`

Part One: Inverse CDF method

a) Suppose that U_1 and U_2 are independent uniform random variables on $(0, 1)$. Show the probability density of the sum, $X = U_1 + U_2$, forms a symmetric triangle above the interval $(0, 2)$. Hence derive the cumulative distribution function of X.

b) Write a function to generate values of X using the inverse CDF method.

```
Finv=function(u}
  {
    # u must be in (0,1)
    # your code
    return(x)
  }
```

Submit `Finv` to CMS.

c) Use the function in part 1b to generate a random sample of $N = 1000$ from the triangular distribution. Display a histogram of the sample values using break points at intervals of 0.2; i.e. divide the support into 10 bins of equal width.

d) Tabulate the observed frequency counts in the 10 bins and the corresponding expected values. Use the Pearson chi-squared statistic to assess the agreement between observed and expected counts.

Part Two: Variance reduction

Problem 3 from Section 20.4 of JMR.

Consider the integral

$$I = \int_0^1 \sqrt{1 - x^2} dx$$

a) Estimate I using Monte-Carlo integration. Include both the estimate and code in the writeup.

b) Estimate I using antithetic sampling, and compute an estimate of the percentage variance reduction achieved by using the antithetic approach. Include the estimates and code in the writeup.

c) Approximate the integrand by a straight line and use a control variate approach to estimate the value of the integral. Estimate the resulting variance reduction achieved. Specify the control variate used and include the estimate and code in the writeup.

d) Use importance sampling to estimate the integral I. Try using three importance sampling densities, and compare their effectiveness. Specify these importance sampling densities, estimates, and include code in the writeup.

Part Three: Generating maximums and minimums

Suppose we have i.i.d. random variables $X_1, X_2, \ldots, X_n$ drawn from the cumulative distributions $F_{X_1}, F_{X_2}, \ldots, F_{X_n}$, where $F_{X_i}(x_i) \leq \mathbb{P}[X_i \leq x_i]$. Further suppose we define $X_{\max} = \max\{X_1, X_2, \ldots, X_n\}$.

One way to find the cumulative distribution function of $F_{X_{\max}}$ is by its definition. By considering $\mathbb{P}[X_{\max} \leq x]$ or otherwise, derive the cumulative distribution function of $F_{X_{\max}}(x)$.

Similarly, suppose we define $X_{\min} = \min\{X_1, X_2, \ldots, X_n\}$.

Derive the cumulative distribution function of $F_{X_{\min}}(x)$.

One method of generating the random variables of maximums and the minimums is using the inverse transformation method.

Suppose we could easily generate random variables from the distribution functions $F_1, F_2, \ldots, F_n$ with the inverse transform method.

Write down the steps of an algorithm (using the inverse transform) to generate $X_{\max}$ and $X_{\min}$ for any arbitrary $X_1, \ldots, X_n$.

One question[1] we might be interested in is speed. How fast does this inverse transform algorithm take compared to other algorithms?

In this part, we will look at the generation of $X_{\max}$, where $X_1, \ldots, X_N$ are i.i.d exponential random variables with the same rate λ. Write a function:

```
inverse_gen<-function(num_var, N, lambda){
  # Code here
  return(vec_max)
}
```

that takes in:

- `num_var`: The number of $X_{\max}$ random variables wanted from the distribution given

[1]Technically, three questions. Those of you who have done CS courses on the side should know this. Those who don't - always consider: *runtime (how fast?)*, *accuracy (how accurate?)*, *memory (how much memory is needed to run this?)* ; though in this course we only look at speed and accuracy, and only in terms of plotting time taken / MSEs (we do not look at big O notation or do calculus).

by $F_{X_{\max}}$. how to generate Fxmax?

- `N`: The total number of exponential random variables X_i, such that we want to find $X_{\max} = \max\{X_1, \ldots, X_N\}$ how to find X1,...Xn
- `lambda`: The rate of these exponential random variables

and gives a vector as an output, corresponding to `num_var` random variables drawn from the distribution given by $F_{X_{\max}}$.

Submit `inverse_gen` to CMS.

We compare this to another method which looks at the tails of the distribution of X_i.

Suppose we define the tail densities f_a to be:

$$f_a(x) = \begin{cases} \frac{f(x)}{p} & x > a \\ 0 & x \leq a \end{cases}$$

and $\overline{f}_a$ to be:

$$\overline{f}_a(x) = \begin{cases} \frac{f(x)}{1-p} & x \leq a \\ 0 & x > a \end{cases}$$

where $p = 1 - F(a)$, i.e. the tail probabilities of F.

Then we can think of the number of X_is that exceed a as being binomially distributed with parameters N and probability p. Note that p depends on a, and is thus up to the user (you!) to pick what they are.

Thus, instead of generating $X_1, X_2, \ldots, X_N$ and taking the maximum, we could just generate a binomial (n, p) random variable $B = b$, generate b random variables from the tail density $f(a)$, and find the maximum.

The algorithm to generate **one** variable of $X_{\max}$ is thus given (in pseudocode):

```
1.  Generate a binomial random variable B = b with parameters N and p
2.  If b = 0
3.    Generate X_1, X_2, .. from f_a_bar until exactly N of them satisfy X_i < a
4.    Set X_max = max(X_i | X_i < a)
5.  Else
6.    Generate X_1, X_2, .. X_b from f_a
7.    Set X_max = max(X_1, .. X_b)
```

Recall that we want to find $X_{\max} = \max\{X_1, \ldots, X_N\}$, where each X_i comes from an exponential distribution with rate λ.

Given some p, a, write down how you would use the inverse transform method to simulate variables from f_a and $\overline{f}_a$. You can assume we will be using the exponential distribution here.

Given the description of the above algorithm, what heuristics would you use to pick a suitable p (and thus a)?

Thus, write a function:

```
proposed_gen<-function(num_var, N, lambda, a){
  # Code here
  return(vec_max)
}
```

that takes in:

- `num_var`: The number of $X_{\max}$ random variables wanted from the distribution given by $F_{X_{\max}}$.
- `N`: The total number of exponential random variables X_i, such that we want to find $X_{\max} = \max\{X_1, \ldots, X_N\}$
- `lambda`: The rate of these exponential random variables

and gives a vector as an output, corresponding to `num_var` random variables drawn from the distribution given by $F_{X_{\max}}$, using the above algorithm. You may use `rbinom` to generate variables from a binomial distribution.

Submit `proposed_gen` to CMS.

Lastly, compare the time taken using these two different algorithms, by generating random variables from $F_{X_{\max}}$, where each X_i is i.i.d. exponential distributed with fixed λ. Your comparisons should reflect what you have learnt in lectures / labs, or done in previous homeworks.

Since this is open-ended, we would be extremely surprised if we see identical comparisons.

Part Four: Rejection Sampling

In this question, we will apply rejection sampling to a problem, and get some intuition as to why rejection sampling works.

In the lecture slides, the following three steps are given under the Generalized Rejection Method.

Call $kh(x)$ the envelope for $f(x)$:

Step 1: Generate $Y \sim h(\cdot)$

Step 2: Generate $Z \sim U(0, kh(Y))$

Step 3: Accept Y if $Z < f(Y)$

One intuitive way to think about this is in terms of height. Suppose we wanted to simulate a Gamma $(2, 1)$ random variable with density function $f(x)$, and we chose the envelope to be an exponential random variable with rate $\frac{1}{2}$ with density function $h(x)$.

Using a grid of equally spaced values from 0 to 10 with increments of 0.01, plot the density functions of $f(x)$ and $2h(x)$.

Color the density function $f(x)$ blue, and the density function $h(x)$ red.

Ensure that the plot is distinctly labelled, and put both plot and code in your writeup.

Optional 1: Using the plot, and thinking of a point (x, y), convince yourself that in the following steps of the above algorithm, we are:

Step 1: Sampling a *random location* on the x axis, thus fixing $\tilde{x}$ in (x, y).

Step 2: At that location on the x axis, sampling a *random y location* which is between the x axis and below our envelope (ask yourself what does k do here), thus fixing $\tilde{y}$ in (x, y).

Step 3: If our point $(\tilde{x}, \tilde{y})$ is below $f(\tilde{x})$, we accept, otherwise we reject.

You may use `abline` in the plot you have created if you feel it is helpful.

Optional 2: Convince yourself further by giving a rigorous proof (this is straightforward if we view the steps as sampling these random locations) that the rejection sampling method

works[2].

From our plot, it *seems* possible that we could simply do the following:

1. Set $B = \max\{f(x)\}$
2. Scale a uniform distribution by B, since the line $y = B$ would always be above $f(x)$
3. Use the rejection sampling method

Why would it be hard (if not impossible), to implement the above method?

We will now use the rejection sampling algorithm to simulate a Gamma(2,1) random variable, *but conditional on its value being greater than 4*. We will denote the density function to be $f(x)$.

Show that the density function of this variable is given by:

$$f(x) = \frac{x}{5} \exp\{-(x-4)\}$$

The next step would be to pick an envelope and the optimal value k. In practice, picking a good envelope is difficult[3].

We consider these three possible envelopes:

- $g_1(x)$ being the density of an exponential random variable with rate $\frac{1}{2}$, conditional on its value being greater than 4
- $g_2(x)$ being the density of an exponential random variable with rate $\frac{1}{5}$, conditional on its value being greater than 4 what is the k here
- $g_3(x)$ being the density of a (standard) Cauchy random variable, conditional on its value being greater than 4

For *each* of these three envelope densities, derive their density functions, and thus find the corresponding optimal values k_i.

Furthermore, for *each* of the three envelope densities, write code that:

[2]In some years back, this would be a nice homework question with guiding questions, but since this proof can easily be found online, there's no point in giving it (there will be students who would just copy the proof without understanding why it works, thus destroying the curve for honest students who don't cheat). That being said, you can refer to this proof together with the plot you have made and try to understand the basic idea behind the rejection sampler.

[3]If you are a mathematician, consider piecewise functions of polynomials of degree 1 and 2 / "splines".

- Generates $N = 1,000$ random variables from $f(x)$
- using the rejection sampling method
- by generating random variables from $g_i(x)$
- and also reports the total number of random variables rejected (from $g_i(x)$)

Submit this code as `rej_sample.r`. At a minimum, the code should be well-documented and relatively optimized. Points will be given for efficient code and presentation (output of code). The grader reserves the right to give zero points if the code is un-optimized (such that it takes the grader an extremely long time to run the code).

Put the results of your code in the writeup as well.

Display the graphs of $f(x)$, $k_1 g_1(x)$, $k_2 g_2(x)$ and $k_3 g_3(x)$ on one plot. Ensure the graphs and the plot are clearly labelled. Is there any relation between the total number of random variables rejected from the rejection sampler and the plot? Explain.