MAST30027: Modern Applied Statistics

Week 9 Lab

1. Consider the following program, which performs a simulation experiment. The function `X.sim()` simulates some random variable X, and we wish to estimate $\mathbb{E}X$.

```
# set.seed(7)
# seed position 1
mu <- rep(0, 6)
for (i in 1:6) {
  # set.seed(7)
  # seed position 2
  X <- rep(0, 1000)
  for (j in 1:1000) {
    # set.seed(7)
    # seed position 3
    X[j] <- X.sim()
  }
  mu[i] <- mean(X)
}
spread <- max(mu) - min(mu)
mu.estimate <- mean(mu)
```

   (a) What is the value of `spread` used for?

   (b) If we uncomment the command `set.seed(7)` at seed position 3, then what is `spread`?

   (c) If we uncomment the command `set.seed(7)` at seed position 2 (only), then what is `spread`?

   (d) If we uncomment the command `set.seed(7)` at seed position 1 (only), then what is `spread`?

   (e) At which position should we set the seed?

2. For $X \sim \text{Poisson}(\lambda)$ let $F(x) = \mathbb{P}(X \leq x)$ and $p(x) = \mathbb{P}(X = x)$. Show that the probability function satisfies

$$p(x+1) = \frac{\lambda}{x+1} p(x).$$

Using this write a function to calculate $p(0), p(1), \ldots, p(x)$ and $F(x) = p(0) + p(1) + \cdots + p(x)$.

If $X \in \mathbb{Z}_+$ is a random variable and `F(x)` is a function that returns the cdf F of X, then you can simulate X using the following program:

```
F.rand <- function () {
  u <- runif(1)
  x <- 0
  while (F(x) < u) {
    x <- x + 1
  }
  return(x)
}
```

In the case of the Poisson distribution, this program can be made more efficient by calculating F just once, instead of recalculating it every time you call the function `F(x)`. By using two new variables, `p.x` and `F.x` for $p(x)$ and $F(x)$ respectively, modify this program so that instead of using the function `F(x)` it updates `p.x` and `F.x` within the `while` loop. Your program should have the form

```
F.rand <- function(lambda) {
  u <- runif(1)
  x <- 0
```

```
  p.x <- ?
  F.x <- ?
  while (F.x < u) {
    x <- x + 1
    p.x <- ?
    F.x <- ?
  }
  return(x)
}
```

You should ensure that at the start of the `while` loop you always have `p.x` equal to $p(x)$ and `F.x` equal to $F(x)$.

Check that your simulation works by choosing a parameter, generating a large number of random variables, using them to estimate the probability mass function, and comparing your estimates to the true values (which you can get using `dpois`).

3. (a) Here is some code for simulating a discrete random variable Y. What is the probability mass function (pmf) of Y?

```
Y.sim <- function() {
  U <- runif(1)
  Y <- 1
  while (U > 1 - 1/(1+Y)) {
    Y <- Y + 1
  }
  return(Y)
}
```

   Let N be the number of times you go around the while loop when `Y.sim()` is called. What is $\mathbb{E}N$ and thus what is the expected time taken for this function to run?

   (b) Here is some code for simulating a discrete random variable Z. Show that Z has the same pmf as Y

```
Z.sim <- function() {
  Z <- ceiling(1/runif(1)) - 1
  return(Z)
}
```

   Will this function be faster or slower that `Y.sim()`?

4. Consider the continuous random variable X with pdf given by:

$$f_X(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} \qquad -\infty < x < \infty.$$

   X is said to have a standard logistic distribution. Find the cdf for this random variable. Show how to simulate a rv with this cdf using the inversion method.

5. The double exponential or Laplace distribution has the following density, for some $\lambda > 0$,

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x|}, \text{ for } -\infty < x < \infty.$$

   Plot the density and the cumulative distribution function, F.

   Suppose that A has an exponential distribution with rate λ, and that B is independent of A and takes on values $+1$ and -1 with equal probability. Show that AB has a Laplace distribution with parameter λ, then write a function to simulate Laplace rv's.

6. (a) Construct an rejection sampling algorithm to generate a truncated exponential distribution, which has the pdf $p(z) = \frac{e^{-z}}{1-e^{-1}}$, $0 < z < 1$.

   (b) Calculate the mean and variance for the pdf $p(z)$ in (a).

(c) Write an R program to implement the algorithm in (a) and use it to generate a sample of 1000 observations. Plot a histogram of the sample. Calculate the sample mean and variance, and compare them with the results in (b).

(d) Show that the following algorithm also simulates from the distribution in (a).

1° Generate U from Unif(0,1);

2° If $U > e^{-1}$ then deliver $Z = -\ln(U)$; otherwise go to 1°.

7. Suppose $g(u)$ is a decreasing function of u. Let a random number X be generated by the following algorithm:

1° Generate U from Unif$(0, 1)$ and V from Unif$(0, 1)$ independently.

2° If $U + V < 1$, then deliver $X = g(U)$; otherwise, go to 1°.

Show that the distribution function of X is given by

$$F(x) = P(X \le x) = (1 - g^{-1}(x))^2, \quad g(1) \le x \le g(0).$$

8. A random number X is to be generated by the following rejection algorithm

1° Generate two independent Unif$(0, 1)$ random numbers U and V.

2° If $U^2 + V^2 \le 1$, deliver $X = U$; otherwise return to 1°.

(a) Find the cdf and pdf of X. Also find the mean and variance of X.
HINT: The integral $\int_0^x \sqrt{1-u^2}du = \frac{1}{2}(\arcsin x + x\sqrt{1-x^2})$.

(b) What is the efficiency of the algorithm? Denote by N the number of times that step 1° is to be executed to get one X value. Name the distribution of N. Also write down the mean and standard deviation of N.

(c) Write an R program to implement the algorithm. Then generate a sample of 1000 X values, and give it a numeric and a graphic summary.