Probability, Computation and Simulation Homework 2

Brian Cervantes Alvarez
October 8, 2024

Problem 4

Give a method for generating a random variable having the distribution function

$$F(x) = 1 - \exp(-\alpha x^{\beta}), \quad 0 < x < \infty$$

This describes a Weibull random variable.

Pseudocode

1. We can generate a random number u using a $Uniform(0, 1)$.
2. Next, we can compute the inverse of the Weibull CDF: $X = \left(-\frac{\log(1-u)}{\alpha}\right)^{\frac{1}{\beta}}$.
3. Lastly, we output the random variable X.

Putting it in practice

```
set.seed(202425)
alpha <- 2
beta <- 3
u <- runif(1)
X <- (-log(1-u)/alpha)^(1/beta)
X
```

```
[1] 0.9606969
```

Problem 6

Let X be an exponential random variable with mean 1. Simulate a random variable whose distribution is the conditional distribution of X given $X < 0.05$. Its density function is:

$$f(x) = \frac{e^{-x}}{1 - e^{-0.05}}, \quad 0 < x < 0.05$$

Pseudocode

1. First, generate an exponential r.v. X with rate 1.
2. If $X < 0.05$, we can accept it. Otherwise, repeat.
3. Finally, we can repeat this process 10000 times to estimate its expected value.

Putting it in practice

```r
set.seed(202425)
N <- 10000
accepted <- replicate(N, {
  accept <- FALSE
  while(!accept) {
    X <- rexp(1)
    if(X < 0.05) accept <- TRUE
  }
  X
})
mean(accepted)
```

```
[1] 0.02479158
```

Problem 10

A casualty insurance company has 1000 policyholders, each presenting a claim with probability 0.05. The claims are exponentially distributed with a mean of $800. Estimate the probability that the total claims exceed $50,000 using simulation.

Pseudocode

1. Let N be the number of simulations $\{N = 1, 2, ..., 10000\}$.
2. For each N simulation:
    - Generate 1000 Bernoulli random variables to model the policyholder claims.
    - For each nonzero claim, generate an $Exponential(\lambda = \frac{1}{800})$.
    - Then, we sum up the claim amounts.
3. To end, estimate the probability by counting the proportion of times the sum exceeds $50,000$.

Putting it in practice

```r
set.seed(202425)
N <- 10000
meanClaim <- 800
lambda <- 1 / meanClaim
claimProb <- 0.05
threshold <- 50000

exceedProb <- replicate(N, {
  claims <- rbinom(1000, 1, claimProb)
  numClaims <- sum(claims)
  if (numClaims > 0) {
    claimAmounts <- rexp(numClaims, rate = lambda)
    totalClaim <- sum(claimAmounts)
  } else {
    totalClaim <- 0
  }
  totalClaim > threshold
})

estimatedProbability <- mean(exceedProb)
print(estimatedProbability)
```

```
[1] 0.1075
```

Problem 19

Show how to generate a random variable with the distribution function

$$F(x) = \frac{1}{2}(x + x^2), \quad 0 \leq x \leq 1$$

Part A: Inverse Transform Method

Pseudocode

1. We need to solve the expression $F(x) = u$ to obtain x.
2. Afterwards, we rearrange the expression to get $x = \frac{-1+\sqrt{1+8u}}{2}$ for u from $Uniform(0, 1)$.
3. Return the output x.

Putting it in practice

```
set.seed(202425)
u <- runif(1)
x <- (-1 + sqrt(1 + 8 * u)) / 2
x
```

```
[1] 0.882196
```

Problem 19

Part B: The Rejection Method

Pseudocode

1. Set $M = \frac{3}{2}$ and use $g(x) = 1$ for $x \in [0, 1]$.
2. Generate $X \sim Uniform(0, 1)$.
3. Generate $U \sim Uniform(0, 1)$.
4. Accept X if $U \leq \frac{1+2X}{3}$, otherwise, repeat from step 2.

Putting it in practice

```
set.seed(202425)
M <- 1.5
repeat {
  X <- runif(1)
  U <- runif(1)
  if (U <= (1 + 2 * X) / 3) break
}
X
```

```
[1] 0.830233
```

Problem 24

In Example 5f, we simulated a normal random variable using the rejection technique with an exponential distribution. Show that the number of iterations is minimized when $\lambda = 1$.

Pseudocode

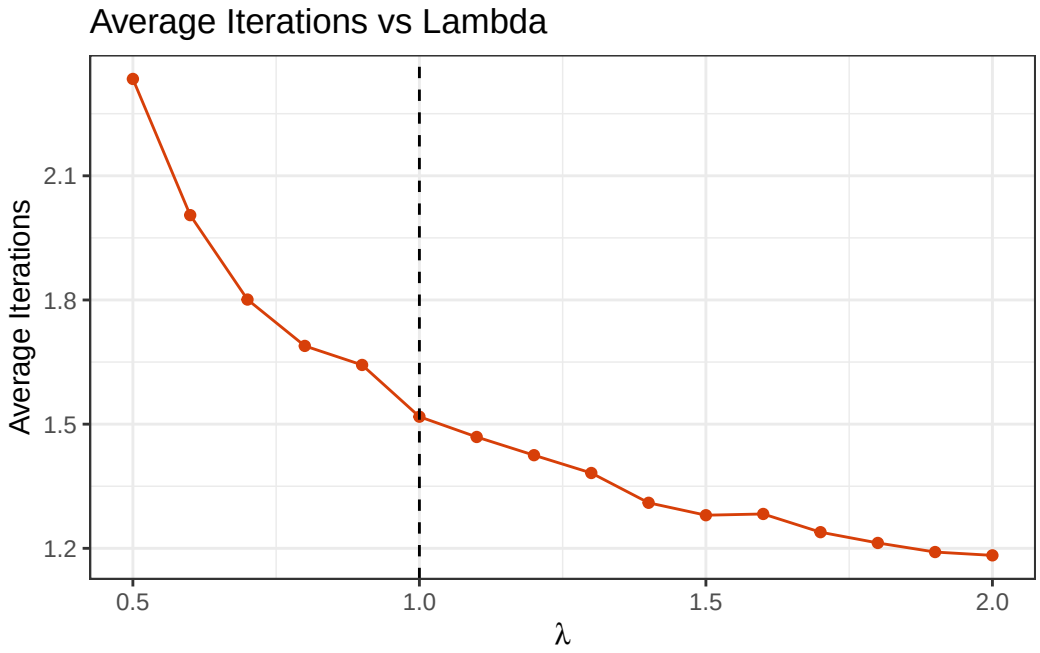
1. Let $g(x) = \lambda e^{-\lambda x}$.
2. To minimize iterations, compare acceptance rates across different λ values.
3. Generate random samples using the rejection sampling method for various λ values.
4. Save results to a dataset & plot results

Putting it in practice

```r
library(purrr)
library(ggplot2)
set.seed(202425)
rejectionIterations <- function(lambda, nTrials = 1000) {
  map_dbl(1:nTrials, function(i) {
    accept <- FALSE
    iterationCount <- 0
    while (!accept) {
      x <- rexp(1, rate = lambda)
      u <- runif(1)
      if (u <= exp(-(x^2) / 2)) accept <- TRUE
      iterationCount <- iterationCount + 1
    }
    return(iterationCount)
  }) %>% mean()
}

lambdaValues <- seq(0.5, 2, by = 0.1)
iterationsPerLambda <- map_dbl(lambdaValues, rejectionIterations)
df <- data.frame(lambda = lambdaValues, iterations = iterationsPerLambda)

ggplot(df, aes(x = lambda, y = iterations)) +
  geom_line(color = "#D73F09") +
  geom_point(color = "#D73F09") +
  geom_vline(xintercept = 1, linetype = "dashed", color = "#000000") +
  labs(x = expression(lambda), y = "Average Iterations",
       title = "Average Iterations vs Lambda") +
  theme_bw()
```


Average Iterations vs Lambda
Average Iterations
2.1
1.8
1.5
1.2
0.5
1.0
1.5
2.0
λ