Computational Statistics, M1 MAS DS, Aix-Marseille University

Houssam BOUKHECHAM

December 27, 2024

Contents

1 Random variable simulation

Are the numbers generated with a Python or R command truly random? In this section, two methods for simulating random variables are presented: the *Transformation* method and the *Accept-Reject* method. The key idea behind both methods is to begin with a uniform random variable in $(0, 1)$ as a foundation for generating samples from more complex distributions. One approach to obtaining such a uniform random variable is through the use of chaotic dynamical systems, such as the *logistic map* or the *doubling map*, which exhibit randomness in their behavior. The interested reader can find more motivations and details in [1].

1.1 Transformation methods

Lemma 1. *Let $F : \mathbb{R} \to [0, 1]$ be an non-decreasing function. If a random variable X has F as its cumulative distribution function (CDF), then the random variable $U = F(X) \sim U(0, 1)$.*

Proof. □

Example 1 (Normal variable generation)**.** *The cumulative distribution function (CDF) of a Gaussian random variable with mean μ and standard deviation σ is given by:*

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2}\, dt. \tag{1}$$

The function F is a diffeomorphism. Assuming $\mu = 0$ and $\sigma = 1$, an approximation F_a^{-1} of the inverse function F^{-1} can be computed to arbitrary precision ([1], Example 2.6). To generate a random sample of size n from the standard Gaussian distribution, we first generate n random samples from the uniform distribution, $\{u_1, u_2, \dots, u_n\}$. Then, we map this sample to the Gaussian distribution using the inverse approximation :

$$\left\{F_a^{-1}(u_1), F_a^{-1}(u_2), \dots, F_a^{-1}(u_n)\right\}.$$

Exercise 1.

1. *Generate a sample $\mathcal{S} = \{u_1, u_2, \cdots, u_n\}$ of size $n = 500$ from the uniform distribution.*

2. *Implement the function*

$$F_a^{-1}(u) = t - \frac{a_0 + a_1 t}{1 + b_1 t + b_2 t^2}, \quad u \in (0, 1),$$

   where $t^2 = \log\left(u^{-2}\right)$ and $a_0 = 2.30753, a_1 = 0.27061, b_1 = 0.99229, b_2 = 0.04481$

3. *Plot the histogram of the set $F_a^{-1}(\mathcal{S})$ and comment the results.*

4. ♣ *Repeat the process for a larger value of n (e.g., $n = 50000$). Compare the generated sample with a standard Gaussian random variable generator and comment the results.*

Exercise 2.

1. *Assume we can simulate $\mathcal{N}(0,1)$, how to generate a sample from $\mathcal{N}(\mu,\sigma^2)$?*
2. *Implement this method in R or Python.*

Exercise 3. *Let $\lambda > 0$ and U be a random variable uniformly distributed on the interval $[0,1]$.*

1. *Prove that the random variable defined by $X = -\frac{\log(U)}{\lambda}$ follows an exponential distribution with scale parameter λ, denoted as $\mathcal{E}xp(\lambda)$.*
2. *Using the uniform distribution and the result from the previous question, generate a sample of size $n = 100$ from $\mathcal{E}xp(1)$.*
3. *Compare the generated sample with a sample produced by a function in R or Python.*

Exercise 4. *Let N be a random variable with a Poisson distribution $\mathcal{P}(\lambda)$, and let X_i be i.i.d. random variables with an $\mathcal{E}xp(\lambda)$ distribution.*

1. *Prove that*
$$Pr(N = k) = Pr(X_1 + \cdots + X_k \leq 1 < X_1 + \cdots + X_{k+1}).$$
2. *Using the previous question, outline the steps to simulate a Poisson distribution $\mathcal{P}(\lambda)$, and then implement your algorithm in R or Python.*
3. ♣ *Do you recommend this method when λ is large ?*

Exercise 5.

1. *Sketch how to generate a discrete random variable using the uniform distribution.*
2. *Generate a sample of size 30 from the binomial distribution with 10 trials and a success rate of 0.3, denoted by $B(n,p)$, where $n = 10$ and $p = 0.3$.*

1.2 Accept-reject method

In the previous subsection, we discussed how to generate Gaussian and exponential distributions from a uniform distribution, but in doing so, we needed an approximation of the inverse of the CDF. In general, the analytic form of the inverse of the CDF is not available, and even if the analytic form exists, approximation of this function may be computationally expensive. An alternative to transformation methods, is the *Accept-Reject* method. Roughly speaking, this method is a technique for generating a sample from a target distribution with density f, when direct sampling from it is not possible. Instead, we use a proposal distribution with density g, from which we generate x, and depending on the values of $g(x)$ and $f(x)$, we either accept x as a sample of f or reject it. More precisely if f has a compact support and is bounded then we have

Theorem 1 (Fundamental theorem of simulation)**.** *Let f be a target density. Then simulating $X \sim f$ is equivalent to simulating*

$$(X, U) \sim \mathcal{U}\left\{(x, u) \ : \ 0 < u < f(x)\right\}. \tag{2}$$

To simulate $X \sim f$, we first choose a value M bigger than the maximum of f. Next, we generate a pair (x, u) from a uniform distribution over the rectangle $[a, b] \times [0, M]$, where $[a, b]$ is the support of f. The value x is accepted if $u \leq f(x)$; otherwise, it is rejected.

Example 2 (Generation of Beta$(2, 4)$ sample)**.** *The target density f of Beta$(2, 4)$ is given by*

$$f(x) = \begin{cases} 20x(1-x)^3, & \text{if } x \in (0,1), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

So, if we take $M = \frac{11}{5}$, then $f \leq M$. If we apply the Accept-Reject algorithm for 1000 iteration, we get Figure 1.

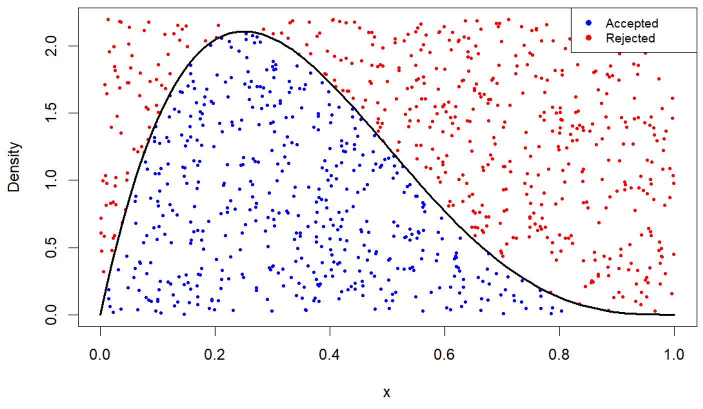

Figure 1: Generating Beta(2,4) using the Accept-Reject method. The Red point are rejected, while the blue ones are accepted.

Exercise 6.

1. *Using the Accept-Reject method, generate a sample of size $n = 1000$ from Beta(3,5).*

2. *How many iteration do we need on average to generate n sample ?*

Now, if we assume that the target density does not have a compact support (f is equal to zero outside an interval $[a, b]$), then we can't apply the previous technique. So we have to bound f by a distribution g which we can sample from (for example a Normal distribution), more precisely Theorem 1 implies

Corollary 1. *Let f be a target density and g a density which we can sample from. Assume that there is $M \in \mathbb{R}$ such that*

$$f(x) \leq Mg(x), \ \forall x.$$

Then to simulate $X \sim f$, we simulate

$$Y \sim g, \quad U|Y \sim \mathcal{U}\big(0, M \cdot g(Y)\big),$$

until $u < f(y)$.

Using Corollary 1, the Accept-Reject method can be done as follows

Algorithm 1 Accept-Reject

```
1: Choose $M \geq 1$ such that $f(x) \leq Mg(x)$ for all $x$.
2:  Generate $X \sim g$ and $u \sim \mathcal{U}[0, 1]$.
3: If $u \leq \frac{f(X)}{Mg(X)}$, accept $Y = X$ and **return** $Y$.
4: Otherwise, go back to step 2.
```

Exercise 7.

1. *Prove that $M \geq 1$.*

2. *Using Corollary 1, prove that Algorithm 1 simulates a sample with distribution f.*

3. *Is it possible to apply the Accept-Reject method if f is known only up to a multiplicative constant?*

Example 3. *Assume we know how to sample from the normal distribution (Using transformation method), and we want to sample from the distribution*

$$f(x) \propto \left(\cos^2(x) + 1 + 2\sin^2(2x)\right) e^{-\frac{1}{2}x^2}.$$

In this case

$$f(x) \leq 4 \cdot \frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}} = 4 \cdot g(x), \ \forall x \in \mathbb{R}, \tag{4}$$

where g is the probability density function of the standard normal distribution. To simulate $X \sim f$, we first generate y from the standard normal distribution $\mathcal{N}(0, 1)$, then generate u from $\mathcal{U}(0, 4 \cdot g(y))$. If $u < f(y)$, the value y is accepted as a sample from f; otherwise, it is rejected.

Exercise 8.

1. *Prove inequality* (4).

2. *Using the Accept-Reject method, generate a sample of size* $n = 2000$ *with distribution* $f(x) \propto \left(\cos^2(x) + 1 + 2\sin^2(2x)\right) e^{-\frac{1}{2}x^2}$.

3. *Is it possible to simulate a chi-squared distribution with 3 degrees of freedom,* $\chi^2(3)$, *using the normal distribution?*

2 Importance sampling

Let f be a probability density function, and assume we are interested in computing the quantity:

$$\mu_f = \int h(x)f(x)\ dx, \tag{5}$$

where h is a problem-specific function (e.g., a loss function in Bayesian statistics). A natural approach to approximate (5) is to use the *Monte Carlo* method. This involves drawing a sample $X_1, \ldots, X_n$ from f and estimating μ_f using:

$$\hat{\mu}_f = \frac{1}{n}\sum_{i=1}^{n} h(X_i). \tag{6}$$

A major challenge in this case is that we may not know how to sample from f. Additionally, even if sampling from f is possible, the function h might have a very small support relative to f. As a result, a large sample size n is required to achieve a good approximation.

Example 4. *Consider f, the probability density function (PDF) of the standard normal distribution, and let $h(x) = I_{[2.5,+\infty)}(x)$, where I_A is the indicator function defined as $I_A(x) = 1$ if $x \in A$ and $I_A(x) = 0$ otherwise.*

The quantity of interest in this case is:

$$\mu_f = \int_{[2.5,+\infty)} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\ dx = 1 - \Phi(2.5),$$

where Φ denotes the cumulative distribution function (CDF) of f. Using a direct Monte Carlo estimation requires a large number of iterations to achieve an accuracy of 0.001. In contrast, importance sampling achieves this accuracy with fewer iterations, as demonstrated in Figure 4.

Definition 1. *Importance Sampling is a method used to evaluate the quantity in (5). This is achieved by generating a sample $X_1, \ldots, X_n$ from an instrumental distribution g, and then using the following approximation:*

$$\mu_f = E_f\left[h(X)\right] \approx \frac{1}{n}\sum_{i=1}^{n} h(X_i)\frac{f(X_i)}{g(X_i)} := \hat{\mu}_f. \tag{7}$$

Using LLN, $\hat{\mu}_f$ converges almost surely to μ_f, in particular the estimator $\hat{\mu}_f$ is *non-biased.*

Exercise 9.

1. *Reproduce the estimations shown in Figure 4 (Use $\mathcal{E}xp(1)$ as an instrumental distribution).*

2. $\clubsuit$ *How many iterations are required in each case to achieve a precision of 0.0001?*

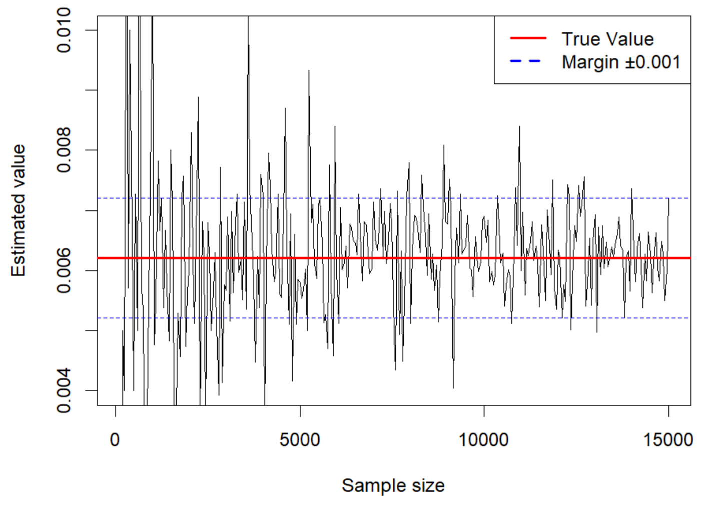

Figure 2: Estimating tail probability with Monte Carlo.

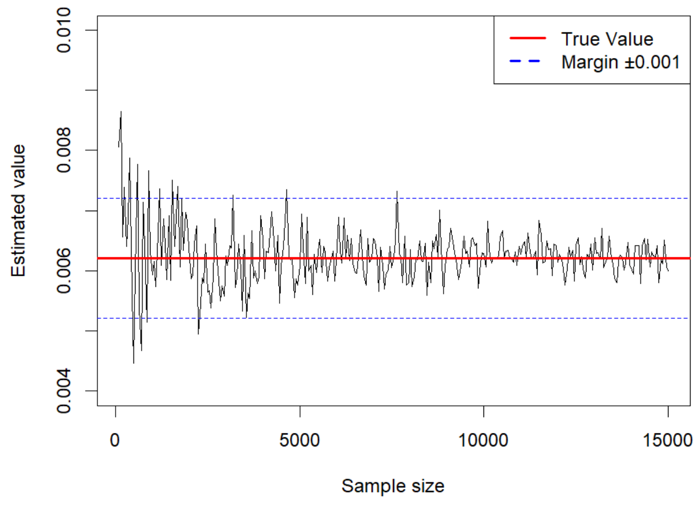

Figure 3: Estimating tail probability with Importance Sampling.

Assume we know the density f only up to multiplicative constant (for example the posterior density), in this case we can't use formula 7, an alternative is to consider the estimation

$$\tilde{\mu}_f := \frac{\sum\limits_i \frac{f(X_i)}{g(X_i)} h(X_i)}{\sum\limits_i \frac{f(X_i)}{g(X_i)}}. \tag{8}$$

Notice that in equation (8), the function f is required only up to a multiplicative constant. This estimator also converges almost surely to μ_f. Interested readers can find further details and developments on the topic of importance sampling in [2].

Example 5. *In this example, we aim to approximate the mean of Gamma$(2.5, 1)$. Although the mean can be computed directly as 2.5, we use importance sampling for illustrative purposes. The instrumental distribution chosen is $\mathcal{N}(2, 2)$.*

A function proportional to the PDF of Gamma$(2.5, 1)$ can be implemented in R as follows:

```r
set.seed(12)

# Define the target distribution (Gamma distribution)
target_dist <- function(x, shape = 2.5, rate = 1) {
  ifelse(x > 0, x^(shape - 1) * exp(-rate * x), 0)
}

# Define the instrumental distribution (Normal distribution)
instr_dist <- function(x) {
  dnorm(x, mean = 2, sd = 2)  # Normal(2, 2) PDF
}
```

If we denote $\frac{f(X_i)}{g(X_i)}$ by ω_i and set h as the identity map, the approximation in (8) simplifies to:

$$\tilde{\mu}_f = \frac{\sum_i \omega_i X_i}{\sum_i \omega_i}, \tag{9}$$

and the approximation can be implemented as follows

```r
# Generate samples from the proposal distribution
n_samples <- 2000

# proposal sample
ps <- rnorm(n_samples, mean = 2, sd = 2)

# Compute the weights for each sample
weights <- target_dist(ps) / instr_dist(ps)
```

```r
# Estimate the mean of the target distribution using IS
importance_sampling_mean <- sum(weights * ps) / sum(weights)
print(importance_sampling_mean)

## [1] 2.487901
```

Exercise 10. *Using importance sampling:*

1. *Approximate the variance of Gamma*$(2.5, 1)$ *to a precision of 0.001.*
2. ♣ *Give an approximation of* $P(X > 5.298317)$*, where* $X \sim$ *Gamma*$(1, 1)$*, to a precision of 0.001.*

3 Bootstrap

Exercise 11. *Let $S = \{x_1, x_2, \ldots, x_n\}$ be a random sample from the uniform distribution on the interval $(0, \theta)$. Assume we want to estimate the unknown parameter θ, so we use the estimator $X_{(n)} = \max x_i$.*

1. *If $X_1, X_2, \cdots, X_n$ are iid with uniform distribution on $(0, \theta)$, what is the distribution of the random variable $X_{(n)} = \max X_i$?*

   (Hint: Determine the cumulative distribution function of $X_{(n)}$)

2. *Is the estimator $X_{(n)}$ biased?*

3. *How would you use the bootstrap to estimate the bias in $X_{(n)}$ for θ?*

4 Gibbs algorithms for bayesian statistics

4.1 Metropolis Hastings algorithm

4.2 MCMC

References

[1] Robert Casella and Roger L. Berger. *Monte Carlo Statistical Methods.* Springer, New York, 2nd edition, 1999.

[2] Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.