Chapter 1 - Random Number Generation

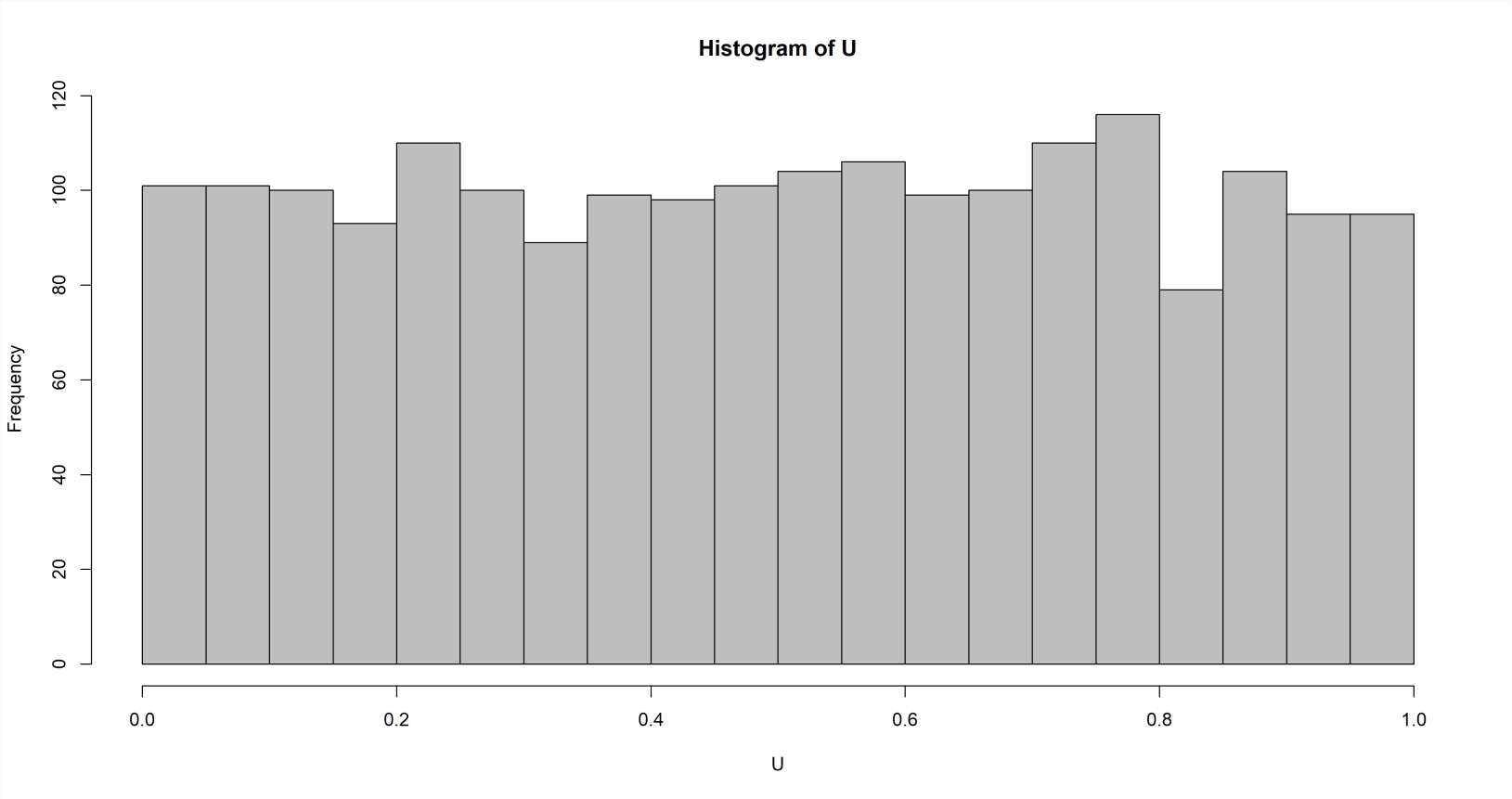
Rejection Sampling

Prof. Alex Alvarez, Ali Raisolsadat

School of Mathematical and Computational Sciences
University of Prince Edward Island

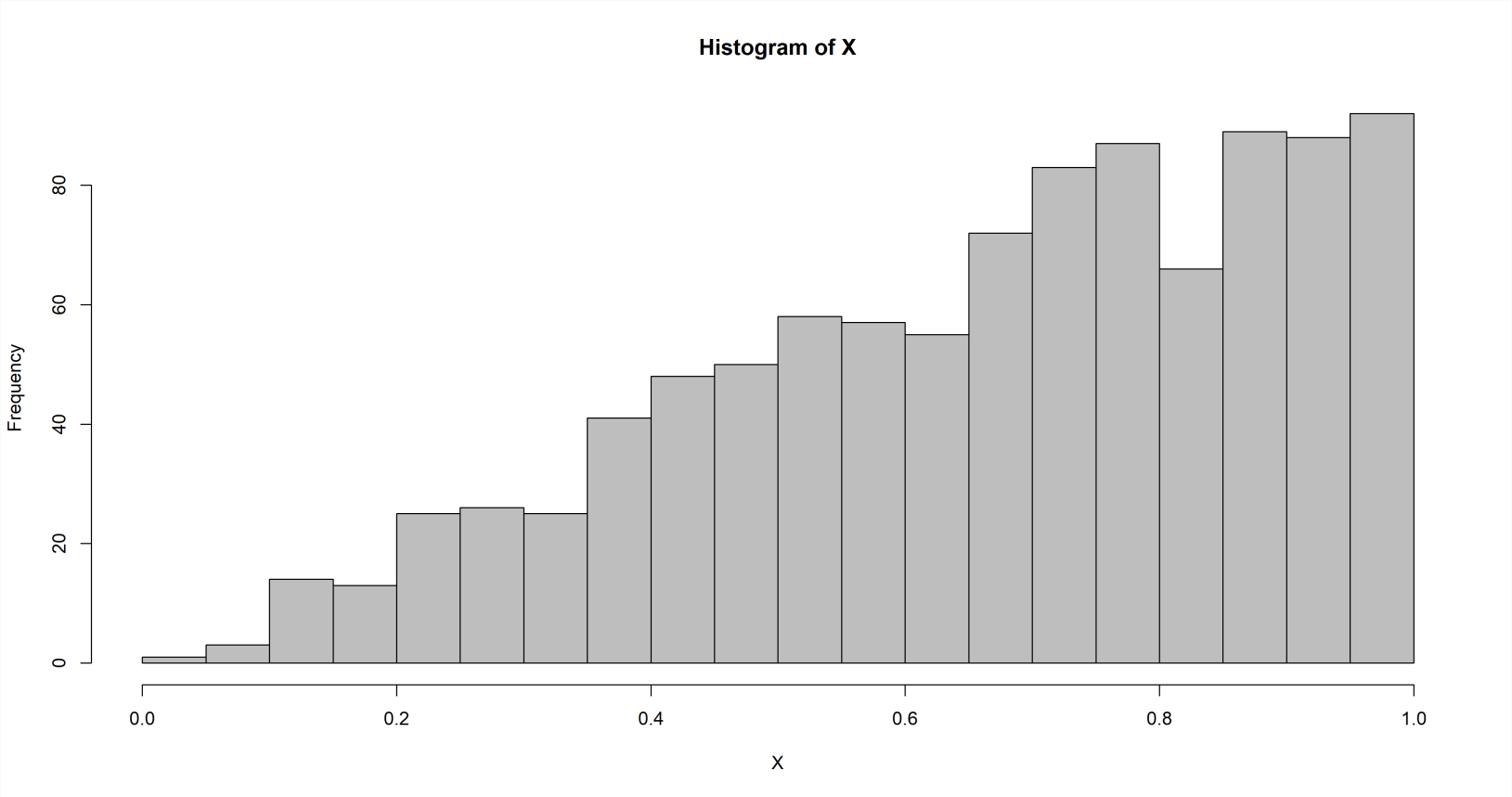
Review: Inverse Transform Method

The following graph corresponds to a histogram of a sample of 2000 generated random numbers uniformly distributed in $[0, 1]$. Let's refer to this as the U distribution.



Review: Inverse Transform Method

The following graph corresponds to a histogram of a sample of 1000 generated random numbers distributed according to the p.d.f. given by $f(x) = 2x$ in $[0, 1]$ and 0 otherwise. The inverse transform method was used to generate the sample. Let's refer to this as the X distribution.



Introduction to Rejection Sampling

Observation:

At an intuitive level, it seems that we can get a sample similar to the one in the histogram of the X Distribution by discarding/rejecting some of the values in the sample taken from the U Distribution.

If we were to do this in this example, it seems that we should reject more of the smaller individual values and less of the larger individual values from the *U* sample.

This informal reasoning is the main idea behind the method known as **Rejection Sampling**.

Rejection Sampling

Suppose that our objective is to **generate a sequence of random numbers that follow a distribution with density** f. **Let's refer to** f **as the target density**.

We will start with a sequence of random numbers from a distribution with density g (the proposal density) which we know how to generate and we will **reject** some of these generated random numbers in order to achieve a new sequence of random numbers that follows the target distribution.

Introduction to Rejection Sampling

Questions:

- Is it always possible to achieve the target distribution in this way?
- How do we decide which random numbers are accepted/rejected?

Introduction to Rejection Sampling

Questions:

- Is it always possible to achieve the target distribution in this way?
- How do we decide which random numbers are accepted/rejected?

In order to achieve the target distribution we should start with a proposal distribution that can take the same values as the target distribution. For example, if the target distribution is the exponential distribution, which takes values in the interval $(0, \infty)$, we cannot use the uniform distribution in $[0, 1]$ as the proposal distribution, as we will never be able to get a proposal random number greater than 1.

Introduction to Rejection Sampling

Questions:

- Is it always possible to achieve the target distribution in this way?
- How do we decide which random numbers are accepted/rejected?

In order to achieve the target distribution we should start with a proposal distribution that can take the same values as the target distribution. For example, if the target distribution is the exponential distribution, which takes values in the interval $(0, \infty)$, we cannot use the uniform distribution in $[0, 1]$ as the proposal distribution, as we will never be able to get a proposal random number greater than 1.

The algorithm that follows answers the question of how to reject some of the proposal random numbers in order to achieve a sample that follows the target distribution.

Basic Rejection Sampling Algorithm

Consider a target density f and a proposal density g such that there exists a function p taking values in $[0, 1]$ such that

$$f(x) = \frac{1}{Z} p(x)g(x) \quad \text{where } Z = \int p(x)g(x)dx$$

Algorithm

To generate one random number from the target distribution:

1. Generate Y (the proposal) with density g
2. Generate $U \sim U[0, 1]$
3. **if** $U \leq p(Y)$ then
   $X = Y$ (meaning that Y is accepted)
4. else return to Step 1
5. Return X

Rejection Sampling Proof

Proof. We have

$$P(X = x) = \sum_{n=1}^{\infty} P(\text{reject } n-1 \text{ times, draw } Y = x \text{ and accept it})$$
$$= \sum_{n-=1}^{\infty} P(\text{reject } Y)^{n-1} P(\text{draw } Y = x \text{ and accept it})$$

Then

$$\begin{aligned}
&P(\text{draw } Y = x \text{ and accept it}) \\
&= P(\text{draw } Y = x) P(\text{accept } Y | Y = x) \\
&= g(x) P\left(U \leq p(y) | Y = x \right) \\
&= g(x) P\left(U \leq Z \frac{f(y)}{g(y)} \middle| Y = x \right) \\
&= Z f(x)
\end{aligned}$$

Rejection Sampling Proof

The probability of having a rejection is

$$\begin{aligned} P(\text{reject } Y) &= \sum_{x\in\Omega} P(\text{draw } Y = x \text{ and reject it}) \\ &= \sum_{x\in\Omega} g(x) P\left(U \geq Z\frac{f(y)}{g(y)}\middle| Y = x\right) \\ &= \sum_{x\in\Omega} g(x)\left(1 - Z\frac{f(x)}{g(x)}\right) = 1 - Z \end{aligned}$$

Hence we have

$$\begin{aligned} P(X = x) &= \sum_{n=1}^{\infty} P(\text{reject } n-1 \text{ times, draw } Y = x \text{ and accept it}) \\ &= \sum_{n=1}^{\infty} (1-Z)^{n-1} Z f(x) \\ &= f(x) Z \sum_{n=1}^{\infty} (1-Z)^{n-1} \quad \left(\text{Note: } \sum_{n=1}^{\infty} r^{n-1} = \frac{1}{1-r} \text{ for } |r| < 1\right) \\ &= Z f(x) \frac{1}{1-(1-Z)} = f(x) \end{aligned}$$

Rejection Sampling

Remarks

- The number of proposals used to generate one accepted value is random.
- Actually, the number of proposals used to generate one accepted value has geometric probability distribution with mean $1/Z$
- Z also represents the probability that one proposal is accepted, so the higher the value of Z the most efficient is the algorithm.

Rejection Sampling Example

The initial example in these slides correspond to the proposal density given by $g(x) = 1$ in $[0, 1]$, and target density $f(x) = 2x$ also in $[0, 1]$. Outside of this interval both densities are null.

We also know that in order to apply the previous algorithm we need a function p taking values in $[0, 1]$ such that

$$f(x) = \frac{1}{Z} p(x) g(x) \text{ where } Z = \int p(x) g(x) dx$$

After substituting f and g we get

$$p(x) = Z \cdot 2x$$

As $0 \leq p(x) \leq 1$ for all $x \in [0, 1]$ then $2Z \leq 1$ or $Z \leq 1/2$.

This means that any value $0 < Z \leq 1/2$ is a valid choice for the algorithm, with $Z = 1/2$ being the best choice.

Code

R Code:

```r
counter <- 1
U <- vector();
A <- vector();
for (i in c(1:2000)){
  U[i] <- runif(1)
  D <- runif(1)
  if (D < U[i]){
    A[counter] = U[i];
    counter <- counter+1;
  }
}
accepted <- counter-1
hist(U)
hist(A)
```

Python Code:

```python
import numpy as np
import matplotlib.pyplot as plt

counter = 0
U = []
A = []

for i in range(2000):
    u = np.random.uniform()
    d = np.random.uniform()
    U.append(u)
    if d < u:
        A.append(u)
        counter += 1

plt.hist(U, bins=20)
plt.show()
```

Envelope rejection sampling

The envelope rejection sampling algorithm can be used even in cases when we start with a non-normalized target density f (meaning that $\int f(x)dx \neq 1$).

If we define $Z_f = \int f(x)dx$, then clearly the function $\tilde{f} = \frac{f}{Z_f}$ is a proper (normalized) density, but in some cases finding Z_f can be challenging.

It is significant that we will still be able to get a sample that follows the density $\tilde{f}$, even in cases where we do not know the exact value of Z_f.

Envelope rejection sampling

Starting with the (non necessarily normalized) target density f and the proposal density g, first find a constant c such that $f(x) \leq cg(x)$ for all x.

Algorithm

1. Generate Y (the proposal) with density g
2. Generate $U \sim U[0, 1]$
3. **if** $U \leq \dfrac{f(Y)}{cg(Y)}$ then
   $X = Y$ (meaning that Y is accepted)
4. else return to Step 1
5. Return X (a random number from normalized density $\tilde{f}$)

Envelope rejection sampling

Remarks:

- The function cg is the "envelope" for f.
- In general, the number of proposals required to generate one accepted random value is geometrically distributed with mean c/Z_f. therefore if we start with a normalized density, we will require c proposals (on average) to generate one accepted value.
- Given that the computational cost of the algorithm is proportional to c, the smaller the value of c the better.
- If $c^* = \sup_{x \in G} \frac{f(x)}{g(x)}$ where $G = \{x : g(x) > 0\}$, then any $c \geq c^*$ can be a valid choice, with c^* being the optimal value.

Envelope Rejection Sampling Example

Example: Use the Envelope Rejection Sampling algorithm to generate a sample of 100 random numbers coming from the (non normalized) density $f(x) = \frac{1}{1+x^4}$ on $(-\infty, \infty)$. Assume that the proposal distribution is the Cauchy distribution with density $g(x) = \frac{1}{\pi(1+x^2)}$.

Envelope Rejection Sampling Example

Example: Use the Envelope Rejection Sampling algorithm to generate a sample of 100 random numbers coming from the (non normalized) density $f(x) = \dfrac{1}{1+x^4}$ on $(-\infty, \infty)$. Assume that the proposal distribution is the Cauchy distribution with density $g(x) = \dfrac{1}{\pi(1+x^2)}$.

Solution: We can check that in this case the function $\dfrac{f(x)}{g(x)}$ attains its maximum at $x^* = \pm\sqrt{\sqrt{2}-1}$. Therefore $c^* = \pi\dfrac{\sqrt{2}}{4-2\sqrt{2}} \approx 3.7922$.

This means that we could use $c = 3.8$ on the algorithm (for example).

Envelope Rejection Sampling Example

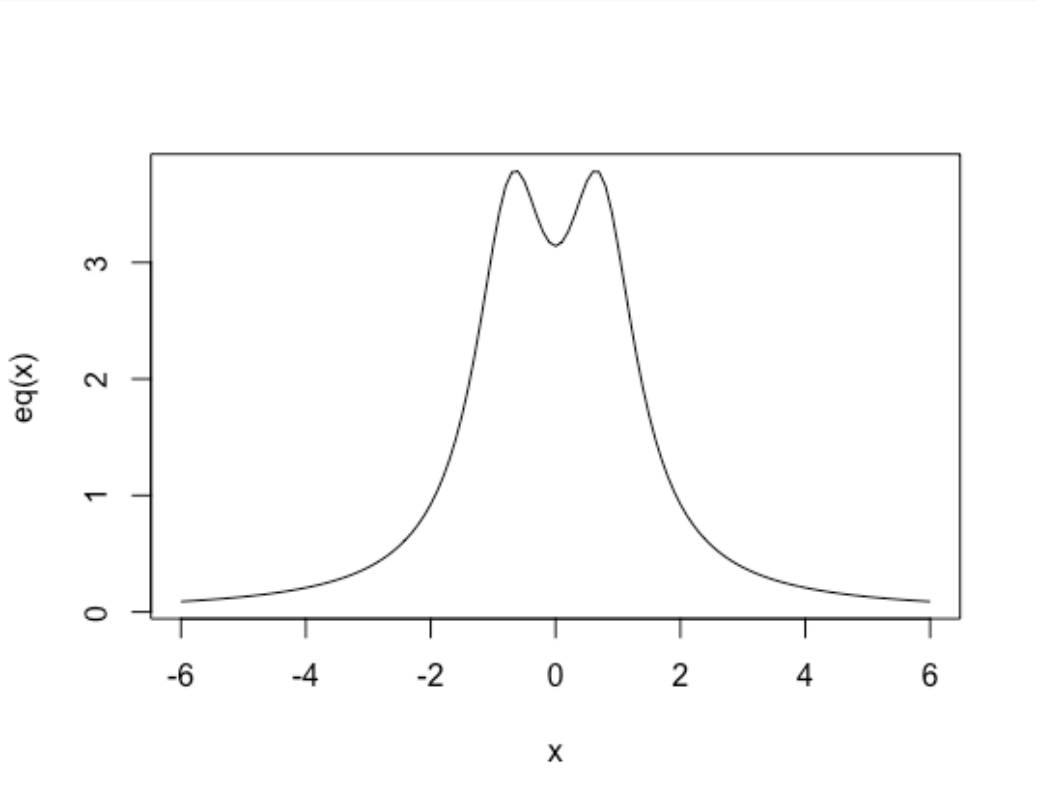
Another way to get a valid value of c is by visual inspection of the graph of f/g

R Code:

```r
x = seq(-6, 6, 0.1);
eq = pi*(1+x^2)/(1+x^4)
plot(x, eq, type='l')
```

Python Code:

```python
x = np.arange(-6, 6.1, 0.1)
eq = np.pi * (1 + x**2) / (1 + x**4)
plt.plot(x, eq)
plt.show()
```



Code

R Code:

```r
n <- 100
counter <- 1
c <- 3.8
target_samples <- vector()

while (counter < n+1) {
   U <- runif(2)
   proposal <- tan(pi*U[1])
   f <- 1 / (1 + proposal^4)
   g <- 1 / (pi * (1 + proposal^2))
   if (U[2] < f/(c*g)) {
        target_samples[counter] <-
            proposal
        counter <- counter+1
   }
}
hist(target_samples)
```

Python Code:

```python
n = 100
c = 3.8
target_samples = []
counter = 0

while counter < n:
    U = np.random.uniform(size=2)
    proposal = np.tan(np.pi * U[0])
    f = 1 / (1 + proposal**4)
    g = 1 / (np.pi * (1 + proposal**
        2))

    if U[1] < f / (c * g):
        target_samples.append(
            proposal)
        counter += 1

plt.hist(target_samples, bins=20)
plt.show()
```

Envelope Rejection Sampling

Remarks

- In the code for the previous example, we used the inverse transform method to generate the proposals according to the Cauchy distribution
- In this case, we know in advance the sample size for the target distribution. Therefore, we do not know how many “proposals” will have to be generated to achieve that. Because of this we use a “while” cycle instead of using a “for” cycle.

Homework

Write a computer program for the implementation of the envelope rejection sampling algorithm to generate a random sample of 1000 random numbers from the distribution given by the density

$$f(x) = \begin{cases} \dfrac{3x^2 + 7x^6}{2} & \text{if } x \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

Use $U \sim U[0,1]$ as the proposal distribution. On average, how many proposals do we need to generate this random sample? Can you suggest a different choice of proposal distribution so that the algorithm is more efficient (fewer proposals rejected)?