DRP Winter 2022: Statistical Simulations

Disclaimer: These aren't slides! Because this topic requires showing a lot of mathematical notation, my slides are of the form of a knitted Rmd file.

Methods of obtaining independent and identically distributed random samples for both continuous and discrete random variable.

Motivation: If we have some kind of computer program, we can ask it to generate any type of distribution we want it to generate, exponential distribution, normal distribution, gamma distribution, but behind the scenes the computer is doing something to generate numbers from that distribution.

Why Sampling? Lack of closed form solution for expectations. Example: estimating the integral of a function that does not have a closed form solution.

The building block of computational simulation is the generation of uniform random numbers. If we can draw from $U(0,1)$, then we can draw from most other distributions. Thus the construction of sampling from U(0, 1) requires special attention. Computers can generate numbers between $(0,1)$, which although are not exactly random (and in fact deterministic), but have the appearance of being U(0, 1) random variables. These draws from $U(0,1)$ are pseudorandom draws.

Pseudorandom number generator

The goal in pseudorandom generation is to draw

$$x_1, x_2, \ldots, x_n \sim U(0,1)\,.$$

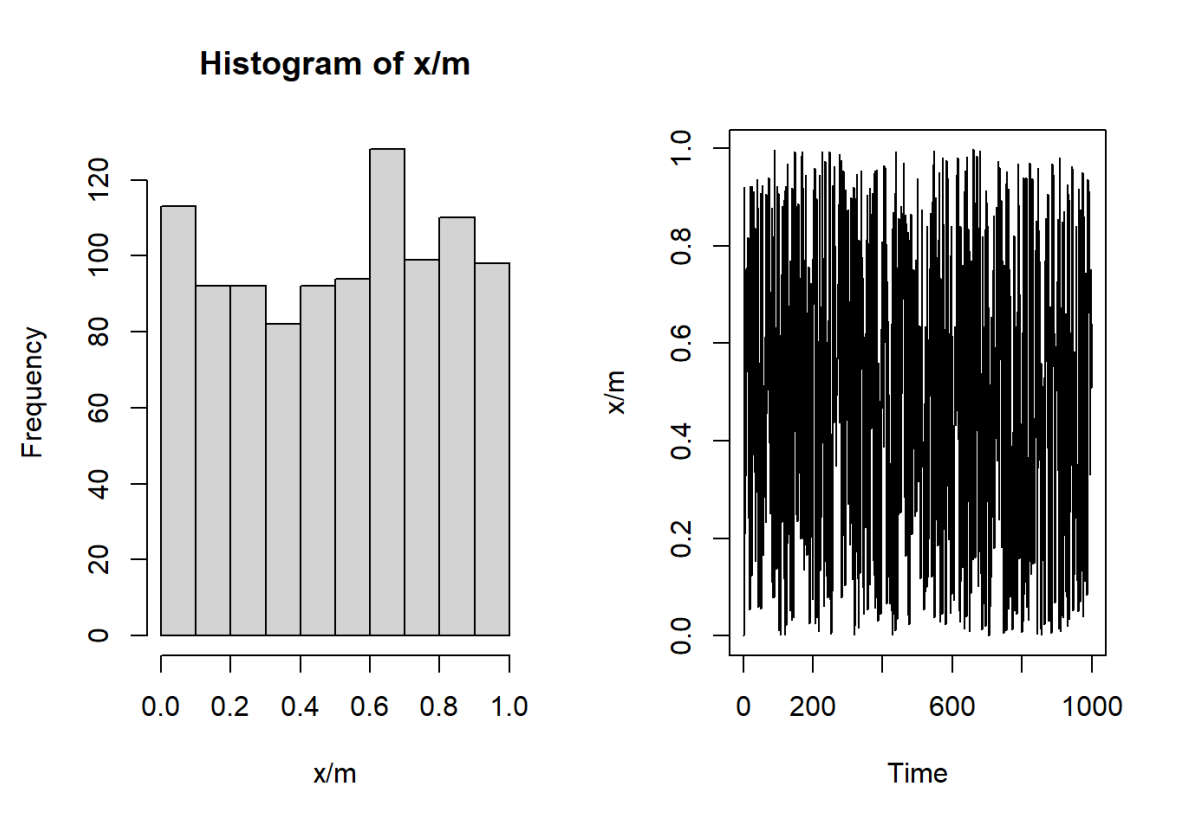
Multiplicative congruential method

1. Set the seed x_0 and select positive integers a and m.
2. Set $x_n = a x_{n-1}$ mod m.
3. Return the sequence $\{x_m/m\}$.

Also note that after some finite number of steps < m, the algorithm will repeat itself, since when a seed x_0 is set, a deterministic sequence of numbers follows.

Now we see an R example with $a = 7^5$ and $m = 2^{31} - 1$.

```
m <- 2^31 - 1
a <- 7^5
x <- numeric(length = 1000)
x[1] <- 7

for (i in 2:1000){
  x[i] <- (a*x[i-1]) %% m
}
par(mfrow = c(1,2))
hist(x/m)
plot.ts(x/m)
```



So we can see that the resulting samples generated are in fact uniform. The left graph shows the distribution of numbers generated from U(0,1) and the left shows the value of the number generated over time and since we can see no clear trend between the number generated and time, we can say that it is random.

Estimating complicated integrals

Consider the integral

$$\theta = \int_0^1 \exp\{e^x\} dx\,.$$

The above integral does not have a standard analytical form. But we are interested in calculating θ. We will turn this mathematical problem into a statistical problem.

If we can get n iid draws $\{x_1, \ldots, x_n\}$ from Unif$[0,1]$, then we can estimation θ using

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} \exp\{e^{x_i}\}\,.$$

```
repeats <- 1e4

## Unif(0,1) case
U <- runif(repeats)
mean(exp(exp(U)))
```

```
## [1] 6.304136
```

We have our set of random numbers, our uniform distribution, so now we can use the uniform distribution as a basis for generating the distribution we want

Simulation Methods

– Inverse transform

– Accept-reject

Discrete Random Variables

Examples of discrete distributions include Bernoulli, Poisson, Geometric, etc. To draw random samples from such a discrete distribution, we will study the following methods

Inverse transform

Suppose X is a discrete random variable with the probability density function

$$f(X = x_j) = p_j \; j = 0, 1, \ldots n \; \sum_{j=1}^{n} p_j = 1$$

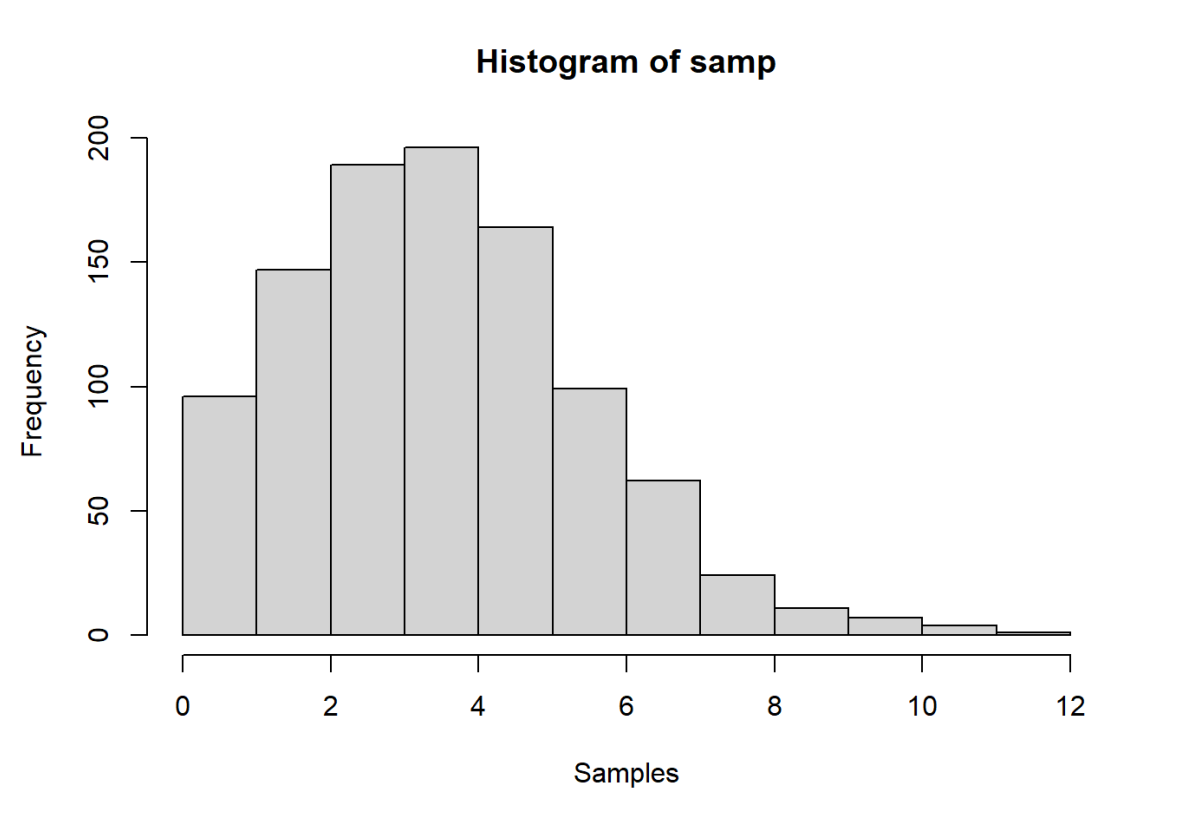
Algorithm:

1. Draw $U \sim \text{Unif}[0,1]$
2. if $U < p_0$, then
3. $X = 0$
4. else if $U < p_0 + p_1$, then
5. $X = 1$
6. ...
7. else if $U < \sum_{i=1}^{j} p_j$, then
8. $X = j$
9. ...

We generate a random number U, and find the interval which U lies on based on the probabilities of p

Example (Poisson distribution): We are going to draw samples from Poisson(λ) where $\lambda = 4$.

```
n <- 1000 ## sample size
lam <- 4 ##
samp <- numeric(n)
for (t in 1:n){
  u <- runif(1)
  i <- 0
  p <- exp(-lam)
  f <- p
  while(u >= f){
    p <- (lam*p)/(i+1)
    f <- f + p
    i <- i+1
  }
  samp[t] <- i
}
hist(samp, xlab = "Samples")
```



Accept-reject

Although we can draw from any discrete distribution using the inverse transform method, you can imagine that for distributions on countably infinite spaces (like the Poisson distribution), the inverse transform method may be very expensive. In such situations, acceptance-rejection sampling may be more reliable.

For accept-reject, suppose we have an efficient method for generating a random variable having pmf q, we can use it as a basis for simulating p. By first simulating a random variable Y having mass function q and accepting this simulated value with a probability proportional to py/qy

Let $\{p_j\}$ denote the pmf of the target distribution with Pr$(X = a_j) = p_j$ and let $\{q_j\}$ denote the pmf of another distribution with Pr$(Y = a_j) = q_j$. Suppose you can efficiently draw from $\{q_j\}$ and you want draw from $\{p_j\}$. Let c be a constant such that

$$\frac{p_j}{q_j} \leq c$$

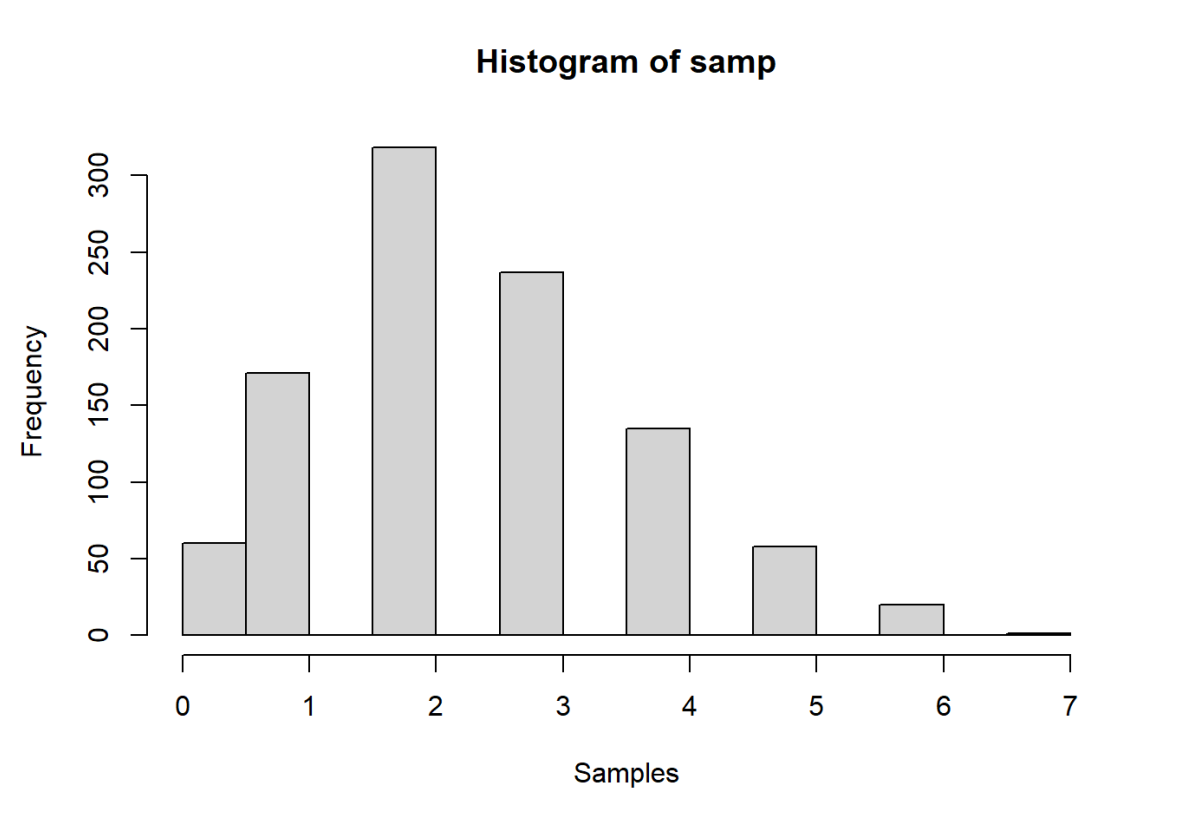
for all j such that $p_j > 0$. If we can find such a $\{q_j\}$ and c, then we can implement an Acceptance-Rejection or Accept-Reject sampler. The idea is to draw samples from $\{q_j\}$ and accept these samples if they seem likely to be from $\{p_j\}$.

Algorithm:

1. Draw $U \sim U[0,1]$.
2. Simulate $Y = y$ with pmf q_j.
3. If $U < \frac{p_y}{c q_y}$ then,
4. Return $X = y$ and stop
5. Else
6. Go to step 1.

Example (Binomial Distribution): In the following code, we draw samples from a Binomial(n, p) using the proposal distribution as Geometric(p) in the AR method. Here we take $n = 10$ and $p = 0.25$.

```
## this function samples ONE sample using AR

draw_binom <- function(n, p)
{
accept <- 0
x <- 0:n
all_c <- choose(n,x) * (1-p)^(n - 2*x) * p^(x-1)
c <- max(all_c) + .001 # final c with slight increase for numerical sta
bility.

while(accept == 0)
{
U <- runif(1)
prop <- rgeom(1, prob = p) #draw proposal
ratio <- dbinom(x = prop, size = n, prob = p)/(c* dgeom(x = prop, prob
 = p))

if(U < ratio)
{
accept <- 1
rtn <- prop
}
}

return(rtn)
}


N <- 1e3 # sample size
samp <- numeric(N)
for(t in 1:N)
{
samp[t] <- draw_binom(n = 10, p = .25)
}
hist(samp, xlab = "Samples")
```



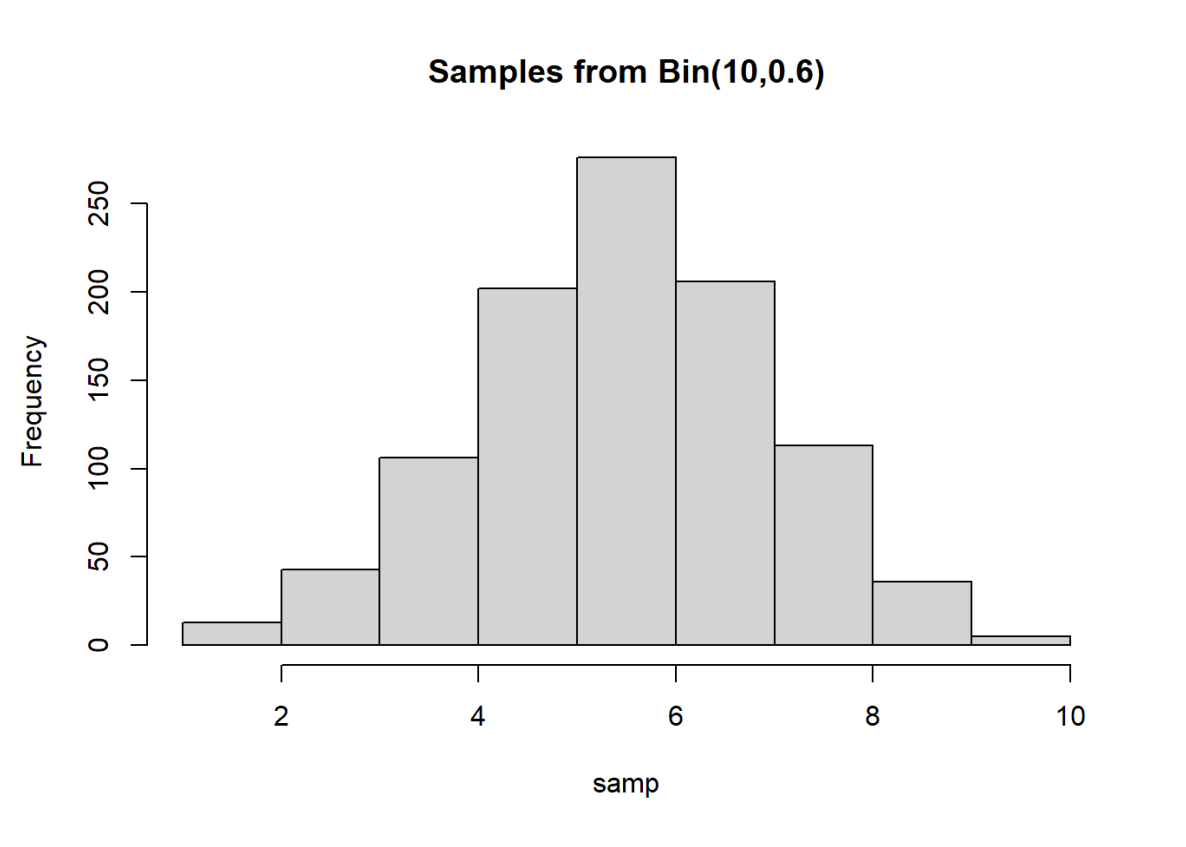
Miscellaneous methods

We can also take advantage of relationships between distributions.

1. Binomial Distribution: We know that if $Y_1, Y_2, \ldots, Y_n \sim^{iid}$ Bern(p), then

$$X = Y_1 + Y_2 + \ldots + Y_n \sim Bin(n, p)$$

Now let's try to get **1000** samples from Bin(10, 0.6) using R.



So, we can simulate n Bernoulli variables, add them up, and we have a realization from a Binomial (n, p).

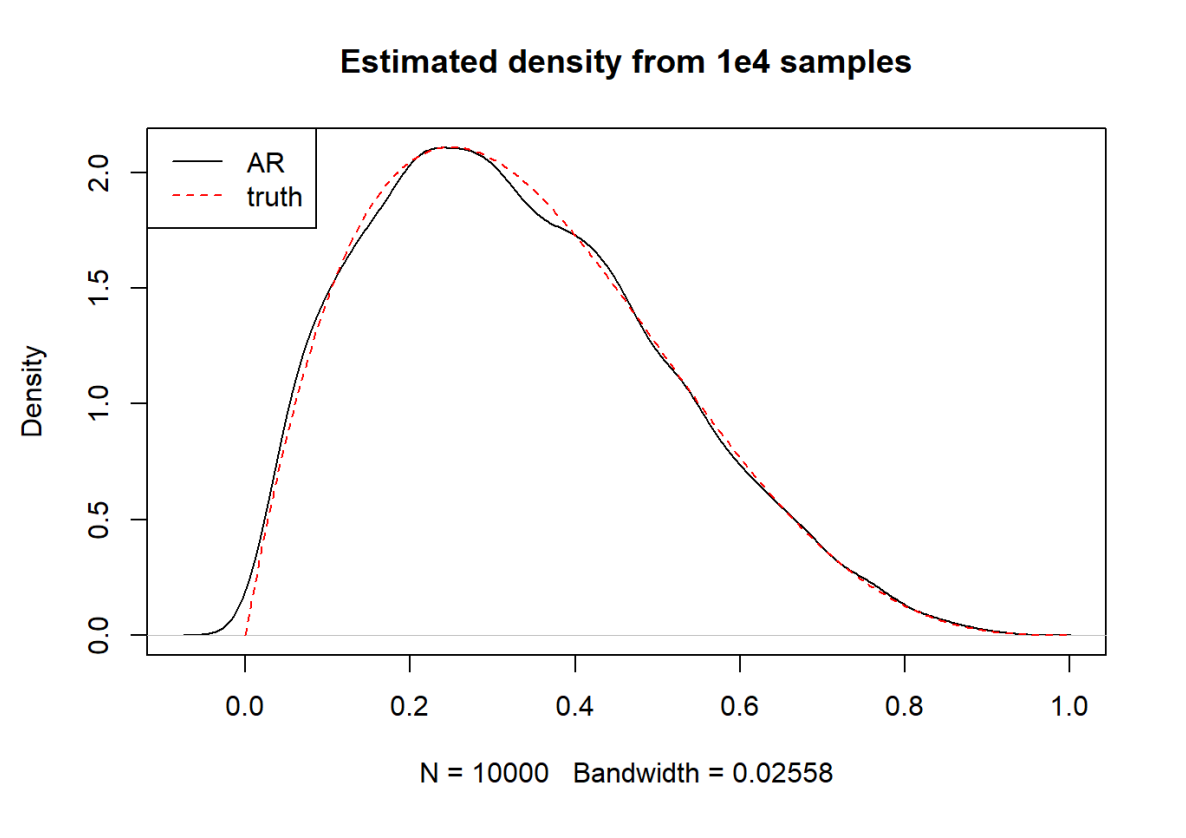
Continuous Random Variables

The same methods as above but sampling continuous random variables instead. We will discuss three sampling methods for continuous random variables.

1. Accept reject
2. Inverse Transform
3. Importance Sampling

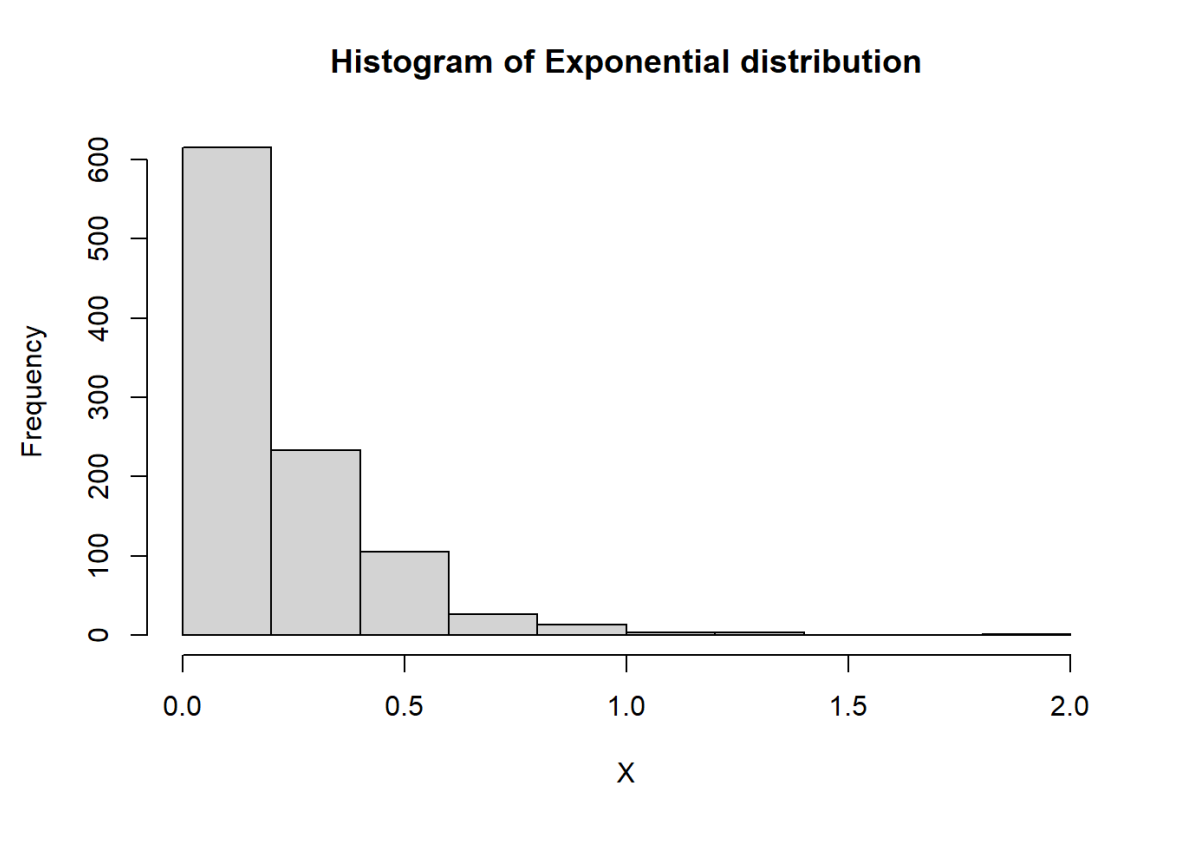
Example: Accept reject for Beta distribution (2,4) (continuous)

```
AR_5d <- function()
{
  accept = 0
  counter = 0
  c = 135/64
  while(accept==0){
    counter = counter + 1
    U_1 = runif(1)
    U_2 = runif(1)
    ratio = (256/27)*U_1*(1-U_1)^3
    if(U_2 <= ratio){
      accept = 1
      return(c(U_1, counter))
    }
  }
}


N <- 1e4
samp <- numeric(length = N)
counts <- numeric(length = N)
for(i in 1:N)
{
  temp <- AR_5d()
  samp[i] <- temp[1]
  counts[i] <- temp[2]
}

x <- seq(0, 1, length = 500)
plot(density(samp), main = "Estimated density from 1e4 samples")
lines(x, dbeta(x, shape1 = 2, shape2 = 4), col = "red", lty = 2)
legend("topleft", lty = 1:2, col = c("black", "red"), legend = c("AR",
                                                                  "trut
h"))
```



Example: Inverse Transform for Exp(λ), $\lambda = 5$

```
N <- 1000
lambda <- 5
U <- runif(N)
X <- - (1/lambda) * log(1 - U)
hist(X, main = "Histogram of Exponential distribution")
```



Importance sampling

Suppose we want to estimate the expectation of the function $h(X)$ where $X \sim P$. In a scenario where $h(X)$ is nearly zero outside of the region A belonging to the domain of random variable X drawing samples directly from P may infact not be a good idea to estimate $\mathbb{E}[h(X)]$. A plain Monte Carlo sample from the distribution of X could fail to have even one point inside the region A. The problem of estimating the expectation of such functions is of significance in applications like nuclear physics, finance, insurance, etc.

Therefore, we instead want to sample from a distribution that can give samples that are more representative of the region A. We do this by sampling from a distribution that overweights the important region, hence the name importance sampling. This distribution is called the proposal distribution and we are going to denote it by Q with density function $q(X)$.

Suppose we are interested in estimating the expectation of a function $h: X \to \mathbb{R}$ with respect to p. That is, we want to estimate

$$\theta = \mathbb{E}_p[h(X)] = \int_{\chi} h(x)p(x)dx\,.$$

We assume that θ is finite. Consider a proposal distribution with density $q(X)$. We will instead construct an estimator that draws samples from q and uses them to estimate θ.

Simple Importance Sampling

Consider

$$\begin{aligned}\mathbb{E}_p[h(X)] &= \int_{\chi} h(x)p(x)dx \\ &= \int_{\chi} h(x)\frac{p(x)}{q(x)}q(x)dx \\ &= \mathbb{E}_q\left[\frac{h(X)p(X)}{q(X)}\right]\end{aligned}$$

Therefore if $X_1, \ldots, X_n$ are samples from Q, then an estimator for θ is

$$\hat{\theta}_s = \frac{1}{n}\sum_{j=1}^{n}\frac{h(X_i)p(X_i)}{q(X_i)}$$

The estimator $\hat{\theta}_s$ is the importance sampling estimator, the method is called simple importance sampling and Q is the importance distribution.

Example: Gamma distribution: We want to estimate the second moment for a gamma distribution using an exponential distribution as the proposal. That is $h(x) = x^k$, $P = \text{Gamma}(\alpha, \beta)$ and $Q = \text{Exp}(\lambda)$.

```
alpha = 2
beta = 5
k = 2
truth = alpha/beta^2 + (alpha/beta)^2

lambda = 3
N = 1e4
samp = rexp(N, rate = lambda)
func = samp^k * dgamma(samp, shape = alpha, rate = beta) / dexp(samp, r
ate = lambda)
est = mean(func)
print(paste("Truth: ", truth, "Estimate: ", est))
```

```
## [1] "Truth:  0.24 Estimate:  0.244696886245796"
```

The true expectation and thesimple importance sampling estimate we got from sampling from q are pretty close!

Weighted Importance Sampling

Often for many distributions, we do not know the target distribution fully, but only know it up to a normalizing constant. That is, the target density is

$$p(x) = a\tilde{p}(x)$$

for some unknown constant a and the proposal density is

$$q(x) = b\tilde{q}(x)$$

for some unknown constant b. Suppose the same θ is of interest. Since a and b are unknown, we can't evaluate $p(x)$ and $q(x)$. So if we can estimate a and b as well, that will allow us estimate θ. Instead, we will estimate b/a, which also works!

Consider samples $X_1, \ldots, X_n$ from Q, then the weighted importance sampling estimator for θ is

$$\hat{\theta}_w = \frac{\sum_{i=1}^{n}\frac{h(X_i)\tilde{p}(X_i)}{\tilde{q}(X_i)}}{\sum_{i=1}^{n}\frac{\tilde{p}(X_i)}{\tilde{q}(X_i)}}\,.$$

Optimal Proposal for Importance Sampling

How do we choose the importance distribution q? Note that, one reason to use importance sampling would be to obtain smaller variance estimators than the original. So, if we can choose q such that the variance of importance samples is minimized that would be ideal. Let σ_q^2 denote the variance of importance samples

Theorem 1: The density q^* that minimizes σ_q^2 is:

$$q^*(x) = \frac{|h(x)|p(x)}{\mathbb{E}_p[|h(x)|]}$$

as long as $\mathbb{E}_p[|h(x)|] \neq 0$.

Now for the same Gamma distribution example, we try to find the optimal proposal. Recall $h(x) = x^k$ and $P = \text{Gamma}(\alpha, \beta)$. Using the theorem above

$$\begin{aligned}q^*(x) &\propto |h(x)|p(x) \\ &\propto |h(x)|\tilde{p}(x) \\ &= x^k x^{\alpha-1}\exp(-\beta x) \\ &= x^{\alpha+k-1}\exp(-\beta x)\,.\end{aligned}$$

This shows that the optimal proposal would be $\text{Gamma}(\alpha + k, \beta)$.

In the following code, we calculate the importance sample estimate of θ firstly using $Q = \text{Gamma}(\alpha, \beta)$ and then using the proposal $Q = \text{Gamma}(\alpha + k, \beta)$. We will show how the variance of importance samples is significantly smaller in the latter case.

```
############################################
### Optimal importance sampling from Gamma
############################################

set.seed(1)
# Function does importance sampling to estimate second moment of a gamm
a distribution 3
imp_gamma <- function(N = 1e3, alpha = 4, beta = 10, moment = 2, imp.al
pha
= alpha + moment)
{
fn.value <- numeric(length = N)
draw <- rgamma(N, shape = imp.alpha, rate = beta) # draw imporance samp
les
fn.value <- draw^moment * dgamma(draw, shape = alpha, rate = beta) /
dgamma(draw, shape = imp.alpha, rate = beta)
return(fn.value) #return all values
}
N <- 1e4
# Estimate 2nd moment from Gamma(4, 10) using Gamma(4, 10)
# this is IID Monte Carlo
imp_samp <- imp_gamma(N = N, imp.alpha = 4)
mean(imp_samp)
```

```
## [1] 0.2002069
```

```
# [1] 0.2002069
var(imp_samp)
```

```
## [1] 0.04421469
```

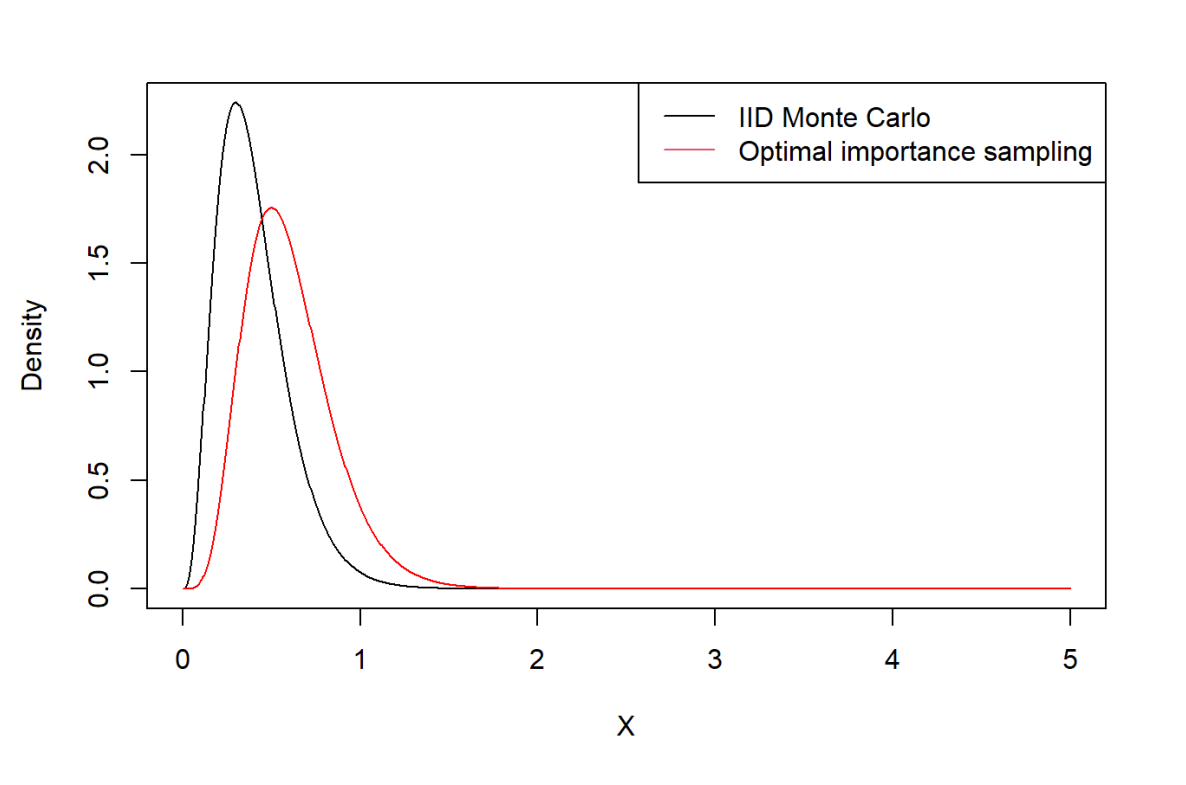
```
# [1] 0.04421469
# Estimate 2nd moment from Gamma(4, 10) using Gamma(6, 10)
# this is the optimal proposal
imp_samp <- imp_gamma(N = N)
mean(imp_samp)
```

```
## [1] 0.2
```

```
# [1] 0.2
var(imp_samp)
```

```
## [1] 9.620443e-33
```

```
# [1] 9.620212e-33
# why is the estimate good
foo <- seq(0.001, 5, length = 1e3)
plot(foo, dgamma(foo, shape = 4, rate = 10), type= "l", ylab = "Densit
y", xlab = "X")
lines(foo, dgamma(foo, shape = 6, rate = 10), col = "red")
legend("topright", col = 1:2, lty = 1, legend = c("IID Monte Carlo", "O
ptimal importance sampling"))
```



This concludes the project report.

References: Ross, Sheldon M. Simulation. Academic Press, 2013.