Simulation of Random Variables

Learning goals

Understand

- Algorithm to simulate discrete random variables
- Inversion method (algorithm to simulate continuous random variables)
- Rejection methods
- Gibbs sampler

Specifically,

- Understand key ideas for each algorithm
- Know when each algorithm can be used
- Be able to derive and implement each algorithm

Why Monte-Carlo simulation?

Y is built up from X
Y's distribution is complication
X's distribution is know

See the section 'Why Monte-Carlo simulation?' in sampling_example.pdf.

Often we are interested in the distribution of a random variable X which is complicated, but which can none-the-less be built up from simple components such as independent rv's with known distributions.

Monte-Carlo simulation is an excellent tool for such problem: we seek to generate a random sample from the distribution of X, which we can use to estimate its mean, median, mode, percentiles, etc.

The starting point for any simulation is the generation of r.v.s with known distributions (binomial, poisson, exponential, normal, etc.), which are the building blocks for more complicated distributions. It turns out that all random variables can be generated by manipulating $U(0, 1)$ rv's.

Seeding

If the random number generator is initialised (called the seed) before you start generating random numbers, then you can reproduce the whole sequence exactly. This is a very good idea from a scientific point of view; being able to repeat an experiment means that your results are verifiable.

If the random number generator is not initialised, then R initialises it using a value taken from the system clock.

Seeding

To generate n random numbers from $U(0,1)$ in R, use `runif(n)`. For a given value of `seed`, the command `set.seed(seed)` always puts you at the same point on the cycle of random numbers.

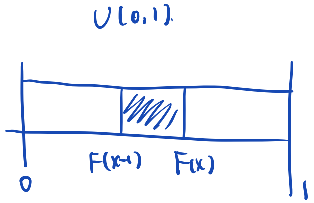
```
> set.seed(42)
> runif(2)
[1] 0.9148060 0.9370754
> runif(2)
[1] 0.2861395 0.8304476
> set.seed(42)
> runif(4)
[1] 0.9148060 0.9370754 0.2861395 0.8304476
```

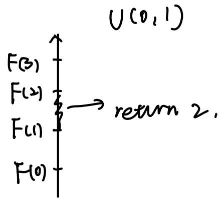
Simulating discrete random variables

Simulating discrete random variables

Let X be a discrete random variable taking values in the set $\{0, 1, \ldots\}$ with cdf F and pmf p. The following snippet of code takes a uniform random variable U and returns a discrete random variable X with cdf F.

```
U ~ U(0,1)
x <- 0
while (F(x) < U) {
    x <- x + 1
}
return x
```





When the algorithm terminates we have $F(x) \geq U$ and $F(x-1) < U$, that is $U \in (F(x-1), F(x)]$. That is,

$$\mathbb{P}(X = x) = \mathbb{P}(U \in (F(x-1), F(x)]) = F(x) - F(x-1) = p(x).$$

Simulating $X \sim$ Binomial$(3, 0.5)$

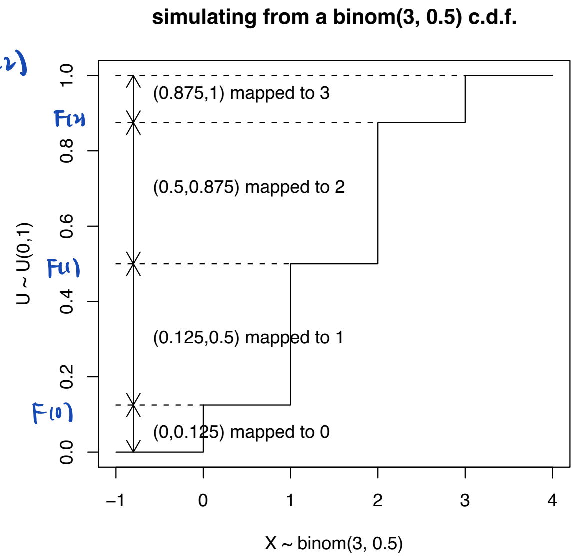
$F_1(0) = P(X=0) = 0.125$

$F_1(1) = P(X=0) + P(X=1)$

$= 0.5$

$F_1(2) = P(X=0) + P(X=1) + P(X=2)$

$= 0.875$



$U(0,1)$.

$U \in (F(x-1), F(x)]$

$\Rightarrow$ return X

To simulate binomial, geometric, negative-binomial or Poisson rv's in R, use `rbinom`, `rgeom`, `rnbinom` or `rpois`.

For simulating other (finite) discrete rv's R provides `sample(x, size, replace = FALSE, prob = NULL)`.
The inputs are

`x`	A vector giving the possible values the rv can take;
`size`	How many rv's to simulate;
`replace`	Set this to `TRUE` to generate an iid sample, otherwise the rv's will be conditioned to be different from each other;
`prob`	A vector giving the probabilities of the values in `x`. If omitted then the values in `x` are assumed to be equally likely.

See the section 'Simulating other (finite) discrete rv using sample function' in sampling_example.pdf.

Simulating continuous random variables

Simulating continuous random variables

Suppose that we are given $U \sim U(0,1)$ and want to simulate a continuous rv X with cdf F_X.

X cdf $F(x)$

$\Rightarrow F_X^{-1}$

Put $Y = F_X^{-1}(U)$ then we have

$$F_Y(y) = \mathbb{P}(Y \le y) = \mathbb{P}(F_X^{-1}(U) \le y) = \mathbb{P}(U \le F_X(y)) = F_X(y).$$

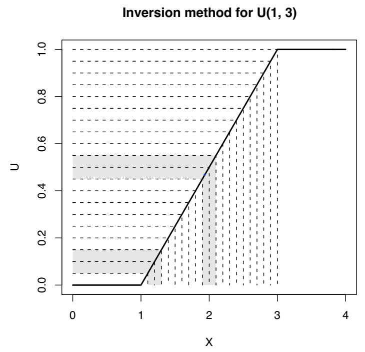
$U \sim U(0,1)$

$Y = F_X^{-1}(U).$ $\quad Y = X$

That is, Y has the same distribution as X.

Thus, if we can simulate a $U(0,1)$ rv, then we can simulate any continuous rv X for which we know F_X^{-1}. This is called the **inverse transformation method** or simply the **inversion method**.

Simulating $X \sim U(1,3)$



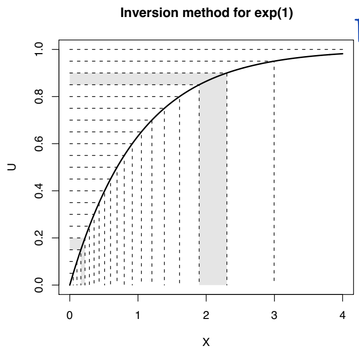
$F_x(x)$

If $X \sim U(1,3)$, $F_X(x) = (x-1)/2$ for $x \in (1,3)$.

Then, $F_X^{-1}(u) = 2u+1$ for $u \in (0,1)$.

See the section 'Simulating X $\sim$ U(1,3)' in sampling_example.pdf.

Simulating $X \sim \exp(\lambda)$



$F_X(x)$.

$U \sim U(0,1)$.

$X = F_X^{-1}(U)$

$1 - e^{-\lambda x} = U$

$e^{-\lambda x} = 1 - U$

$x = \frac{1}{-\lambda} \log(1-U)$.

If $X \sim \exp(\lambda)$, $F_X(x) = 1 - e^{-\lambda x}$ for $x \geq 0$.

Then, $F_X^{-1}(u) = -\frac{1}{\lambda} \log(1-u)$ for $u \in (0,1)$.

See the section 'Simulating X $\sim$ exp(1)' in sampling_example.pdf.

Random variable simulators in R

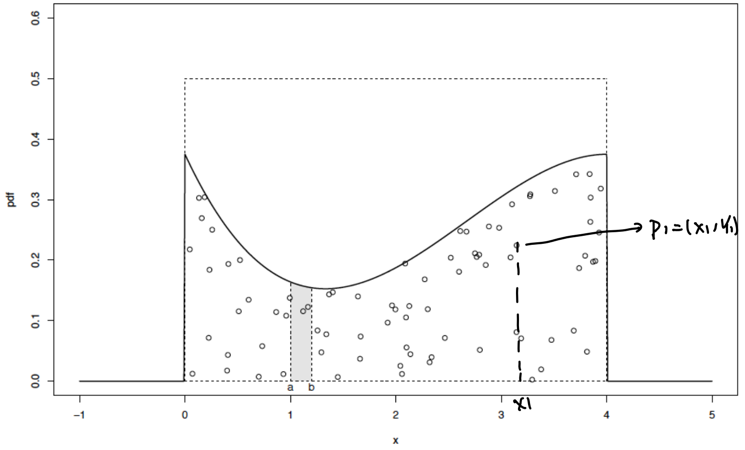
Distribution	R command
binomial	`rbinom`
Poisson	`rpoisson`
geometric	`rgeom`
negative binomial	`rnbinom`
uniform	`runif`
exponential	`rexp`
normal	`rnorm`
gamma	`rgamma`
beta	`rbeta`
student t	`rt`
F	`rf`
chi-squared	`rchisq`
Weibull	`rweibull`

The rejection method

The rejection method

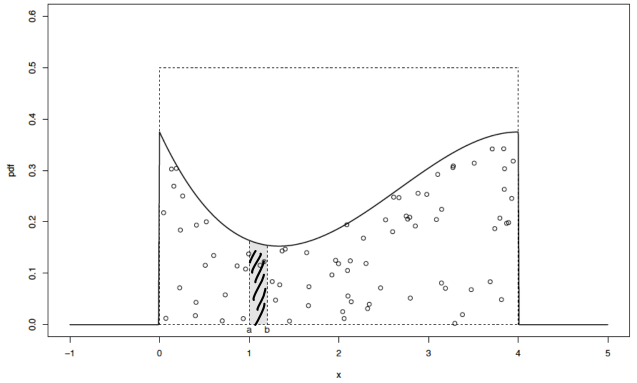
unknow F^{-1}

The inversion method works well if we can find F^{-1} analytically. If not, one method in this situation, which is often faster, is the rejection method.



distribution of x_1
is the distribution of X

We start with an example. Suppose that we have a continuous random variable X with pdf f_X concentrated on the interval $(0, 4)$. We imagine 'sprinkling' points $P_1, P_2, \ldots$, uniformly at random under the density function, and consider the distribution of X_1, the x coordinate of P_1.



Let R be the shaded region under f_X between a and b, then

$$\mathbb{P}(a < X_1 < b) = \mathbb{P}(P_1 \text{ hits R}) = \frac{\text{Area of R}}{\text{Area under density}}$$

$$= \frac{\int_a^b f_X(x)dx}{1} = \int_a^b f_X(x)dx.$$

$f_X(x)$ is density function

So X_1 has the same distribution as X.

X_1 pdf f_X

$P_1 = (X_1, Y_1)$

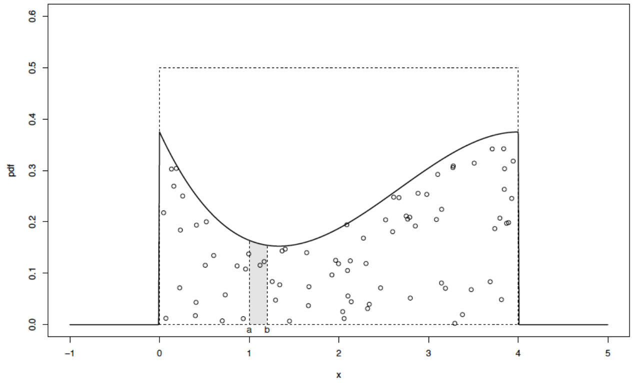
$P_2 = (X_2, Y_2)$

$P_3 = (X_3, Y_3)$

$\vdots$

$P_n = (X_n, Y_n)$

collect X_i.



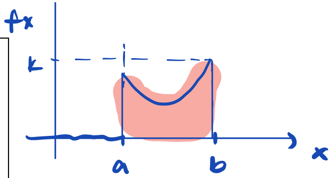
But how do we generate the points P_i uniformly under f_X?

The answer is to generate points at random in the rectangle $[0, 4] \times [0, 0.5]$, and then *reject* those that fall above the pdf.

Rejection method (uniform envelope)

Suppose that f_X is non-zero only on $[a, b]$, and $f_X \le k$.

1. Generate $X \sim U(a, b)$ and $Y \sim U(0, k)$ independent of X (so $P = (X, Y)$ is uniformly distributed over the rectangle $[a, b] \times [0, k]$).
2. If $Y < f_X(X)$ then return X, otherwise go back to step 1.



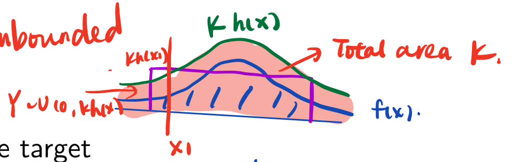
Example: consider the triangular pdf f_X defined as

$$f_X(x) = \begin{cases} x & \text{if } 0 < x < 1; \\ (2 - x) & \text{if } 1 \le x < 2; \\ 0 & \text{otherwise.} \end{cases}$$

We apply the rejection method as follows: see rejecttriangle.pdf

General rejection method

rejection method when f_x is unbounded



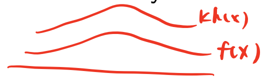
Our rejection method uses a rectangular envelope to cover the target density f_X. What to do if f_X is unbounded?

kh(x) should cover the area under f(x)

Let X have pdf h and let $Y \sim U(0, kh(X))$, then (X, Y) is uniformly distributed under the curve kh:

$$
\begin{aligned}
&\mathbb{P}((X, Y) \in (x, x + dx) \times (y, y + dy)) \\
&\quad = \boxed{\mathbb{P}(Y \in (y, y + dy) \mid X \in (x, x + dx))}\,\mathbb{P}(X \in (x, x + dx)) \\
&\quad = \frac{dy}{kh(x)} \underbrace{h(x)dx}_{P(X\in(x,x+dx))} = \frac{1}{k} dxdy.
\end{aligned}
$$

Suppose we wish to simulate from the density f_X. Let h be a density we can simulate from, and choose k such that →



$$k \geq k^* = \sup_x \frac{f_X(x)}{h(x)}.$$

kh(x) can cover f(x)

Then kh forms an envelope for f_X, and we can generate points uniformly within this envelope. By accepting points below the curve f_X, we get the general rejection method.

General rejection method

To simulate from the density f_X, we assume that we have envelope density h from which you can simulate, and that we have some $k < \infty$ such that $\sup_x f_X(x)/h(x) \leq k$.

1. Simulate X from h.
2. Generate $Y \sim U(0, kh(X))$.
3. If $Y < f_X(X)$ then return X, otherwise go back to step 1.

Another version of the general rejection method

Suppose we wish to simulate from the density f_X, but we know f_X up to constant (i.e., we know only $f_X^* \propto f_X$).

This situation often happens when we wish to simulate samples from the posterior distribution and it is hard to compute a normalisation factor for the posterior distribution. For the posterior $p(\theta|x) = \frac{f(x|\theta)p(\theta)}{f(x)}$, it is often hard to compute $f(x) = \int_\Theta f(x|\theta)p(\theta)d\theta$.

$f(x|\theta)p(\theta) \propto p(\theta|x)$.

f_x^*

Then the general rejection method is equivalent to:

Another version of the general rejection method

To simulate from the density f_X when we know only $f_X^* \propto f_X$, we assume that we have envelope density h from which you can simulate, and that we have some $k < \infty$ such that $\sup_x f_X^*(x)/h(x) \leq k$.

1. Simulate X from h.
2. Generate $Y \sim U(0, kh(X))$.
3. If $Y < f_X^*(X)$ then return X, otherwise go back to step 1.

Efficiency of the rejection method

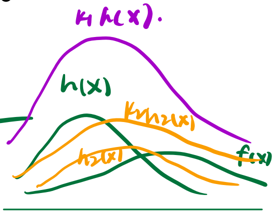
The efficiency of the rejection method is measured by the expected number of times you have to generate a candidate point (X, Y) to return (accept) one sample.

The area under the curve kh is k and the area under the curve f_X is 1, so the probability of accepting a candidate is $1/k$.

Thus the number of times N we have to generate a candidate point has distribution geom$(1/k)$, with mean

$$\mathbb{E}N = k.$$

So, the closer h is to f_X, the smaller we can choose k, and the more efficient the algorithm.



$k_2 < k_1$

$h_2(x)$ is more efficient

Example: gamma

For $m, \lambda > 0$, the $\Gamma(\lambda, m)$ density is

$$f(x) = \lambda^m x^{m-1} e^{-\lambda x} / \Gamma(m), \text{ for } x > 0.$$

There is no explicit formula for the cdf F or its inverse, so we will use the rejection method to simulate from f.

We will use an exponential envelope $h(x) = \mu e^{-\mu x}$, for $x > 0$. Using the inversion method we can easily simulate from h using $\underbrace{-\log(U)/\mu}_{\downarrow}$, where $U \sim U(0, 1)$.

sample from exp(x)

To envelop f we need to find

$$k^* = \sup_{x>0} \frac{f(x)}{h(x)} = \sup_{x>0} \frac{\lambda^m x^{m-1} e^{(\mu-\lambda)x}}{\mu\Gamma(m)}.$$

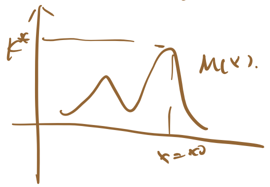
k^* will be infinite if $m < 1$ or $\lambda \leq \mu$ (check by yourself). For $m = 1$ the gamma is just an exponential. Thus we will assume $m > 1$ and choose $\mu < \lambda$.

we know how to sample with inverse method

For $m \in (0, 1)$ the rejection method can still be used, but a different envelope is required.

otherwise $k = \infty$

$k^* = \arg\max M(x)$



$M(x)$

$$k^* = \sup_{x>0} \boxed{\frac{f(x)}{h(x)}}$$

$\Rightarrow k^* h(x) \geq f(x)$ for $\forall x > 0$

To find k^* we take the derivative of the right-hand side above and set it to zero, to find the point where the maximum occurs. You can check that this is at the point $x = (m-1)/(\lambda - \mu)$, which gives

$$M(x) = \overset{C}{\frac{\lambda^m}{\mu\Gamma(m)}} x^{m-1} e^{(\mu-\lambda)x}$$

$$M'(x) = C\left[(m-1)x^{m-2}e^{(\mu-\lambda)x} + x^{m-1}e^{(\mu-\lambda)x}(\mu-\lambda)\right]$$

$$= C \cdot x^{m-2} e^{(\mu-\lambda)x}\left[(m-1) + (\mu-\lambda)x\right] = 0.$$

$$x = \frac{m-1}{\lambda-\mu}$$

$$k^* = \frac{\lambda^m (m-1)^{m-1} e^{-(m-1)}}{\mu(\lambda-\mu)^{m-1}\Gamma(m)}.$$

we need to set

① k ② μ

To improve efficiency we would like to choose our envelope to make k^* as small as possible. Looking at the formula for k^* this means choosing μ to make $\mu(\lambda-\mu)^{m-1}$ as large as possible. Setting the derivative with respect to μ to zero, we see that the maximum occurs when $\mu = \lambda/m$. Plugging this back in we get $k^* = m^m e^{-(m-1)}/\Gamma(m)$.

We can now code up our rejection algorithm: see gamma_sim.pdf

$$\frac{d}{d\mu}\left(\mu(\lambda-\mu)^{m-1}\right)$$

$$= (\lambda-\mu)^{m-1} + \mu(m-1)(\lambda-\mu)^{m-2}\cdot(-1)$$

$$= (\lambda-\mu)^{m-1}\left[1-\frac{\mu(m-1)}{\lambda-\mu}\right] = 0.$$

$$m\mu = \lambda$$

$$\mu = \frac{\lambda}{m}$$

The Gibbs sampler

by Guoqi Qian

Gibbs sampler

goal: simulate multivariate r.v.

$(X,Y,Z) \sim f(X,Y,Z)$ multivariate hard to simulate.

know how to sample from $X \sim f(X|Y,Z)$

$Y \sim f(Y|X,Z)$

$Z \sim f(Z|X,Y)$

The Gibbs sampler is a random vector generation method which does not require the complete information of the target multivariate probability distribution. Instead, it only requires the information of a set of the associated conditional distributions. Gibbs sampler will be computationally feasible if the conditional distributions are ready to be simulated.

Gibbs sampler

Suppose we want to generate a vector $\mathbf{u} = (u_1, \cdots, u_K)$ from the random variable $\mathbf{U} = (U_1, \cdots, U_K)$ which has a joint pdf $f(\mathbf{u})$.

Suppose the pdf $f(\mathbf{u})$ has a very complicated form. But for each $k = 1, \cdots, K$, the conditional pdf of U_k given $\mathbf{U}_{-k} = (U_1, \cdots, U_{k-1}, U_{k+1}, \cdots, U_K)$ is known and relatively easy to simulate. We use

$$f(u_k|\mathbf{u}_{-k}) \equiv f(u_k|u_1, \cdots, u_{k-1}, u_{k+1}, \cdots, u_K)$$

to denote such a conditional pdf.

Then the Gibbs sampler used to generate one observation of $\mathbf{U}$ can be described as follows.

Gibbs sampler

Algorithm : Gibbs sampler

$U^{(0)} = (U_1^{(0)}, U_2^{(0)}, \cdots U_K^{(0)})$

1° Arbitrarily generate/assign an initial vector $\mathbf{u}^{(0)} = (u_1^{(0)}, \cdots, u_K^{(0)})$ from the support of $f(\mathbf{u})$.

2° Generate a value $u_1^{(1)}$ from $f(u_1 | u_2 = u_2^{(0)}, \cdots, u_K = u_K^{(0)})$; then generate a value $u_2^{(1)}$ from $f(u_2 | u_1 = u_1^{(1)}, u_3 = u_3^{(0)}, \cdots, u_K = u_K^{(0)})$; continue until generate a value $u_{K-1}^{(1)}$ from $f(u_{K-1} | u_1 = u_1^{(1)}, \cdots, u_{K-2} = u_{K-2}^{(1)}, u_K = u_K^{(0)})$, and generate a value $u_K^{(1)}$ from $f(u_K | u_1 = u_1^{(1)}, \cdots, u_{K-1} = u_{K-1}^{(1)})$. The generated values are delivered as $\mathbf{u}^{(1)} = (u_1^{(1)}, \cdots, u_K^{(1)})$.

condition on most recent value.

Gibbs sampler

Algorithm : Gibbs sampler (continued)

3° For $j = 2, 3, \cdots$, do the following:
Generate a value $u_1^{(j)}$ from $f(u_1|u_2 = u_2^{(j-1)}, \cdots, u_K = u_K^{(j-1)})$; then generate a value $u_2^{(j)}$ from
$f(u_2|u_1 = u_1^{(j)}, u_3 = u_3^{(j-1)}, \cdots, u_K = u_K^{(j-1)})$;
continue until generate a value $u_{K-1}^{(j)}$ from
$f(u_{K-1}|u_1 = u_1^{(j)}, \cdots, u_{K-2} = u_{K-2}^{(j)}, u_K = u_K^{(j-1)})$,
and generate a value $u_K^{(j)}$ from $f(u_K|u_1 = u_1^{(j)}, \cdots, u_{K-1} = u_{K-1}^{(j)})$.
The generated values are delivered as $\mathbf{u}^{(j)} = (u_1^{(j)}, \cdots, u_K^{(j)})$.

Note the conditioning is always based on the latest values of $(u_1, \cdots, u_K)$.

Gibbs sampler : remarks

- Using theory of Markov chain, it can be shown that

$$\mathbf{u}^{(j)} \xrightarrow{d} f(\mathbf{u}) \quad \text{as } j \to \infty$$

under fairly general conditions. Details are not pursued here.

- This implies that $\mathbf{u}^{(j)}$ can be roughly regarded as an observation of $\mathbf{U}$ from pdf $f(\mathbf{u})$ when j is sufficiently large. Empirical methods are available to determine how large j should be (we will revisit this topic later).
- In practice, we usually generate a long sequence $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(m)}, \mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ using the Gibbs sampler; then ignore the **burn-in** sequence $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(m)}$ and use only the part $\mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ as a random sample from $f(\mathbf{u})$.
- $\mathbf{u}^{(m+1)}, \cdots, \mathbf{u}^{(m+J)}$ are (not independent) of each other but constitute a Markov chain.

Gibbs sampler : example

Eggs of certain species of insects hatch under appropriate conditions. Suppose there are N eggs of this species of insects in a particular area, and X eggs of them will hatch. Also let p be the probability that such an egg will hatch.

We model

$$X|p, N \sim \text{binomial}(N, p),$$

consider N and p as parameters, and put the following priors on the parameters based on a previous study on these insects:

$$N \sim \text{Poi}(16), \quad p \sim \text{Beta}(2, 4).$$

We wish to learn the property of the marginal distribution of X.

Gibbs sampler : example

$$X|p, N \sim \text{binomial}(N, p),$$

$$N \sim \text{Poi}(16), \quad p \sim \text{Beta}(2, 4).$$

We wish to learn the property of the marginal distribution of X.

1. Find the joint pdf of (X, p, N) and the marginal pdf of X. Do they have simple closed form?
2. Derive a Gibbs sampler for generating samples from the joint pdf of (X, p, N). Can we use the samples to learn the property of the marginal distribution of X?
3. Implement the algorithm in R.

Gibbs sampler : example

$f(x,p,N) = f(x|p,N)\cdot f(p,N)$
$= f(x|p,N)\cdot f(p)\cdot f(N)$

When $N \sim \text{Poi}(16)$ and $p \sim \text{Beta}(2,4)$, the joint pdf of (X, p, N) is

$$\begin{aligned} f(x,p,N) &= f(x|p,N)\cdot f(p)\cdot f(N) \\ &= \binom{N}{x} p^x(1-p)^{N-x}\cdot \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1}\cdot \frac{\lambda^N}{N!}e^{-\lambda} \\ &= 20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}; \\ & \quad x = 0,1,\cdots,N;\ 0\le p\le 1;\ N = 0,1,2,\cdots. \end{aligned}$$

The marginal pdf of X is

$$f(x) = \sum_{N=x}^{\infty}\int_0^1 f(x,p,N)dp = \sum_{N=x}^{\infty}\frac{20e^{-16}(x+1)(N-x+3)(N-x+2)(N-x+1)16^N}{(N+5)!}$$

which is difficult to be further simplified.

Gibbs sampler : example

$f(p|x,N) \propto f(p,x,N) \propto p^{x+1}(1-p)^{N-x+3}$

$\sim \text{Beta}(x+2, N-x+4)$

But it is not difficult to find the full conditional pdf's:

$$X|p,N \sim \text{bin}(N,p), \qquad p|x,N \sim \text{Beta}(x+2, N-x+4),$$
$$\text{and} \qquad N-x|x,p \sim \text{Poi}(16(1-p)).$$

Proof:

$$f(p|x,N) = \frac{f(x,p,N)}{f(x,N)} = \frac{20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}}{\int_0^1 20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}dp}$$
$$= \frac{p^{x+1}(1-p)^{N-x+3}}{\int_0^1 p^{x+1}(1-p)^{N-x+3}dp} = \frac{\Gamma(N+6)}{\Gamma(x+2)\Gamma(N-x+4)}p^{x+1}(1-p)^{N-x+3}$$

which is a pdf of Beta$(x+2, N-x+4)$.

Gibbs sampler : example

$f(N|x,p) \propto f(N,x,p)$

$\propto \underbrace{\binom{N}{x}}_{\frac{N!}{x!(N-x)!}} \frac{16^N}{N!} (1-p)^N \propto \frac{1}{(N-x)!}[16(1-p)]^N$

$\propto \frac{1}{(N-x)!}[16(1-p)]^{N-x} e^{-16(1-p)}$

$X \sim \text{Poisson}(\lambda)$

$\frac{\lambda^x e^{-\lambda}}{x!}$

$$
\begin{aligned}
f(N|x,p) &= \frac{f(x,p,N)}{f(x,p)} = \frac{20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}}{\sum_{N=x}^{\infty} 20e^{-16}\binom{N}{x}\frac{16^N}{N!}p^{x+1}(1-p)^{N-x+3}} \\
&= \frac{\binom{N}{x}[16(1-p)]^N\frac{1}{N!}}{\sum_{\tilde{N}=x}^{\infty}\binom{\tilde{N}}{x}\frac{16^{\tilde{N}}}{\tilde{N}!}(1-p)^{\tilde{N}}} = \frac{\frac{1}{(N-x)!}[16(1-p)]^N}{\sum_{\tilde{N}=x}^{\infty}\frac{1}{(\tilde{N}-x)!}[16(1-p)]^{\tilde{N}}} \\
&= \frac{[16(1-p)]^{N-x}}{(N-x)!}e^{-16(1-p)}; \quad N = x, x+1, x+2, \cdots .
\end{aligned}
$$

This implies that $(N-x|x,p) \overset{d}{=}$ Poisson$(16(1-p))$. □

Now a sample of (X,p,N) can be generated using the following Gibbs sampler algorithm.

$N \Rightarrow \underline{N-x} + \underline{x}.$

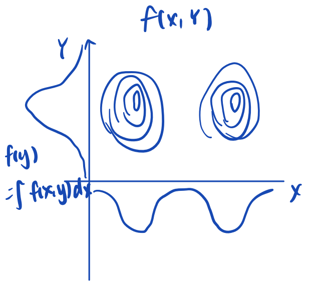
Gibbs sampler : example

Algorithm : Gibbs sampler for (X, p, N)

1° Arbitrarily choose an initial vector, e.g. $(x^{(0)}, p^{(0)}, N^{(0)}) = (8, 0.5, 16)$.

2° $(x^{(1)}, p^{(1)}, N^{(1)})$ is obtained by:
generating $x^{(1)}$ from $\text{Bin}(N^{(0)}, p^{(0)})$;
generating $p^{(1)}$ from $\text{Beta}(x^{(1)} + 2, N^{(0)} - x^{(1)} + 4)$; and
generating an $N^{(1)}$ from $\text{Poi}(16(1 - p^{(1)})) + x^{(1)}$. $\underline{N^{(1)} - x^{(1)}} + x^{(1)}$

3° $(x^{(j)}, p^{(j)}, N^{(j)})$, $j = 2, 3, \cdots$, is obtained by:
generating $x^{(j)}$ from $\text{Bin}(N^{(j-1)}, p^{(j-1)})$;
generating $p^{(j)}$ from $\text{Beta}(x^{(j)} + 2, N^{(j-1)} - x^{(j)} + 4)$; and
generating an $N^{(j)}$ from $\text{Poi}(16(1 - p^{(j)})) + x^{(j)}$.



$f(x)=\int f(x,y)\,dy$

Gibbs sampler : example

burn-in

$\cancel{X^{(1)}, p^{(1)}, N^{(1)}}$

$\cancel{X^{(2)}, p^{(2)}, N^{(2)}}$

$\vdots$

$\boxed{X^{(j)}, p^{(j)}, N^{(j)}} \longrightarrow \begin{array}{l} j\to\infty \\ f(x,p,N) \end{array}$

Once samples of (X, p, N) are obtained from the Gibbs sampler, we can obtain samples simulated from the marginal distribution of X by just collecting samples of X from the samples of (X, p, N).

The Gibbs sampler algorithm was implemented in R. Then, 1000 (X, p, N) samples are generated, and the simulated marginal pdf's of X, p and N are obtained. See simulation_Gibbs.pdf.

Learning goals

Understand

- Algorithm to simulate discrete random variables
- Inversion method (algorithm to simulate continuous random variables)
- Rejection methods
- Gibbs sampler

Specifically,

- Understand key ideas for each algorithm
- Know when each algorithm can be used
- Be able to derive and implement each algorithm