Continuous Random Variables II

COSC/DATA 405/505

More on Continuous Random Variable Simulation

More detail on the Exponential Distribution

Another Approach to Simulating Poisson Processes

Simulating Conditional Distributions

Monte Carlo Integration

Rejection Sampling - a way to simulate more complicated distributions

Exponential Random Variables

Exponential random variables are used as simple models for such things as failure times of mechanical or electronic components, or for the time it takes a server to complete service to a customer. The exponential distribution is characterized by a constant *failure rate*, denoted by λ.

T has an exponential distribution with rate $\lambda > 0$ if

$$\mathbf{P}(T \leq t) = 1 - e^{-\lambda t}$$

for any non-negative t.

Exponential Probabilities

The pexp() function can be used to evaluate the distribution function.

```
pexp(q, rate)
```

The output from this is the value of $P(T \leq q)$, where T is an exponential random variable with parameter `rate`.

Example

Suppose the service time at a bank teller can be modeled as an exponential random variable with rate 3 per minute.

Then the probability of a customer being served in less than 1 minute is

```
pexp(1, rate = 3)

## [1] 0.9502129
```

Thus, $P(X \leq 1) = 0.95$, when X is an exponential random variable with rate 3.

Exponential Density Function and Moments

Differentiating the right hand side of the distribution function with respect to t gives the exponential probability density function:

$$f(t) = \lambda e^{-\lambda t}.$$

The dexp() function can be used to evaluate this. It takes the same arguments as the pexp() function. The qexp() function can be used to obtain quantiles of the exponential distribution.

The expected value of an exponential random variable is $1/\lambda$, and the variance is $1/\lambda^2$.

Exponential Pseudorandom Numbers

A simple way to simulate exponential pseudorandom variates is based on the *inversion* method.

For an exponential random variable $F(x) = 1 - e^{-\lambda x}$**, so** $F^{-1}(U) = -\frac{\log(1-U)}{\lambda}$**.**

Therefore, for any $x \in (0, 1)$**, we have**

$$P(F(T) \leq x) = P(T \leq F^{-1}(x)) = F(F^{-1}(x)) = x.$$

Thus, $F(T)$ **is a uniform random variable on the interval** $(0, 1)$**.**

Algorithm to Compute Exponential Pseudorandom Numbers

Generate a uniform pseudorandom variable U on [0,1], and set

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

Solving this for T, we have

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

T has an exponential distribution with rate λ.

Exponential Pseudorandom Numbers

The R function rexp() can be used to generate n random exponential variates.

rexp(n, rate)

Example

A bank has a single teller who is facing a lineup of 10 customers. The time for each customer to be served is exponentially distributed with rate 3 per minute. We can simulate the service times (in minutes) for the 10 customers.

```
servicetimes <- rexp(10, rate = 3)
servicetimes
```

Example

```
##  [1] 0.1884296 0.1467667 0.4509145 0.1104748
##  [5] 0.3304051 0.3523051 0.9007785 0.3800181
##  [9] 0.5902491 0.2487142
```

The total time until these 10 simulated customers will complete service is around 4 minutes.

```
sum(servicetimes)
```

```
## [1] 3.699056
```

Another way to simulate a Poisson process

It can be shown that the points of a homogeneous Poisson process with rate λ on the line are separated by independent exponentially distributed random variables which have mean $1/\lambda$.

This leads to another simple way of simulating a Poisson process on the line.

Example

Simulate the first 25 points of a Poisson 1.5 process, starting from 0.

```
X <- rexp(25, rate = 1.5)
cumsum(X)
```

```
##  [1]  1.243522  1.524950  1.656004  2.917379
##  [5]  3.184172  3.193563  3.733177  4.193292
##  [9]  5.154919  5.915369  8.102377  8.797579
## [13]  8.970467  8.977002  9.248910 11.745159
## [17] 11.810911 12.052011 12.232306 14.860722
## [21] 15.766072 15.936573 17.216926 18.253731
## [25] 19.477939
```

Simulating Numbers that Follow Conditional Distributions

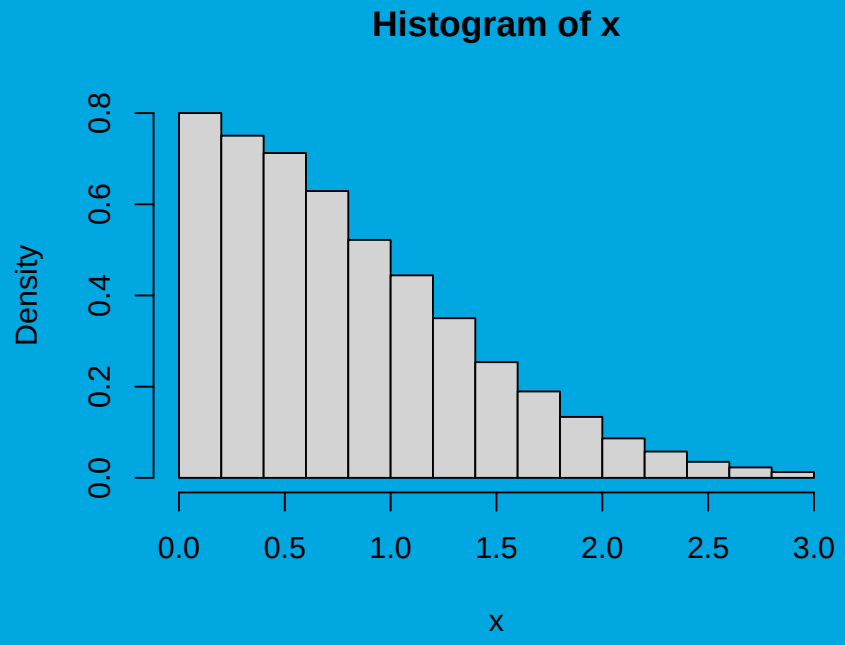
We can simulate random numbers from certain conditional distributions by first simulating according to an unconditional distribution, and then rejecting those numbers which do not satisfy the specified condition.

Example

Simulate x from the standard normal distribution, conditional on the event that $0 < x < 3$. We will simulate from the entire normal distribution and then accept only those values which lie between 0 and 3.

We can simulate a large number of such variates as follows:

```
x <- rnorm(100000)        # simulate from the standard normal
x <- x[(0 < x) & (x < 3)]   # reject all x's outside (0,3)
hist(x, probability=TRUE)   # show the simulated values
```



This plot shows how the histogram tracks the rescaled normal density over the interval $(0, 3)$.

Monte Carlo Integration

Suppose $g(x)$ is any function that is integrable on the interval $[a, b]$.

The integral

$$\int_a^b g(x)dx$$

gives the area of the region with $a < x < b$ and y between 0 and $g(x)$ (where negative values count towards negative areas).

Monte Carlo integration uses simulation to obtain approximations to these integrals. It relies on the law of large numbers.

This law says that a sample mean from a large random sample will tend to be close to the expected value of the distribution being sampled.

If we can express an integral as an expected value, we can approximate it by a sample mean.

Monte Carlo Integration

For example, let $U_1, U_2, \ldots, U_n$ be independent uniform random variables on the interval $[a, b]$. These have density $f(u) = 1/(b-a)$ on that interval. Then

$$E[g(U_i)] = \int_a^b g(u)\frac{1}{b-a}du$$

so the original integral $\int_a^b g(x)dx$ can be approximated by $b-a$ times a sample mean of $g(U_i)$.

Example

To approximate the integral $\int_0^1 x^4 dx$, use the following lines:

```
u <- runif(100000)
mean(u^4)

## [1] 0.1986411
```

Compare with the exact answer, 0.2, which can easily be computed.

Example

To approximate the integral $\int_2^5 \sin(x)dx$, use the following lines:

```
u <- runif(100000, min = 2, max = 5)
mean(sin(u))*(5-2)


## [1] -0.7091414
```

The true value can be shown to be -0.700.

Multiple integration

Now let $V_1, V_2, \ldots, V_n$ be an additional set of independent uniform random variables on the interval $[0, 1]$, and suppose $g(x, y)$ is now an integrable function of the two variables x and y. The law of large numbers says that

$$\lim_{n\to\infty} \sum_{i=1}^{n} g(U_i, V_i)/n = \int_0^1 \int_0^1 g(x, y)dxdy$$

with probability 1.

So we can approximate the integral $\int_0^1 \int_0^1 g(x, y)dxdy$ by generating two sets of independent uniform pseudorandom variates, computing $g(U_i, V_i)$ for each one, and taking the average.

Example

Approximate the integral $\int_3^{10}\int_1^7 \sin(x-y)dxdy$ using the following:

```
U <- runif(100000, min = 1, max = 7)
V <- runif(100000, min = 3, max = 10)
mean(sin(U - V))*42

## [1] 0.1331901
```

The factor of $42 = (7-1)(10-3)$ compensates for the joint density of U and V being $f(u,v) = 1/42$.

Using non-uniform pseudorandom numbers

The uniform density is by no means the only density that can be used in Monte Carlo integration.

If the density of X is $f(x)$, then

$$E[g(X)/f(X)] = \int [g(x)/f(x)]f(x)dx = \int g(x)dx$$

so we can approximate the latter by sample averages of $g(X)/f(X)$.

Example

To approximate the integral $\int_1^\infty \exp(-x^2)dx$, write it as

$$\int_0^\infty \exp[-(x+1)^2]dx,$$

and use an exponential distribution for X:

```
X <- rexp(100000)
mean( exp( -(X + 1)^2 ) / dexp(X) )

## [1] 0.1388715
```

The true value of this integral is 0.1394.

Caution

Monte Carlo integration is not always successful: sometimes the ratio $g(X)/f(X)$ varies so much that the sample mean doesn't converge.

Try to choose $f(x)$ so this ratio is roughly constant, and avoid situations where $g(x)/f(x)$ can be arbitrarily large.

Advanced Simulation Methods

The simulation methods discussed so far will only work for particular types of probability densities or distributions.

General purpose simulation methods can be used to draw pseudorandom samples from a wide variety of distributions.

Rejection sampling

The idea of rejection sampling was used earlier to sample from a conditional distribution: sample from a convenient distribution, and select a subsample to achieve the target distribution.

We will show how to use rejection sampling to draw a random sample from a univariate density or probability function $g(x)$, using a sequence of two examples.

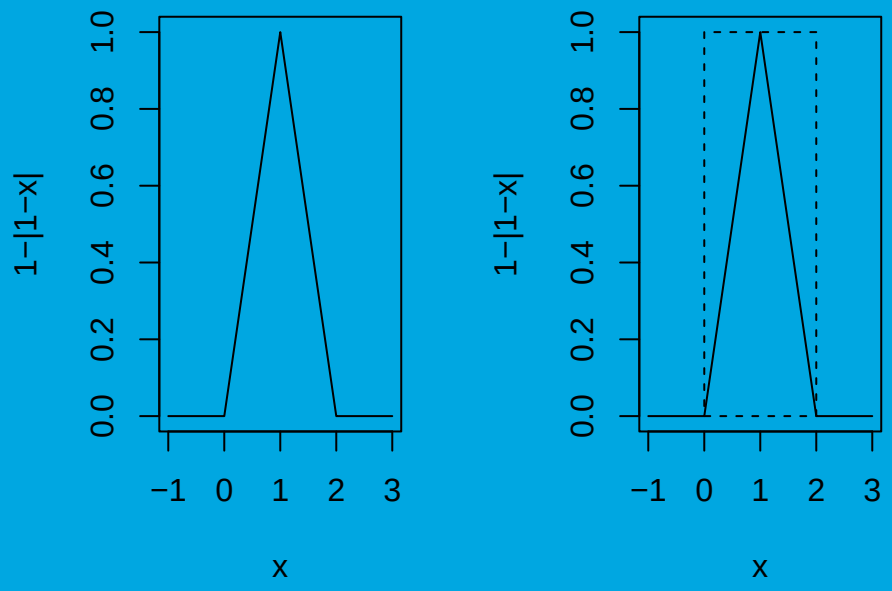
Example

Simulate pseudorandom variates from the triangular density function

$$g(x) = \begin{cases} 1 - |1 - x|, & 0 \leq x < 2 \\ 0, & \textbf{otherwise} \end{cases}$$

Example

```
par(mfrow=c(1,2))
curve((1-abs(1-x))*(x<2)*(x>0), from = -1, to = 3, ylab="1-|1-x|")
curve((1-abs(1-x))*(x<2)*(x>0), from = -1, to = 3, ylab="1-|1-x|")
lines(c(0,2,2,0,0),c(0,0,1,1,0), lty=2)
```



The graph of the triangular density function on $(0, 2)$, together with a dashed rectangle in the right hand panel.

Rejection Sampling Example

If we could draw points uniformly from the triangular region below the density, the x-coordinate would be distributed with density $g(x)$.

The right hand panel of the figure shows that the graph where the density is nonzero can be entirely contained in a rectangle of height 1 and width 2.

A subset of uniformly distributed points in the rectangle will be uniformly distributed in the triangular area under the triangular density.

Simulating from the Triangular Density

Thus, a strategy for simulating values from the triangular density is:

1. **Simulate a point (U_1, U_2) uniformly in the rectangle.**

2. **If (U_1, U_2) is located within the triangular region, accept U_1 as a pseudorandom variate; otherwise, reject it, and return to step 1.**

Simulating from the Triangular Density

Since the triangular density occupies half of the area of the rectangle, we would expect to sample roughly 2 uniform points from the rectangle for every point we accept from the triangular distribution.

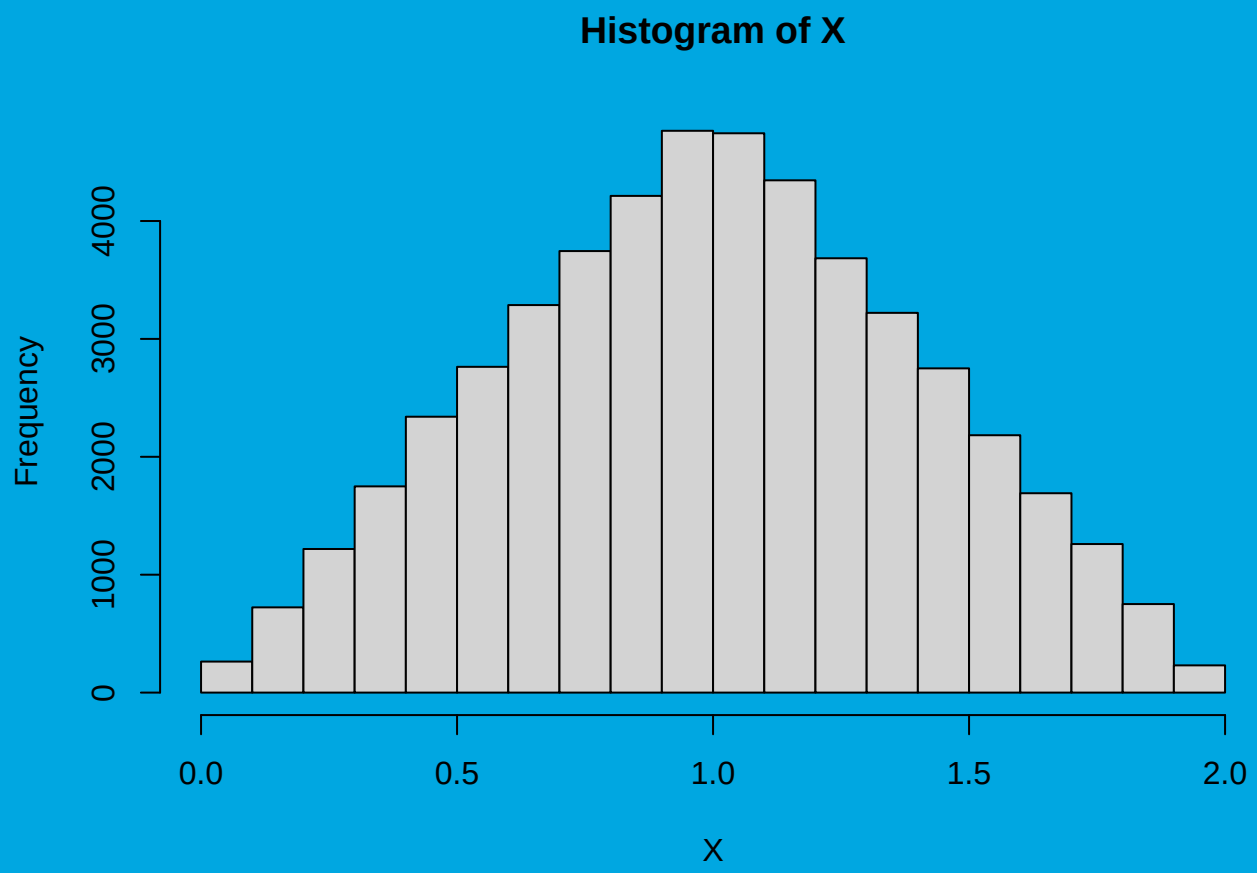
In vectorized form, the steps are:

```
U1 <- runif(100000, max=2)
U2 <- runif(100000)
X <- U1[U2 < (1 - abs(1 - U1))]
```

The vector `X` will contain approximately 50000 simulated values from the triangular distribution.

How did we do?

```
hist(X)
```



... looks okay, but ...

How did we do?

It is easier to check if we include an overlaid density curve:

```
hist(X, freq=FALSE)   # total area = 1
curve(1-abs(1-x), -.5, 2.5, add=TRUE)
```