Statistical Computing (36-350)
Lecture 15: Simulation I: Generating Random Variables

Cosma Shalizi

22 October 2012
Agenda

- The basic random-variable commands
- Transforming uniform random variables to other distributions
  - The quantile method
  - The rejection method
- Where the uniform random numbers come from

**REQUIRED READING:** Matloff, chapter 8
*R Cookbook*, chapter 8

**OPTIONAL READING:** Chambers, section 6.10
Stochastic Simulation

Why simulate?
Stochastic Simulation

Why simulate?
- We want to see what a probability model actually does
Why simulate?

- We want to see what a probability model actually does
- We want to understand how our procedure works on a test case
Stochastic Simulation

Why simulate?

- We want to see what a probability model actually does
- We want to understand how our procedure works on a test case
- We want to use a partly-random procedure
Why simulate?

- We want to see what a probability model actually does
- We want to understand how our procedure works on a test case
- We want to use a partly-random procedure

All of these require drawing random variables from distributions
Built-in Random Variable Generators

runif, rnorm, rbinom, rpois, rexp, etc. etc.
First argument is always n, number of variables to generate
Subsequent arguments are parameters to distribution, and vary with the distribution
Many Distributions at Once

Parameters are recycled:

```r
> rnorm(n=4,mean=c(-1000,1000),sd=1)
[1]  -999.3637  1000.4710  -1000.4449  1000.1040
```

Each of the \( n \) draws can get its own parameters
sample(x, size, replace=FALSE, prob=NULL)

draw random sample of size points from x, optionally with replacement and/or weights
x can be anything where length() makes sense, basically
sample

```
sample(x, size, replace=FALSE, prob=NULL)
```

draw random sample of size points from x, optionally with replacement and/or weights
x can be anything where `length()` makes sense, basically `sample(x)` does a random permutation

```
sample(cats$Sex) # Randomly shuffle sexes among cats
```
sample

sample(x, size, replace=FALSE, prob=NULL)

draw random sample of size points from x, optionally with replacement and/or weights
x can be anything where length() makes sense, basically sample(x) does a random permutation

sample(cats$Sex) # Randomly shuffle sexes among cats

If x is a single number, treat it like 1:x

> sample(5)
[1] 1 4 3 2 5
Biased Coins

Given: uniform random variable $U$, success probability $p$
Wanted: A Bernoulli($p$) random variable
Biased Coins

Given: uniform random variable $U$, success probability $p$
Wanted: A Bernoulli($p$) random variable
Return 1 if $U \leq p$, else return 0

\texttt{ifelse(runif(n) =< p, 1, 0)}
or just \texttt{rbinom(n, size=1, prob=p)}
Given: uniform random variable \( U \), success probability \( p \)

Wanted: A Bernoulli\((p)\) random variable

Return 1 if \( U \leq p \), else return 0

\[
\text{ifelse(runif(n) } \leq p, 1, 0)
\]

or just

\[
\text{rbinom(n, size=1, prob=p)}
\]
Given: uniform $U$, category probabilities $p_1, p_2, \ldots, p_k$
Wanted: a categorical random variable with that p.m.f.
Given: uniform $U$, category probabilities $p_1, p_2, \ldots, p_k$
Wanted: a categorical random variable with that p.m.f.
If $U \leq p_1$, return 1
else if $U \leq p_1 + p_2$, return 2
etc.

```r
rmultinoulli <- function(n, prob) {
  return(sample(1:length(prob), replace=TRUE, size=n, prob=prob))
}
```

(``rmultinom`` gives counts, not a sequence)
Given: uniform $U$, category probabilities $p_1, p_2, \ldots, p_k$
Wanted: a categorical random variable with that p.m.f.
If $U \leq p_1$, return 1
else if $U \leq p_1 + p_2$, return 2
etc.

\[
\text{min(which}(u < \text{cumsum}(p)))
\]

(needs some thought to vectorize)
Categorical or Discrete Variables

Given: uniform $U$, category probabilities $p_1, p_2, \ldots, p_k$
Wanted: a categorical random variable with that p.m.f.
If $U \leq p_1$, return 1
else if $U \leq p_1 + p_2$, return 2
etc.

$$\min(\text{which}(u < \text{cumsum}(p)))$$

(needs some thought to vectorize)

```
rmultinoulli <- function(n, prob) {
    return(sample(1:length(prob), replace=TRUE, size=n, prob=prob))
}
```

(rmultinom gives counts, not a sequence)
The Quantile Transform Method

Given: uniform random variable $U$, CDF $F$
Claim: $X = F^{-1}(U)$ is a random variable with CDF $F$
The Quantile Transform Method

Given: uniform random variable $U$, CDF $F$
Claim: $X = F^{-1}(U)$ is a random variable with CDF $F$
Proof:

$$
\mathbb{P}(X \leq a) = \mathbb{P}(F^{-1}(U) \leq a) = \mathbb{P}(U \leq F(a)) = F(a)
$$

$F^{-1}$ is the quantile function
∴ if we can generate uniforms and we can calculate quantiles, we can generate non-uniforms
To turn $U$ into a coin-toss with bias $p$: is $U \leq p$ or not?
Less Mathematically

To turn $U$ into a coin-toss with bias $p$: is $U \leq p$ or not?
To turn $U$ into a binomial: start with $X = 0$; if $U \leq F(X)$, stop, otherwise add 1 to $X$ and check again.
To turn $U$ into a coin-toss with bias $p$: is $U \leq p$ or not?
To turn $U$ into a binomial: start with $X = 0$; if $U \leq F(X)$, stop, otherwise add 1 to $X$ and check again
Tedious do this iteratively
No next value for continuous random variables
Less Mathematically

To turn $U$ into a coin-toss with bias $p$: is $U \leq p$ or not?

To turn $U$ into a binomial: start with $X = 0$; if $U \leq F(X)$, stop, otherwise add 1 to $X$ and check again.

Tedious do this iteratively.

No next value for continuous random variables.

Quantiles solve both difficulties.
Quantile functions often don’t have closed form, and don’t have nice numerical solutions
But we know the probability density function — can we use that?
Suppose the pdf $f$ is zero outside an interval $[c, d]$, and $\leq M$ on the interval
Draw the rectangle $[c, d] \times [0, M]$, and the curve $f$
Area under the curve $= 1$
Area under curve and $x \leq a$ is $F(a)$
How can we uniformly sample area under the curve?
Base R commands
Transforming Uniform Random Numbers
Where Do the Uniforms Come From?

Categorical Random Variables
Quantile Transform
Rejection Method

\[ M \leftarrow 3.3; \text{curve}(\text{dbeta}(x, 5, 10), \text{from}=0, \text{to}=1, \text{ylim}=c(0, M)) \]
We sample uniformly from the box, and take the points under the curve
We sample uniformly from the box, and take the points under the curve

\[ R \sim \text{Unif}(c, d) \]
\[ U \sim \text{Unif}(0, 1) \]

If \( MU \leq f(R) \) then \( X = R \), otherwise try again
```
r <- runif(300, min=0, max=1); u <- runif(300, min=0, max=1)
below <- which(M*u <= dbeta(r, 5, 10))
points(r[below], M*u[below], pch="+"); points(r[-below], M*u[-below], pch="-")
```
Base R commands
Transforming Uniform Random Numbers
Where Do the Uniforms Come From?

Categorical Random Variables
Quantile Transform
Rejection Method

```r
hist(r[below], xlim=c(0,1), probability=TRUE); curve(dbeta(x,5,10), add=TRUE)
points(r[below], M*u[below], pch="+"); points(r[-below], M*u[-below], pch="-")
```
If $f$ doesn’t go to zero outside $[c,d]$, try to find another density $\rho$ where

- $\rho$ also has unlimited support
- $f(a) \leq M\rho(a)$ everywhere
- we can generate from $\rho$ (say by quantiles)

Then $R \sim \rho$, and accept when $MU\rho(R) \leq f(R)$ (Uniformly distributed on the area under $\rho$)
Need to make multiple “proposals” $R$ for each $X$
eq generated 300 for figure, only accepted 78
Important for efficiency to keep this ratio small
Ideally: keep the proposal distribution close to the target
Uniform numbers come from finite algorithms, so really only pseudo-random

We want:

- Number of $U_i$ in $[a, b] \subseteq [0, 1]$ is $\propto (b - a)$
- No correlation between successive $U_i$
- No detectable dependences in larger or longer groupings

Modern pseudo-random generators are now very good at all three

Too involved to go into here, but will show a simpler cousin
plot(r, main="100 draws from runif")
plot(hist(r), freq=FALSE, main="Histogram of 100 draws from runif")
plot(r[-100], r[-1], xlab="r[i]", ylab="r[i+1]",
     main="Scatterplot of successive draws from runif")
Rotations

Take

\[ U_{i+1} = U_i + \alpha \mod 1 \]

If \( \alpha \) is irrational, this never repeats and is uniformly distributed.
If \( \alpha \) is rational but the denominator is very large, the period is very long, and it is uniform on those points.
More Complicated Dynamics

Arnold Cat Map:

\[ U_{t+1} = U_t + \phi_t \text{ mod } 1 \]
\[ \phi_{t+1} = U_t + 2\phi_t \text{ mod } 1 \]

If we report only \( U_t \), the result is uniformly distributed and hard to predict.
Wikipedia, s.v. “Arnold’s cat map”
successive values from \texttt{runif} vs. Arnold cat map
Similar ideas are built into the random-number generator in R (with more internal dimensions)
Generally: Long periods, rapid divergence of near-by points (unstable), uniform distribution, low correlation
Using the default generator is a very good idea, unless you really know what you are doing
Setting the Seed

The sequence of pseudo-random numbers depends on the initial condition, or **seed**

Stored in `.Random.seed`, a global variable

To reproduce results exactly, set the seed

```r
> old.seed <- .Random.seed # Store the seed
> set.seed(20010805) # Set it to the day I adopted my cat
> runif(2)
[1] 0.1378908 0.7739319
> set.seed(20010805) # Reset it
> runif(2)
[1] 0.1378908 0.7739319
> .Random.seed <- old.seed # Restore old seed
```

See Chambers, §6.10, for some subtleties about working with external programs
Summary

- Unstable dynamical systems give us something very like uniform random numbers
- We can transform these into other distributions when we can compute the distribution function
  - The quantile method when we can invert the CDF
  - The rejection method if all we have is the pdf
- The basic R commands encapsulate a lot of this for us