Simulation

MX

2025-12-28

Chap 3 Methods for Generating Random Variables

3.1 Intro

Methods for generating random variates from other probability distributions all depend on the uniform random number generator.

In this text we assume that a suitable uniform pseudo-random number generator is available.

The uniform pseudo-random number generator in R is runif (Random UNIForm).

```
n = 3
runif(n)  # lb = 0, ub = 1, by default
```

```
## [1] 0.9194429 0.5337425 0.1945558
```

```
runif(n, 0, 3)
```

```
## [1] 1.4691957 0.9157732 0.7496369
```

```
matrix(runif(2*3), nrow=2, ncol=3)
```

```
##           [,1]      [,2]       [,3]
## [1,] 0.3124036 0.7499898 0.77900643
## [2,] 0.2371743 0.9722245 0.09185777
```

Generators for many of common distributions are available in R: rbeta, rgeom, rchisq, etc. (r for random, d for density or pmf, p for cdf, q for quantile. So pchisq, etc.)

Example 3.1. Sampling from a finite population using the `sample` function.

```
# toss coins
sample(0:1, size = 10, replace = T)
```

```
##  [1] 1 0 1 0 1 1 1 1 1 0
```

```
# lottery numbers
sample(1:100, size = 6, replace = F)
```

```
## [1] 61 17 71 53 96 69
```

```
# permutation of letters a-z
sample(letters, size = 10) # built-in constants letters & LETTERS
```

```
##  [1] "g" "z" "q" "p" "v" "h" "i" "e" "m" "k"
```

```r
# sample from a multinomial distr, use "prob" param
x = sample(1:3, size = 20, replace = T, prob = c(0.1, 0.2, 0.7))
table(x)
```

```
## x
##  1  2  3
##  1  5 14
```

3.2 The Inverse Transform Method

Thr 3.1 Probability Integral Transformation. Consider continuous r.v. X ~ F(x) and define the inverse transformation:

$$F^{-1}(u) = inf\{x : F(x) \geq u\}$$

Given the probability level u, find the least possible x of all x's for which CDF value $\geq$ u. Can be interpreted as the quantile function, for example $F^{-1}(0.25)$ means the first quartile, $F^{-1}(0.5)$ the median.

If U ~ Uniform(0,1) then for all x,

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

This means r.v. $F^{-1}(U)$ has the same distr as X. Thus to generate a random obs X, generate an obs U and deliver the inverse value $F^{-1}(u)$.

3.2.1 The Inverse Transform Method, continuous case

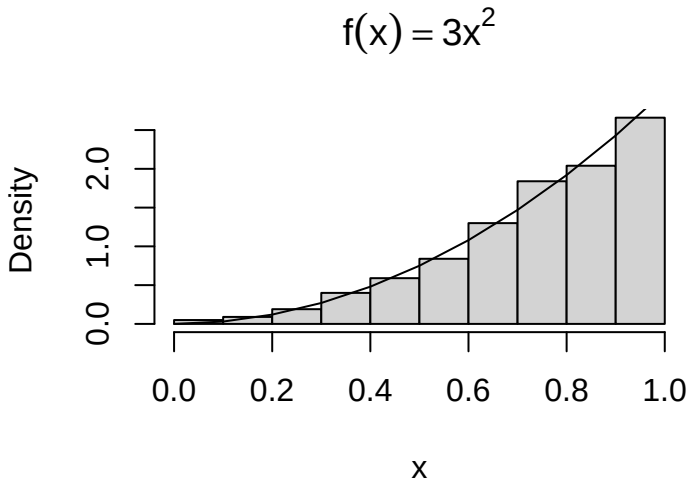
Example 3.2. Simulate a random sample from the distr with density, CDF, and inverse CDF

$$f(x) = 3x^2, 0 < x < 1$$

$$F(x) = x^3$$

$$F^{-1}(u) = u^{1/3}$$

```r
n = 1000
u = runif(n)
x = u^(1/3)  # random sample simulated
hist(x, prob = T,
     main = expression(f(x)==3*x^2))  # density histogram of the sample
y = seq(0,1,0.1)
lines(y, 3*y^2)  # density curve f(x)
```



3.2.2 The Inverse Transform Method, discrete case

Pages 65 - 69

3.3 the Acceptance-Rejection Method

Suppose X and Y are rvs with density or pmf f and g, respectively, and there exists a constant c such that $\frac{f(t)}{g(t)} \leq c$ for all t such that $f(t) > 0$. Then the acceptance-rejection method can be applied to generate X:

1. Find a r.v Y with a density g, satisfying
$$\frac{f(t)}{g(t)} \leq c$$
for all t such that $f(t) > 0$. Y should be easy to simulate and c small (see proof below).
2. Generate y from g. And generate u from Uniform(0,1).
3. If
$$u < \frac{f(y)}{c \cdot g(y)}$$
accept y and deliver $x = y$; otherwise reject y and repeat from step 2.

The probability of accepting given Y in Step 3 is

$$P(accept|Y) = P(U < \frac{f(Y)}{cg(Y)}|Y) = F_U(\frac{f(Y)}{cg(Y)}) = \frac{f(Y)}{cg(Y)}$$

The total probability of acceptance for any iteration is (comment: the following appears to be the discrete case)

$$P(accept) = \sum_y P(accept|Y = y) \cdot P(Y = y) = \sum_y \frac{f(y)}{cg(y)} \cdot P(Y = y) = \sum_y \frac{f(Y)}{cg(Y)} \cdot g(y) = 1/c$$

Thus each sample value of X requires c iterations. For efficiency, Y should be easy to simulate and c small.

To see that the accepted sample has the same distr as X, apply Bayes' Theorem. In the discrete case for each k such that $f(k) > 0$:

$$P(X = k|accepted) = P(Y = k|accepted) = \frac{P(accepted|Y = k) \cdot P(Y = k)}{P(accepted)} = \frac{\frac{f(k)}{cg(k)} \cdot g(k)}{1/c} = f(k)$$

The first equality is because if accepted, deliver $x = y$. The means for those x accepted, the probability matches the pmf.

Example. Suppose we want to simulate $Beta(\alpha = 2, \beta = 2)$ whose density is $f(x) = 6x(1 - x), 0 < x < 1$. Choose Y $\sim$ Uniform(0,1) with $g(y) = 1$, and so $\frac{f(t)}{g(t)} = 6t(1 - t) \leq 6 \cdot 0.5 \cdot 0.5 = 1.5$. And so a constant $c = 1.5$ exists. If generated y and u satisfy $u < \frac{f(y)}{cg(y)} = 4y(1 - y)$, then y is accepted and let $x = y$.

```
n = 1000  # generate n beta variables
x = numeric(n)

k = 0  # counter for accepted
j = 0  # counter for iterations

while (k<n){
  u = runif(1)
  y = runif(1)
  j = j + 1

  if (4*y*(1-y)>u){
```

```
    # accept y and deliver y=x
    x[k] = y
    k = k + 1
  }
}

cat(j, 'iterations,', n, 'accepted variables')
```

```
## 1481 iterations, 1000 accepted variables
```

Compare empirical and theoretical percentiles:

```
p = seq(0.1, 0.9, 0.1)
qhat = quantile(x, p)  # quantiles of sample
q = qbeta(p, shape1=2, shape2=2)  # theoretical quantiles
se = sqrt(p*(1-p) / n*dbeta(q, 2, 2))  # see chap 2

round(rbind(qhat, q, se), 3)
```

```
##        10%   20%   30%   40%   50%   60%   70%   80%   90%
## qhat 0.187 0.295 0.351 0.417 0.492 0.559 0.631 0.713 0.800
## q    0.196 0.287 0.363 0.433 0.500 0.567 0.637 0.713 0.804
## se   0.009 0.014 0.017 0.019 0.019 0.019 0.017 0.014 0.009
```

Repeating the simulation with a greater n will produce more precise estimates.

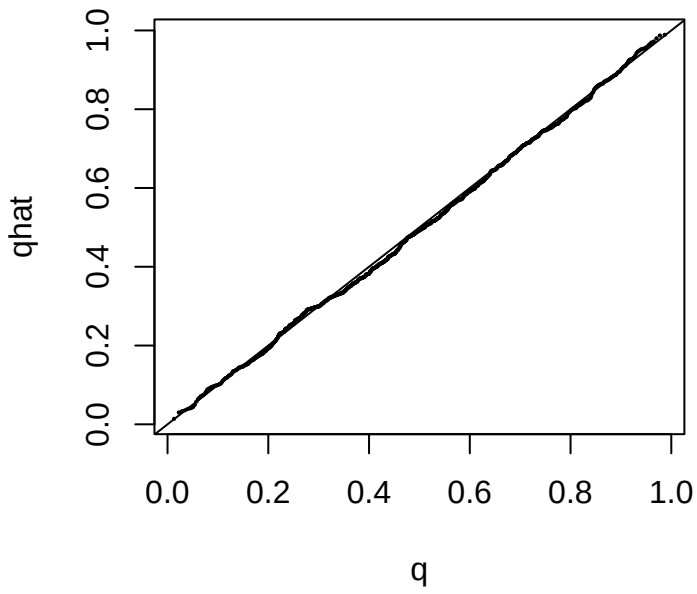
3.4 Transformation Methods

Pages 71 - 75

Compare sample data with the theoretical distr using a quantile-quantile (QQ) plot. If the sampled distr is the theoretical one, the QQ plot should be nearly linear.

```
prob_points = ppoints(1000)
q = qbeta(prob_points, 2, 2)
qhat = quantile(x, prob_points)  # quantiles of sample

qqplot(q, qhat, cex=0.15)
abline(0,1)
```



3.5 Sums and Mixtures

Let $X_1, ..., X_n$ be iid with distribution $X_j \sim X$, and let $S = X_1 + ... + X_n$. The distribution function of the sum S is called the n-fold convolution of X and denoted $F_X^{(n)}$. It is straightforward to simulate a convolution by directly generating $X_1, ..., X_n$ and computing the sum.

Example 3.10. Chisquare $\chi^2(v)$. If $Z_1, ..., Z_v$ are iid N(0,1) random variables, then $V = Z_1^2 + ... + Z_v^2$ has the $\chi^2(v)$ distribution. Theoretical moments are $E(V) = v, E(V^2) = (E(V))^2 + Var(V) = v^2 + 2v$.

```
n = 1000
v = 2
X = matrix(rnorm(n*v), n, v)^2  # matrix of squared normals
y = rowSums(X)  # sum the squared normals

mean(y)  #  the 1st moment (sample)
```

```
## [1] 1.984264
```

```
mean(y^2)  # the 2nd moment (sample)
```

```
## [1] 7.499856
```

Mixtures

A r.v. X is a *discrete* mixture if the distribution of X is a weighted sum $F(x) = \sum \theta_i F_{X_i}(x)$ for some sequence of rvs $X_1, X_2, ...$ and $\theta_i > 0$ such that $\sum_i \theta_i = 1$. The constants θ_i are called the mixing weights or mixing probabilities.

Example 3.12. Mixture of several gamma distrs. The mixture is

$$F_X(x) = \sum_{i=1}^{5} \theta_i F_{X_i}(x),$$

where

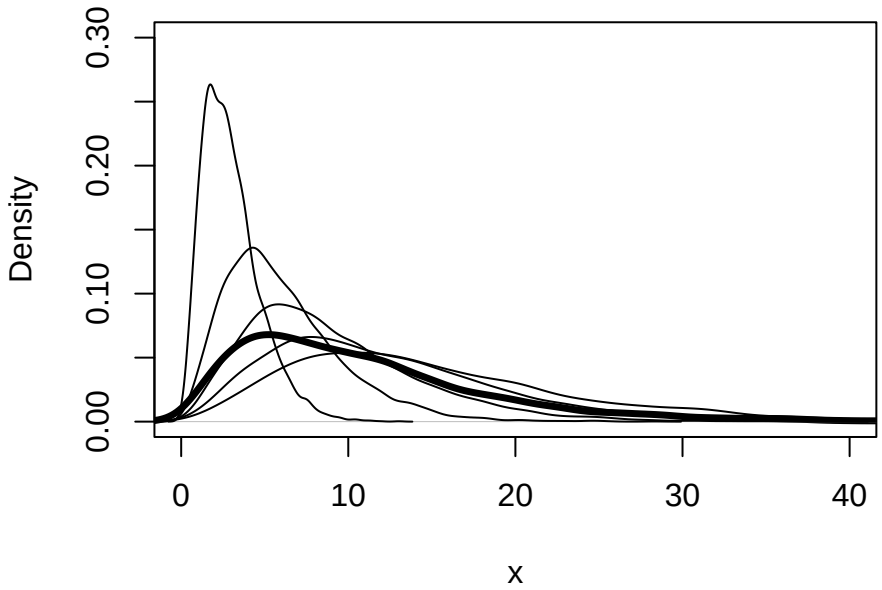
$$X_i \sim Gamma(shape = 3, rate = 1/i) \text{ and } \theta_i = i/15, i = 1, ..., 5$$

Note the first param of Gamma distr means the number of events to wait for, the 2nd controls how fast events happen. A greater shape averages out waiting times making the distribution less skewed or more symmetric.

```
n = 5000  # a random sample of size n
# select 1 gamma distr among 5, using mixing prob theta
k = sample(1:5, size=n, replace = T, prob = (1:5)/15)
rate = 1/k  # set the rate parameter
x = rgamma(n, shape=3, rate = rate)

# plot the density of the mixture
plot(density(x), xlab="x", main="", lwd=3.5, ylim=c(0, 0.3), xlim=c(0,40))

# plot the densities of the components
for (i in 1:5){
  lines(density(rgamma(n, 3, 1/i)))
}
```



3.6 Multivariate Distributions

Pages 83 - 95

Chap 4 Generating Random Processes

A stochastic process is a collection $\{X(t) : t \in T\}$ of random variables indexed by the set T, which usually represents time. The index set T could be discrete or continuous. The set of possible values $X(t)$ (at a fixed time point) can take is the state space, which also can be discrete or continuous.

4.2 Brownian Motion

One of the most fundamental models in finance and science is Brownian Motion or Wiener Process. The real-valued stochastic process $\{W_t\}_{t\geq 0}$ has

1. $W_0 = 0$ almost surely (Prob = 1, but not always)
2. W_t is almost surely continuous
3. $W_t - W_s \sim N(0, t-s), 0 \leq s < t$. The increment is normal with mean = 0 and variance = time difference
4. If $0 \leq s' \leq t' \leq s < t$, then $W_{t'} - W_{s'}$ is independent of $W_t - W_s$ (independent increments)

The d-dimensional BM is an R^d-valued process $W(t) = (W_1(t), W_2(t), ..., W_d(t))$ where the components $W_j(t)$ are each independent 1-d BM.

Example 4.7. Simulate a path of a BM on an internal $[0, T]$. First divide the internal into $n+1$ sub-intervals with points $0 = t_0 < t_1 < ... < t_i < ... < t_n = T$ and then generate a realization $W(\omega)$ (ω means a complete path outcome, a trajectory) as follows

1. set $W_0(\omega) = 0$
2. For $i = 1$ to n do:

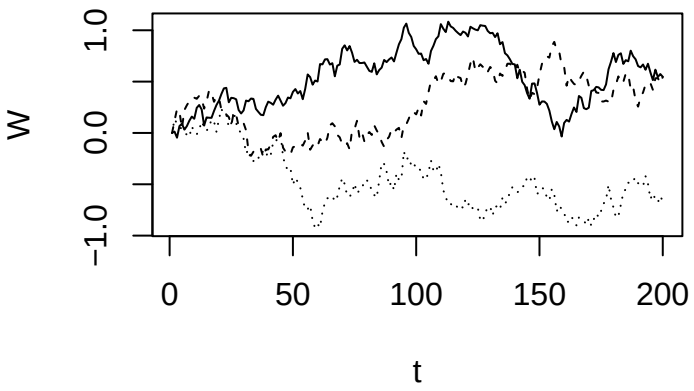
(a) Randomly generate a standard normal Z_i and compute its standard deviation $\sigma_i = \sqrt{t_i - t_{i-1}}$

(b) Set $W_{t_i} = W_{t_{i-1}} + \sigma_i Z_i$

(c) For points between t_{i-1} and t_i use linear interpolation to approximate $W_t(\omega)$

```
simBM = function(n, T){
  times = seq(0, T, length = n+1)
  z = rnorm(n)   # generate standard normal in (a)
  s = sqrt(diff(times))  # get SD in (a)
  w = rep(0, n)  # initialize W as in (b)
  for (i in 2:n){
    w[i] = w[i-1] + s[i] * z[i]  # implement (b)
  }
  return (list(w=w, t=times))
}

# generate 3 independent BMs on [0,1]
set.seed(1)
n = 200
x1 = simBM(n,1)
x2 = simBM(n,1)
x3 = simBM(n,1)
r = range(c(x1$w, x2$w, x3$w))
plot(x1$w, type="l", main="", xlab="t", ylab="W", ylim=r)
lines(x2$w, lty=2)  # interpolate between endpoints as in (c)
lines(x3$w, lty=3)
```


W
1.0
0.0
−1.0
0
50
100
150
200
t