PART A MATHEMATICS AND STATISTICS, SIMULATION AND STATISTICAL PROGRAMMING: SIMULATION LECTURES

GEOFF NICHOLLS

Contents

1. Organisation

Aims and Objectives. Building on Part A probability and Mods statistics, this course introduces Monte Carlo methods, collectively one of the most important toolkits for modern statistical inference. In parallel, students are taught programming in R, a programming language widely used in statistics. Lectures alternate between Monte Carlo methods and Statistical Programming so that students learn to programme by writing simulation algorithms.

Synopsis. (1) Simulation: Transformation methods. Rejection sampling including proof for a scalar random variable, Importance Sampling. Unbiased and consistent IS estimators. MCMC including the Metropolis-Hastings algorithm. (2) Statistical Programming: Numbers, strings, vectors, matrices, data frames and lists, and

Date: 14 lectures, Hilary 2014.

Boolean variables in R. Calling functions. Input and Output. Writing functions and flow control. Scope. Recursion. Runtime as a function of input size. Solving systems of linear equations. Numerical stability. Regression. Monte Carlo and optimisation examples for elementary Bayesian inference.

Course Structure. The workload of this course is equivalent to an 16-lecture course. There are 14 lectures and 6 practicals. Lectures 4-5pm Mondays in the Lecture Theater, Statistics Department, 1 South Parks Road will focus on Simulation. Lectures 2-3pm Fridays in weeks 2-6 and 8 will take place in the Evenlode Room of the OUCS building 13 Banbury road. These lectures focus on Statistical Programming. See

$$\texttt{http://www.oucs.ox.ac.uk/about/}$$

if you need help finding this. The OUCS lectures are followed by practical teaching sessions 3-4pm Fridays in weeks 2-6 and 8 in the same room.

1.1. **R.** R is a high quality open source software package for statistical computing. We will use R for the Statistical Programming segment. You may find it useful for checking your understanding of the applied probability we use in simulation. You will find it convenient to install R on your own computer. The software, as well as manuals and introductory tutorials, are available from

$$\texttt{http://www.r-project.org/}$$

Problem sheets in classes will include a separate section with some optional examples of simulation using R.

1.2. **Classes.** There are four classes this term: classes 4-5pm on Tuesdays in weeks 3 and 7 and 10-11am on Fridays in week 5 and 8.

Texts.

Reading. The following texts have a large overlap with the course.

W.J. Braun and D.J. Murdoch, "A First Course in Statistical Programming with R". CUP 2007
C.P. Robert and G Casella, "Introducing Monte Carlo Methods with R"

Reference. The last two listed are rather advanced.

W. Venables and B.D. Ripley, "Modern Applied Statistics with S"
J.R Norris, "Markov Chains", CUP, 1997
S.M Ross, "Simulation", Elsevier, 4th edition, 2006
C.P. Robert and G Casella, "Monte Carlo Statistical Methods", Springer, 2004
B.D Ripley, "Stochastic Simulation", Wiley, 1987

Geoff Nicholls
nicholls@stats.ox.ac.uk

2. Introduction

Computational statistics is changing the way we do statistics. We can carry out the statistical inference the math tells us we should be doing, without essential compromise. In this way computation has made the subject more 'principled'. Monte Carlo methods are a large part of this. Monte Carlo equips us to estimate the values of very complex integrals, which is to say we can estimate statistical expectations. Computing expectations is one of the fundamental operations of statistics, something we need to do to summarize data, fit parameters, test hypotheses and choose and average models. It is one of the last steps in many chains of inference.

The problem of finding efficient and provably correct Monte Carlo Algorithms is a problem in applied probability. You know some probability from last term. The problem of implementing these Monte Carlo algorithms, running them on the numbers, and realizing the inference on a computer is a problem of Statistical Programming. Take programming seriously. Too many scientists (including 'data scientists') sit down at a computer and try to implement a complex algorithm without planning and structuring the code they produce. This is a good way to waste your time. In order to avoid this we build complex programmes in self-contained pieces and test the pieces (modularity). Write code that other people can read and make sense of (be literate) and make use of. Dont reproduce the same pieces of code in many different places.

In the following we always assume we have standard uniform random numbers, $X \sim U(0,1)$ at our disposal and build other random numbers by taking various functions of these numbers. A discussion of the problem of generating uniform random numbers on a computer is part philosophy, and part number theory. See the Ripley text for algorithms and a discussion of the issues involved. You may feel that random numbers generated by a deterministic device like a computer cannot be truly random, and you would be correct. However they behave, for some purposes, like random numbers, and that is often enough. The problem of defining exactly what randomness is turns out to be a deep problem. Roberts and Cassela mention some of the literature.

3. Inversion

This is the base method. It is used for simulation of scalar random variables with only very simple pmf or pdf, since we need the cdf of the random variable.

Let X be a scalar random variable with cumulative distribution function (cdf) $F(x) = \Pr(X \leq x)$ at $X = x$ in some space of states Ω. Let $F^{-1}(u)$ be the smallest value of x such that $F(x)$ is greater than or equal to u. If F is continuous and strictly increasing then this is just the inverse of F. Simulation by the method of inversion exploits the fact that if $U \sim U(0,1)$ and $X = F^{-1}(U)$ then $X \sim F$ (meaning, the rv X is distributed with cdf F). This follows (for the simple case) by $\Pr(X \leq x) = \Pr(F^{-1}(U) \leq x) = \Pr(U \leq F(x)) = F(x)$.

Example 3.1. if we want $X \sim \text{Exp}(r)$ (*ie* $X \sim f_X$ with $f_X(x) = r\exp(-rx)$) then $F(x) = 1 - \exp(-rx)$ and

$$F^{-1}(u) = -(1/r)\log(1-u).$$

The algorithm is

Algorithm 3.1.
$U \sim U(0,1)$
$X \leftarrow -\log(U)/r$

since U and $1-U$ have the same distribution, so $X \leftarrow -\log(U)$ will do.

```
#Example Inversion Continuous Exponential
n<-100000
r<-0.5
U<-runif(n)             #n U[0,1]'s
X<--log(U)/r            #n Exp(r)'s

#check
mean(X)                 #should equal 1/r with a sd.dev by CLT
sd(X)/sqrt(n)           #or in this case about sqrt(1/n)/r
```

Taking the smallest value, *ie* defining $F^{-1}(u) = \min(z; F(z) \geq u)$ allows us to handle an arbitrary cdf, without the continuity and strict increasing conditions. See Ripley page 59 or Roberts and Casella page 39 for details beyond the scope of the course.

If $X \sim F$ with X a discrete r.v. with probability mass function (pmf) $p(x)$ then $F(x) = \sum_{i=0}^{x} p(i)$ and $F^{-1}(u)$ is x such that

$$\sum_{i=0}^{x-1} p(i) < u < \sum_{i=0}^{x} p(i)$$

with the $LHS \equiv 0$ if $x = 0$.

Example 3.2. If $0 < p < 1$ and $q = 1-p$, and we want to simulate $X \sim$ Geometric(p) then $p(x) = pq^{x-1}$ and $F(x) = 1 - q^x$ for $x \in \mathbb{N}$. The smallest natural number x giving $1 - q^x \geq u$ is the smallest $x \geq 1$ satisfying $x \geq \log(1-u)/\log(q)$ (since $\log(q) < 0$), and this is given by $x = F^{-1}(u)$ where

$$F^{-1}(u) = \left\lceil \frac{\log(1-u)}{\log(q)} \right\rceil$$

when $\lceil x \rceil$ rounds up and we could replace $1-U$ with U.

```
#Example Inversion Discrete Geometric
n<-100000
p<-0.3
U<-runif(n)             #n U[0,1]'s
X<-ceiling(log(U)/log(1-p)) #n Geometric(p)'s

#check
mean(X==1)              #should equal p with a sd.dev by CLT
sd(X==1)/sqrt(n)            #or in this case about sqrt(p*(1-p)/n)
```

4. Transformation methods

Suppose we have a rv $Y \sim Q,\ Y \in \Omega_Q$ which we **can** simulate (eg, by inversion) and some other rv $X \sim P,\ X \in \Omega_P$ which we **wish** to simulate. It may be that we can find a function $f : \Omega_Q \to \Omega_P$ with the property that if we simulate $Y \sim Q$ and then set $X = f(Y)$ then we get $X \sim P$. Inversion is a special case of this idea, since $X = F^{-1}(U)$ so Q is the uniform distribution on $0 \leq U \leq 1$ and P is the distribution with cdf F used in inversion. We may generalize this idea to take functions of collections of rv with different distributions.

Example 4.1. If for $i = 1, 2, 3, ..., a$, Y_i are iid rv distributed $Y_i \sim \text{Exp}(1)$ (we can simulate these as above) and

$$X = \beta^{-1} \sum_{i=1}^{a} Y_i$$

then $X \sim \Gamma(a, \beta)$, so now we can simulate Gamma-distributed variates (so long as the shape parameter a is integer). How does this work? The Gamma(a, β) density is

$$f_X(x) \propto x^{a-1} \exp(-x\beta).$$

The MGF of a random variable X is $M_X(t) = E(e^{tX})$ and the MGF of a sum of rv is the product of their MGF's. Since the MGF for Y_i/β is $1/(1 - t/\beta)$, the MGF of X must be $(1 - t/\beta)^{-a}$, and that is the MGF of a Gamma(a, β) rv.

```
#Example Transformation Gamma
#in this example we will simulate X~Gamma(7,0.5)
n<-10000
a<-7
b<-0.5
U<-matrix(rexp(a*n),a,n)    #rexp() is the built in R Exp(1), U is a x n matrix
X<-apply(U,2,'sum')/b       #of Exp(1)'s, so sum columns for n Gamma(a,b)'s

#check
mean(X)                     #should equal a/b=14 with a sd.dev by CLT
sd(X)/sqrt(n)               #or in this case about sqrt(a/n)/b
```

The transformation method is used in the Box Muller algorithm for simulation of normal random variables. Since, for any particular application, very large numbers of rv may need to be simulated, there is a great deal of emphasis on computational speed and efficiency in the design of simulation algorithms. Variations on the Box-Muller algorithm give the fastest simulation algorithms for normal random variables, in many applications of practical interest. The algorithm is based on the following observation.

Example 4.2. If U_1 and U_2 are independent $U(0, 1)$ rv, and

$$\begin{aligned} X_1 &= \sqrt{-2\log(U_1)}\cos(2\pi U_2) \\ X_2 &= \sqrt{-2\log(U_1)}\sin(2\pi U_2) \end{aligned}$$

then X_1 and X_2 are independent (!) standard normal random variables.

Proof: think of (X_1, X_2) as a random point in the plane. In polar coordinates this point has radius $R = \sqrt{-2\log(U_1)}$ (so $R^2 \sim \text{Exp}(0.5)$) and angle $\Theta = 2\pi U_2$

(with $\Theta \sim U(0, 2\pi)$). In order to get the joint density $f_{X_1,X_2}(x, y)$ of X_1 and X_2 we make a change of variables from Θ, R^2 (notice, not Θ, R since it is R^2 for which we have the distribution) to X_1, X_2:

$$f_{X_1,X_2}(x, y) = f_{R^2,\Theta}(r^2, \theta) \left| \frac{\partial(r^2, \theta)}{\partial(x, y)} \right|.$$

Now $x = r\cos(\theta), y = r\sin(\theta)$ so $\partial\theta/\partial x = -y/r^2$ *etc* and

$$f_{X_1,X_2}(x, y) = \frac{1}{2}\exp(-r^2/2) \times \frac{1}{2\pi} \times \begin{vmatrix} 2x & -y/r^2 \\ 2y & x/r^2 \end{vmatrix}.$$

so that $f_{X_1,X_2}(x, y) = \exp(-x^2/2 - y^2/2)/(2\pi)$.

```
#example transformation Box-Muller
n<-100000
U<-matrix(runif(2*n),2,n)
X<-sqrt(-2*log(U[1,]))*cos(2*pi*U[2,])
X<-c(X,sqrt(-2*log(U[1,]))*sin(2*pi*U[2,])) #concatenate batches of N(0,1)'s

#check
mean(X)                                     #should equal 0 with a sd.dev by CLT
sd(X)/sqrt(2*n)                             #or in this case about 1/sqrt(2*n)
```

Note: see Ross page 80 sec 5.3 for a strategy for avoiding the sin and cos evaluations (Problem 2.8 page 63 of Robert and Casella) and the Box Muller algorithm itself at the bottom of Page 81.

5. Multivariate Normal

This is a multivariate application of the transform method to a distribution of particular importance.

Let $\mu = (\mu_1, \mu_2, ..., \mu_m)^T$ be a (column) vector of m mean values,

$$X = (X_1, X_2, ..., X_m)^T$$

a vector of normal random variables with $E(X_i) = \mu_i$ and $\mathrm{cov}(X_i, X_j) = \Sigma_{i,j}$. The multivariate normal distribution for X is written $X \sim N(\mu, \Sigma)$ (sometimes $X \sim MVN(\mu, \Sigma)$). In this case the joint density of the collection of rv X at $X = x$ in $\mathbb{R}^m$ is

$$f_X(x) = (2\pi)^{-m/2} \det(\Sigma)^{-1/2} \exp\left(-(x-\mu)^T \Sigma^{-1} (x-\mu)/2\right).$$

Theorem 5.1. *Let $Z = (Z_1, Z_2, ..., Z_m)$ be a collection of m independent standard normal random variables. Let L be a real $m \times m$ matrix satisfying $LL^T = \Sigma$, and $X = LZ + \mu$. Then $X \sim N(\mu, \Sigma)$.*

Proof: The joint density of the new variables (the components of X) is

$$f_X(x) = f_Z(z(x))|\partial z/\partial x|.$$

Now

$$f_Z(z) = \exp(-z^T z/2)/(2\pi)^m$$

and $|\partial z/\partial x| = \det(L^{-1})$. Also, $\det(L) = \det(L^T)$ so $\det(L)^2 = \det(\Sigma)$, and $\det(L^{-1}) = 1/\det(L)$ so $\det(L^{-1}) = \det(\Sigma)^{-1/2}$. Also, $z^T z = (x-\mu)^T[L^{-1}]^T L^{-1}(x-\mu)$ so $z^T z = (x-\mu)^T\Sigma^{-1}(x-\mu)$ so X is indeed $N(\mu, \Sigma)$.

There are many ways to define L so that $LL^T = \Sigma$. For example, if $\Sigma = VDV^T$ is the diagonalization, or spectral representation, of Σ, then $L = VD^{1/2}$ would do. The eigenvalues of $\Sigma = E((X-\mu)(X-\mu)^T)$ are not negative so this representation is real. We can assume the eigenvalues are actually strictly greater than zero (Σ positive definite) without loss of generality, since zero eigenvalues arise when some of the X_i are linear combinations of others, so the corresponding row may be removed. In this case the LU factorization $\Sigma = LU$ into lower and upper triangular matrices has a special form with $L = U^T$, and a special name, the Cholesky factorization. The Cholesky factorization $L = \text{chol}(\Sigma)$ is particularly convenient for computational use. It (*ie*, L) is the unique lower triangular matrix satisfying $LL^T = \Sigma$. See the Part B course in numerical analysis for details of fast chol(Σ) computation.

```
#MVN, X~N(mu,s), use the cholesky factorization of the covariance matrix
#In this exam[pe we will take m=dim(mu)=2, mu=(-1,1), covariance matrix s
#s=| 5 -3|
#  |-3  4|
mu<-c(-1,1)
s<-matrix(c(5,-3,-3,4),2,2)
u<-chol(s)                  #s=t(u)%*%u so my L is t(u) here
(X<-t(u)%*%rnorm(2)+mu)     #X~N(mu,s)

#check - generate n sets of MVN vectors X
n<-10000
X<-t(u)%*%matrix(rnorm(2*n),2,n)+mu
plot(t(X),asp=1)
#estimate mean mu-hat and covariance s-hat
apply(X,1,'mean')           #average along the rows
cov(t(X))                   #cov() is covariances of row vectors
```

6. Rejection

Inversion is restricted to the univariate case. Also, we need to have the cdf of target distribution in a form that makes it at least numerically easy to invert. We found a transformation to convert independent standard normal variates into (correlated) multivariate normal random variables. That worked because the normal distribution is 'special'. We cant rely on finding a suitable transformation to make up any given distribution.

We will start with simulation of X a discrete rv. The following sentence may sound familiar. Suppose that for $x \in \Omega$ we have a probability mass function $p(x)$ (the target distribution) which we *want to* sample, and another pmf $q(x)$ defined on the same space which we *can* sample.

Theorem 6.1. *Suppose we can find a constant M satisfying $M \geq p(x)/q(x)$ for all $x \in \Omega$. The following 'Rejection algorithm' returns $X \sim p$.*

Algorithm 6.1.

1 Let $Y \sim q$ and $U \sim U(0,1)$. Simulate $Y = y$ and $U = u$.
2 If $u \leq p(y)/(Mq(y))$ then stop and return $X = y$, and otherwise, start again at 1.

Since this is rather important idea, we will look at a couple of ways of proving this result. The second proof is short but hides some subtleties. The first is more explicit.

Proof (1): Let $\Pr(X = i)$ give the probability for the value of X returned by the algorithm to equal i. I will partition on the the number of times through the loop. In order to end up with $X = i$ we could draw $Y = i$ at the first step and accept it, or we could reject whatever was drawn at the first pass, and then at the second pass draw $Y = i$ and accept it, and so on. Events at each pass through the loop are independent of events in other passes, so

$$\begin{aligned}\Pr(X=i) &= \sum_{n=1}^{\infty} \Pr(\text{reject } n-1 \text{ times, then draw } Y=i \text{ and accept it}) \\ &= \sum_{n=1}^{\infty} \Pr(\text{reject } Y)^{n-1} \Pr(\text{draw } Y=i \text{ and accept it}). \qquad (6.1)\end{aligned}$$

At a particular pass through the loop the probability we draw $Y = i$ and accept is

$$\begin{aligned}\Pr(\text{draw } Y=i \text{ and accept it}) &= \Pr(\text{accept } Y|Y=i)\ \Pr(\text{draw } Y=i) \\ &= \Pr\left(U \leq \frac{p(i)}{Mq(i)}\right) q(i) \\ &= \frac{p(i)}{M}. \qquad (6.2)\end{aligned}$$

The probability we have a rejection at any pass through the loop is

$$\begin{aligned}\Pr(\text{reject } Y) &= \sum_{j\in\Omega} \Pr(\text{draw } Y=j \text{ and reject it}) \\ &= \sum_{j\in\Omega} \Pr(\text{reject } Y|Y=j) \Pr(\text{draw } Y=j) \\ &= \sum_{j\in\Omega} \Pr\left(U > \frac{p(j)}{Mq(j)}\right) q(j) \\ &= \sum_{j\in\Omega} \left(1 - \frac{p(j)}{Mq(j)}\right) q(j) \\ &= 1 - \frac{1}{M}\end{aligned}$$

since both $\sum_j q(j) = \sum_j p(j) = 1$ to get the last line. We succeed or fail at each pass through the loop independently of events in previous loops. Picking up where

we left Eqn 6.1,

$$\begin{aligned}\Pr(X=i) &= \sum_{n=1}^{\infty}\left(1-\frac{1}{M}\right)^{n-1}\frac{p(i)}{M}\\ &= \frac{p(i)}{M}\frac{1}{1-(1-M^{-1})}\\ &= p(i),\end{aligned}$$

as supposed. Notice that the number of accept/reject trials (the number of times N we have to repeat steps 1 and 2 before we accept, and get one sample returned) has a geometric distribution with success probability $1/M$, so the mean number of trials is M. If we are conservative, in order to ensure the bound $M \geq p/q$ holds, and choose large M, we will have an inefficient algorithm.

Proof (2): here is a proof that the rejection algorithm returns realisations of X distributed with probability density $p(x)$ for X a continuous scalar rv. The proof is easily adapted to X discrete, by changing integrals to sums. Think of the Y and U values simulated at each cycle through the loop as a pair. We generate lots of these pairs (U, Y) and take as our realisations of X those Y-values belonging to pairs that satisfy $U \leq p(Y)/(Mq(Y))$. This is the same as saying that the cdf of X is the cdf of Y conditioned on $U \leq p(Y)/(Mq(Y))$.

Let X be the value returned by a call to Alg. 6.1. If $p_{U,Y}(u, y)$ is the joint density of U and Y then $p_{U,Y}(u, y) = q(y)$ because U and Y are independent (before we condition) and U has the uniform density on $0 < U < 1$, which is equal one.

$$\begin{aligned}\Pr(X<x) &= \Pr(Y<x|U<p(Y)/Mq(Y))\\ &= \frac{\Pr(Y<x, U<p(Y)/Mq(Y))}{\Pr(U<p(Y)/Mq(Y))}\\ &= \frac{\int_{-\infty}^{x}\int_{0}^{p(y)/Mq(y)} p_{U,Y}(u,y)dudy}{\int_{-\infty}^{\infty}\int_{0}^{p(y)/Mq(y)} p_{U,Y}(u,y)dudy}\\ &= \frac{\int_{-\infty}^{x}\int_{0}^{p(y)/Mq(y)} q(y)dudy}{\int_{-\infty}^{\infty}\int_{0}^{p(y)/Mq(y)} q(y)dudy}\\ &= \frac{\int_{-\infty}^{x} p(y)/Mdy}{\int_{-\infty}^{\infty} p(y)/Mdy}\\ &= \int_{-\infty}^{x} p(y)dy\end{aligned}$$

since $\int_{-\infty}^{\infty} p(y)dy = 1$ because $p(y)$ is the probability density of a real scalar.

Example 6.1. Here is an algorithm simulating a rv $X \sim$Beta(α, β), which works for $\alpha, \beta \geq 1$.

The beta density $p(x)$ is

$$p(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{Z_p} \qquad \text{for } 0 < x < 1 \text{ and } \alpha, \beta > 0$$

where

$$Z_p = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

is a normalising constant. In order to apply the rejection algorithm we need to find an envelope probability density $q(x)$ and a constant $M \geq p(x)/q(x)$ for all $0 < x < 1$.

When α and β are both greater than or equal one, $p(x)$ is bounded (and otherwise, not). Since $p(x)$ has support on $[0,1]$ and is bounded, we can simply use the uniform density $q(x) = 1$, *ie* the constant function, as our envelope, and $M = \max_{0<x<1} p(x)$ to ensure $M \geq p(x)/q(x)$ for all $0 < x < 1$.

Now, $p'(x^*) = 0$ is satisfied by $x^* = (\alpha-1)/(\alpha+\beta-2)$ so $M = p(x^*)$ is

$$M = \frac{1}{Z_p}\left(\frac{\alpha-1}{\alpha+\beta-2}\right)^{\alpha-1}\left(\frac{\beta-1}{\alpha+\beta-2}\right)^{\beta-1}$$

will do for the bound, and our acceptance probability at step 2 of Alg. 6.1 is

$$\begin{aligned} p(y)/(Mq(y)) &= \frac{y^{\alpha-1}(1-y)^{\beta-1}}{MZ_p} \\ &= \frac{y^{\alpha-1}(1-y)^{\beta-1}}{M'} \end{aligned}$$

where

$$M' = \left(\frac{\alpha-1}{\alpha+\beta-2}\right)^{\alpha-1}\left(\frac{\beta-1}{\alpha+\beta-2}\right)^{\beta-1}$$

The point is that the factor Z_p cancels with a corresponding factor of Z_p in the expression for M so we needn't work it out. This kind of simplification will always occur, as we shall see.

Larger M would have given a correct algorithm - can you think why we want the minimum upper bound?

```
#Rejection, beta(a,b) for both a,b>=1

my_big_ab_beta<-function(a=1,b=1) {
    #simulate X~Beta(a,b) variate, defalts to U(0,1)
    if (a<1 || b<1) stop('a<1 or b<1');
    M<-(a-1)^(a-1)*(b-1)^(b-1)*(a+b-2)^(2-a-b)
    finished<-FALSE
    while (!finished) {
        Y<-runif(1)
        U<-runif(1)
        accept_prob<-Y^(a-1)*(1-Y)^(b-1)/M
        finished<-(U<accept_prob)
    }
    X<-Y
X}

my_big_ab_beta(a=1.5,b=2.5)

#check, example values a=1.5,b=2.5
```

```
a<-1.5; b<-2.5;
n<-100000
X<-rep(NA,n)        #clear X before the loop - its entries are NA
for (i in 1:n) {
    X[i]<-my_big_ab_beta(a,b)
}
mean(X);a/(a+b)         #should equal a/(a+b) with a sd.dev by CLT
sd(X)/sqrt(n)           #or in this case about sqrt(a*b/(a+b+1)/n)/(a+b)
```

This gets inefficient when a or b are big (the beta density is then sharply peaked, so the ratio p/Mq is typically very small and it takes many trials to get an acceptance).

Exercise: when I first wrote the function `my_big_ab_beta()` above, I forgot to divide by M in the line `accept_prob<-Y^(a-1)*(1-Y)^(b-1)/M` but everything still seemed to be correct (the means and variances were right). Can you explain why the algorithm was still correct, and why I am better off with M in place, as above?

A note on normalising constants: notice the way the normalising constant in p canceled the corresponding constant in M. If $\tilde{p}$ and $\tilde{q}$ are unnormalised densities with $p = \tilde{p}/Z_p$ and $q = \tilde{q}/Z_q$ then $M \geq p/q$ iff $M' \geq \tilde{p}/\tilde{q}$, with $M' = Z_p M/Z_q$. This means we can throw the normalising constants out at the start: if we can find M' to bound $\tilde{p}/\tilde{q}$ then it is correct to accept with probability $\tilde{p}/M'\tilde{q}$ in the rejection algorithm. In this case the mean number N of accept/reject trials will equal $Z_q M'/Z_p$ (that is, M again).

The art of making good rejection algorithms for a given target pdf p is to find some probability density q which (a) you can can sample by elementary methods, and (b) is 'close' to p in shape, so that the ratio $p(x)/q(x)$ is close to one for any x value you are likely to draw from p.

Example 6.2. Here is an algorithm simulating a rv $X \sim$Gamma(α, β) (example from R& C) which works for any $a \geq 1$ (not just integers).

The Gamma density $p(x)$ is

$$p(x) = \frac{x^{\alpha-1}\exp(-\beta x)}{Z_p} \qquad \text{for } x > 0 \text{ and } \alpha, \beta > 0,$$

with $Z_p = \Gamma(\alpha)/\beta^\alpha$, so

$$\tilde{p} = x^{\alpha-1}\exp(-\beta x)$$

will do as an unnormalized form for p. When $\alpha = a$ is a positive integer we can simulate $X \sim$ Gamma(a, β) by adding a independent Exp(β) rv's, $Y_i \sim$ Exp(β), $X = \sum_{i=1}^{a} Y_i$. How could we simulate $X \sim$ Gamma(α, β) for non-integer α? We can sample densities 'close' in shape to Gamma(α, β) since we can sample Gamma$(\lfloor\alpha\rfloor, \beta)$. Perhaps this, or something like it, would make an envelope density?

Let $a = \lfloor\alpha\rfloor$ and use Gamma(a, b) as the envelope, so $Y \sim$ Gamma(a, b) for integer $a \geq 1$ and some $b > 0$. In order to use rejection, we need to calculate the acceptance probability p/Mq. The density of Y is

$$q(y) = \frac{x^{a-1}\exp(-bx)}{Z_q}$$

so

$$\tilde{q}(y) = x^{a-1}\exp(-bx)$$

will do as our unnormalised envelope function. The problem then is to bound the ratio

$$\tilde{p}/\tilde{q} = x^{\alpha-a}\exp(-(\beta-b)x).$$

Is this bounded? Consider (a) $x \to 0$ and (b) $x \to \infty$. For (a) we need $a \leq \alpha$ so $a = \lfloor \alpha \rfloor$ is fine. For (b) we need $b < \beta$ (not $b = \beta$ since we need the exponential to kill off the growth of $x^{\alpha-a}$).

Given that we have chosen a and b so the ratio is bounded, we should now compute the bound. Now $d(\tilde{p}/\tilde{q})/dx = 0$ at $x = (\alpha-a)/(\beta-b)$ (and this must be a maximum at $x \geq 0$ under our conditions on a and b), so $\tilde{p}/\tilde{q} \leq M$ for all $x \geq 0$ if

$$M = \left(\frac{\alpha-a}{\beta-b}\right)^{\alpha-a}\exp(-(\alpha-a)).$$

We will accept Y at step 2 of Alg. 6.1 if $U \leq Y^{\alpha-a}\exp(-(\beta-b)Y)/M$.

Exercise: how to choose b? Any $0 \leq b < \beta$ will do, but is there a best choice? One idea would be to choose b to minimize the expected number of Y-simulations per sample X output. Since the number of trials N say is Geometric, with success probability Z_p/MZ_q, the expected number of trials is $E(N) = Z_qM/Z_p$. Now $Z_p = \Gamma(\alpha)\beta^{-\alpha}$ where Γ is the Gamma function related to the factorial. Show that the optimal b solves $d(b^{-a}(\beta-b)^{-\alpha+a})/db = 0$ so use $b = \beta(a/\alpha)$ for best results.

```
#Rejection, Gamma(a,b)

my_gamma<-function(a=1,b=1) {
    #simulate X~Gamma(a,b) variate, defalts to Exp(1)
    if (a<1 || b<=0) stop('a<1 or b<=0');
    aq<-floor(a)
    bq<-b*(aq/a)                  #best choice but any 0<bq<b OK
    del_a<-a-aq
    del_b<-b-bq
    finished<-FALSE
    while (!finished) {
        Y<-sum(rexp(aq))/bq     #Y~Gamma(aq,bq)
        U<-runif(1)
        accept_prob<-(Y*del_b/del_a)^del_a*exp(-del_b*Y+del_a)
        finished<-(U<accept_prob)
    }
    X<-Y
X}

my_gamma(a=7.5,b=0.5)

#check, in this example, a=7.5,b=0.5
a<-7.5; b<-0.5;
n<-100000
X<-rep(NA,n)            #clear X before the loop - its entries are NA
for (i in 1:n) {
```

```
    X[i]<-my_gamma(a,b)
}
mean(X)                  #should equal a/b with a sd.dev by CLT
sd(X)/sqrt(n)            #or in this case about sqrt(a/n)/b
```

Exercise: modify the function `my_gamma()` to return the number of trials needed in order to generate X, and check that the mean number of trials is MZ_q/Z_p.

Notice that it is important to find q which is heavy tailed compared to p, so, $p(x)/q(x)$ goes to zero as x tends to either end of the support of p (and of course the ratio has also to be bounded throughout the support, but that is usually the easy bit).

7. Importance sampling

Importance sampling is, among other things, a strategy for recycling samples. It is useful also when we need to make an accurate estimate of the probability a random variable exceeds some very high threshold. In this context the naive estimator (proportion of samples over threshold) has a high variance (relative to the mean). In this context it is referred to as a *variance reduction* technique.

There is a slight variation on the basic set up: we can generate samples distributed according to q but we want to estimate an expectation that depends on p (before it was "but we want samples distributed according to p"), so we want to estimate $E_p(f(X))$ for some function f. In importance sampling we avoid sampling the target distribution p. Instead, we take samples distributed according to q and *reweight* them.

7.1. **Importance Sampling (I).** Here is the key idea. Let $Y_i \sim q$, $i = 1, 2, ..., n$ be iid continuous random variables distributed for $Y_i \in \Omega$ with density q. We will require $p(x) > 0 \Rightarrow q(x) > 0$ for all $x \in \Omega$ (which is weaker than p/q bounded as in rejection). Then

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} p(Y_i)f(Y_i)/q(Y_i)$$

is an unbiased estimator for $E_p(f(X))$ since

$$\begin{aligned} E_q(\overline{f}) &= \frac{1}{n}\sum_{i=1}^{n} E_q\left(\frac{p(Y_i)}{q(Y_i)}f(Y_i)\right) \\ &= E_q\left(\frac{p(Y_1)}{q(Y_1)}f(Y_1)\right) \\ &= \int_\Omega \left[\frac{p(y)}{q(y)}f(y)\right] \times q(y)\,dy \\ &= \int_\Omega p(y)f(y)\,dy \\ &= E_p(f(X)). \end{aligned}$$

All this works in a very similar way in the case where X and the Y_i are discrete rv.

We call $w(x) = p(x)/q(x)$ the weight function and $w_i = w(y_i)$ the weights for importance sampling estimates of expectations in p using samples $Y_i = y_i$ from q and write

$$\overline{f} = \frac{1}{n}\sum_{i=1}^{n} w_i f(y_i). \tag{7.1}$$

Here $\overline{f}$ is called an IS (importance sampling) estimator for $E_p(f)$.

Example 7.1. Say we have simulated $Y_i \sim$Gamma(a, b) and we want to estimate $E_p(f(X))$ where $X \sim$Gamma(α, β).

Recall that the Gamma(α, β) density is

$$p(x) = x^{\alpha-1}\exp(-\beta x)\beta^{\alpha}/\Gamma(\alpha).$$

Now p/q is

$$\frac{p(x)}{q(x)} = x^{\alpha-a}e^{-(\beta-b)x}\frac{\Gamma(a)\beta^{\alpha}}{\Gamma(\alpha)b^{a}}$$

so the weights are

$$w_i = {Y_i}^{\alpha-a}e^{-(\beta-b)Y_i}\frac{\Gamma(a)\beta^{\alpha}}{\Gamma(\alpha)b^{a}},$$

and we estimate $E_p(f(X))$ via $\overline{f}$ as in Eqn 7.1.

```
#in this example f(x)=x, X~Gamma(7.8,2), Y~Gamma(7,1) and we use Importance sampling
#to estimate E(f(X))
alpha<-7.8
beta<-2

n<-100000
a<-floor(alpha)
b<-1                                    #Exercise: what is the optimal choice of b here?
U<-matrix(rexp(a*n),a,n)
Y<-apply(U,2,'sum')/b

exp_Y1<-mean(Y*Y^(alpha-a)*exp(-(beta-b)*Y)*(gamma(a)/gamma(alpha))*(beta^alpha/b^a))

#compare estimate (exp_Y1) and exact result (E(X)=alpha/beta)) - TODO check variance
exp_Y1
alpha/beta
```

We now consider the variance of $\overline{f}$. We assume $E_p(f(X))$ and $\mathrm{var}_p(f(X))$ are finite. For f a function $h : \Omega \to R$, and $Y \sim q$, let

$$\mathrm{var}_q(f(Y)) = E_q(f(Y)^2) - E_q(f(Y))^2$$

be the variance for f. The variance of the importance-sampling estimator for $E_p(f)$ is

$$\begin{aligned}\operatorname{var}(\overline{f}) &= \frac{1}{n}\operatorname{var}_q\left(\frac{p(Y_1)}{q(Y_1)}f(Y_1)\right)\\ &= \frac{1}{n}\left(E_q\left(\frac{p^2}{q^2}f^2\right) - E_q\left(\frac{p}{q}f\right)^2\right)\\ &= \frac{1}{n}\left(E_p\left(\frac{p}{q}f^2\right) - E_p(f)^2\right). \qquad (7.2)\end{aligned}$$

(and $\operatorname{var}(\overline{f}) = (E_p\left(f^2\right) - E_p(f)^2)/n$ when $q = p$ as expected). Each time we do IS we should check that this variance is finite, otherwise our estimates are somewhat untrustworthy! We check $E_p\left(pf^2/q\right)$ is finite.

Example 7.2. Let us check that the variance of $\overline{f}$ in Example 7.1 is actually finite. The variance of $\overline{f}$ depends on two expectations, $E_p(pf^2/q)$ and $E_p(f)$. Refering to Eqn 7.2, it is enough to show that $E_p(pf^2/q)$ is finite. The functions of α, β, a and b appearing are finite so we can those factors throught, and begin with

$$f(X)^2\frac{p(X)}{q(X)} \propto f(X)^2 X^{\alpha-a}e^{-(\beta-b)X}.$$

The expectation of interest is

$$\begin{aligned}E_p(pf^2/q) &\propto E_p\left(f(X)^2X^{\alpha-a}\exp(-(\beta-b)X)\right)\\ &= \int_0^\infty p(x)\,f(x)^2x^{\alpha-a}\exp(-(\beta-b)x))\,dx.\\ &\leq M\int_0^\infty p(x)f(x)^2\,dx\\ &= ME_p(f^2).\end{aligned}$$

where

$$M = \max_{x>0} x^{\alpha-a}\exp(-(\beta-b)x)$$

is finite if $a < \alpha$ and $b < \beta$ (see Example 6.2). Since f has finite mean and variance in X, we know $E_p(f^2) < \infty$, and so the integral is finite. If these conditions (on the parameters) are not satisfied, our IS-estimator is useless (unless f saves the day). These same conditions applied to our rejection sampling for Gamma(α, β), though note that it is enough for M to exist, we dont have to work out its value.

Notice that the setup allows us to choose q: while p is given, q needs to cover p ($p > 0 \Rightarrow q > 0$) and be simple to sample. The requirement that the variance of the estimator be finite further constrains our choice. Referring to Eqn 7.2, we need $E_p(fp/q) = E_q(fp^2/q^2)$ to be finite. If $\operatorname{var}_p(f)$ is known finite then (as we saw in Example 7.1) it may be easy to get a sufficient condition for $\operatorname{var}_q(\overline{f})$ finite. Further analysis will depend on the details of f.

What is the choice of q that actually minimizes the variance of the importance sampling (I.S.) estimator? Look at Eqn 7.2. Suppose for the moment $f > 0$. The variance of $\overline{f}$ cannot be negative, but we can make it zero by the choice $q(x) = p(x)f(x)/E_p(f)$. Zero variance estimators! In practice we are not able to implement this choice, since the weight function is $w(x) = p/q = E_p(f)/f(x)$,

but $E_p(f)$ is the thing we are trying to estimate anyway, so we cant compute the weights in Eqn 7.1.

One important class of application of IS is to problems in which we estimate the probability for a rare event. In this case we may be able to sample p directly, but it doesn't help us. If, for example, $X \sim p$ with $\Pr(X > x) = \delta$ say, with δ very small, we may not get any samples $X_i > x$ and our estimate $\hat{\delta} = \sum_i \mathbb{I}_{X_i>x}/n$ is simply zero. By distorting the proposal distribution we can actually reduce the variance of our (IS) estimator.

Example 7.3. Let $X \sim N(\mu, \sigma^2)$ be a scalar normal rv. If we would like to estimate $\delta = \Pr(X > x)$ for some x much larger than 3σ we could *exponentially tilt* the density of X towards larger values so that we get some samples in the target region, and then allow for our tilting *via* an IS estimator. If $p(x)$ is the density of X then $q(x) = p(x)\exp(tx)/M_p(t)$ is called a tilted density of p (see eg Ross section 8.6 page 185). Here the normalizing constant $M_p(t) = E_p(\exp(tX))$ happens to be the moment generating function of X.

For p the normal density,

$$\exp(-(x-\mu)^2/2\sigma^2)\exp(tx) = \exp(-(x-\mu-t\sigma^2)^2/2\sigma^2)\exp(\mu t + t^2\sigma^2/2)$$

so the normalized tilted density is

$$q(x) = (2\pi)^{-1/2}\exp(-(x-\mu-t\sigma^2)^2/2\sigma^2),$$

normal mean $\mu + t\sigma^2$ and variance σ^2. For the normal, $M_p(t) = \exp(\mu t + t^2\sigma^2/2)$.

The IS weight function is $p/q = \exp(-tx)M_p(t)$ so

$$w(x) = \exp(-t(x-\mu-t^2\sigma^2/2)).$$

We take samples $Y_i \sim N(\mu + t\sigma^2, \sigma^2)$, compute $w_i = w(Y_i)$ and form our IS estimator for $\delta = \Pr(X > x)$ according to Eqn 7.1,

$$\hat{\delta} = \frac{1}{n}\sum_{i=1}^{n} w_i \mathbb{I}_{Y_i>x}$$

since $f(Y_i)$ here is $\mathbb{I}_{Y_i>x}$.

We havnt said how to choose t. The point here is that we want samples in the region of interest. I will choose the mean of the tiled distribution so that it equals x, then I am sure to have samples in the region of interest. I chose t so that $\mu + t\sigma^2 = x$, or $t = (x-\mu)/\sigma^2$. The densities p and q and the threshold used in the example below are depicted in Figure **??**

```
#######################################################
#IS - normal example, method I, Pr(Z>4) Z~N(0,1)
#note I chose x=4 so we could check using the naive method
#for x=6 say the advantage is even more dramatic

sigma<-1
mu<-0
x<-4

n<-1000000
```

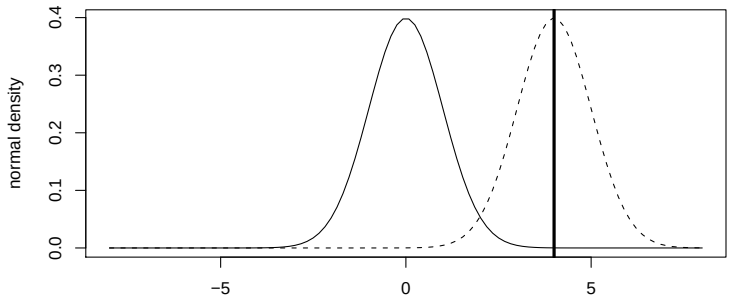


FIGURE 1. (solid) $N(0,1)$ density p. (dashed) $N(x,1)$ tilted tensity q.

```
Z<-rnorm(n)
#naive estimator
(plain_est<-mean(Z>x))

#Y are samples from the tilted distribution (weighted towards region of interest)
t<-(x-mu)/sigma^2
Y<-mu+t*sigma^2+sigma*rnorm(n)
#IS-estimator corrects for weighting
(IS_est<-mean( (Y>x)*exp(-(Y-mu)^2/(2*sigma^2)+(Y-mu-t*sigma^2)^2/(2*sigma^2))))

#The precision of our estimate improves by a factor of about 100
(plain_sd<-sd(Z>x)/sqrt(n))
#ran above, got 0.000033 +/- 0.000006 about 20% error

(IS_std<-sd( (Y>x)*exp(-(Y-mu)^2/(2*sigma^2)+(Y-mu-t*sigma^2)^2/(2*sigma^2)))/sqrt(n))
#ran above got 0.00003170 +/- 0.00000007 about 0.2% error
```

Exercise: how to choose t? Show that the t-value that minimizes the variance, minimizes $E_p(f^2p/q) = M_p(t)\int_x^\infty p(z)\exp(-tz)dz$ over t, and hence obtain a equation for the optimal tilt t.

7.2. **Importance Sampling (II).** When we introduced rejection sampling, we emphasized that normalizing constants are often unknown. In order to do IS-estimation as in Section 7.1 we need p and q normalized. There is an IS-estimator for un-normalized densities too, and it is in this form that IS-estimation is usually conducted.

Let $\tilde{p}(x) \geq 0$, $\tilde{q}(x) \geq 0$ and $f(x)$ be scalar functions of real $x \in \mathbb{R}$. Suppose the integrals $Z_p = \int_{\mathbb{R}^n} \tilde{p}(x)dx$ and $Z_q = \int_{\mathbb{R}^n} \tilde{q}(x)dx$ exist so that $p = \tilde{p}/Z_p$ and $q = \tilde{q}/Z_q$ are probability densities.

We want to estimate $E_p(f) = E_q(pf/q)$. Now the normalizing constants Z_p and Z_q are no longer available, so separate them out, writing

$$E_q(pf/q) = E_q(\tilde{p}f/\tilde{q})Z_q/Z_p.$$

We have an expectation giving Z_p/Z_q, namely $E_q(\tilde{p}/\tilde{q}) = Z_p/Z_q$. It follows that

$$E_p(f) = \frac{E_q(\tilde{p}f/\tilde{q})}{E_q(\tilde{p}/\tilde{q})}, \tag{7.3}$$

and we can estimate the numerator and denominator separately using samples distributed according to q.

Our recipe for estimation of the mean of f is then

Algorithm 7.1.

(1) Simulate $Y_i \sim q$ iid for $i = 1, 2, ..., n$ and let
$$\tilde{w}(Y_i) = \tilde{p}(Y_i)/\tilde{q}(Y_i).$$
(2) Evaluate an estimator $\overline{a}$
$$\overline{a} = \frac{1}{n}\sum_{i=1}^{n} \tilde{w}(Y_i)f(Y_i)$$
for the numerator of $E_p(f)$ in Eqn 7.3, and another estimator $\overline{b}$
$$\overline{b} = \frac{1}{n}\sum_{i=1}^{n} \tilde{w}(Y_i)$$
for the denominator.
(3) The IS-estimator for $E_p(f)$ is $\overline{f} = \overline{a}/\overline{b}$

Notice that $E_q(\overline{a}) = E_q(\tilde{p}f/\tilde{q})$ and $E_q(\overline{b}) = E_q(\tilde{p}/\tilde{q})$ so these two estimators are unbiased, even if $\overline{f}$ itself may be biased.

An estimator (such as $\overline{f}$) is *consistent* for $E_p(f)$ if $\overline{f}$ tends in probability to $E_p(f)$ as $n \to \infty$. That means that for each $\epsilon > 0$,

$$\lim_{n\to\infty} \Pr(|\overline{f} - E_p(f)| > \epsilon) = 0.$$

Exercise: show that $\overline{a}$ and $\overline{b}$ are consistent estimators (of the numerator and denominator of Eqn 7.3). (*via* Markov's Inequality or using the CLT - this is the same as Q6 of the supplementary problem sheet of Part A probability, HT08).

Theorem 7.1. *$\overline{f}$ is a consistent estimator, that is, $\overline{f}$ tends in probability to $E_p(f)$ as $n \to \infty$.*

We omit the proof which is outside the scope of this course. If you are interested in following this up, apply the delta-method, Taylor expanding $\overline{f}$ as a function of $\overline{a}$ and $\overline{b}$ about $\overline{a} = E_q(\tilde{w}f)$ and $\hat{b}_N = E_q(\tilde{w})$.

Example 7.4. Revisit Example 7.1. We wish to estimate $E_p(f(X)$ where the target distribution is $X \sim$Gamma(α, β) and $Z_p = \beta^{-\alpha}\Gamma(\alpha)$ and the proposal distribution is $X \sim$Gamma(a, b). Suppose we didn't know Z_p and Z_q (or we were not confident we could calculate it correctly or we couldn't be bothered calculating it!). Now $\tilde{p}(x)/\tilde{q}(x) = x^{\alpha-a}\exp(-(\beta - b)x)$ so our algorithm is

Algorithm 7.2.

1 Simulate $Y_i \sim$Gamma(a, b), for $i = 1, 2, ..., n$.
2 Calculate the weights

$$w_i = Y_i^{\alpha - a} \exp(-(\beta - b)Y_i).$$

3 Form the IS-estimator

$$\overline{f} = \frac{\sum_i w_i f(Y_i)}{\sum_j w_j}.$$

```
#in this example f(x)=x, X~Gamma(7.8,2), Y~Gamma(7,1) and we use Importance sampling
#to estimate E(f(X)) using the ratio estimator
alpha<-7.8
beta<-2

n<-100000
a<-floor(alpha)
b<-1                                  #Exercise: what is the optimal choice of b here?
U<-matrix(rexp(a*n),a,n)
Y<-apply(U,2,'sum')/b

#version II do not know normalizations
num_Y<-mean(Y*Y^(alpha-a)*exp(-(beta-b)*Y))
den_Y<-mean(Y^(alpha-a)*exp(-(beta-b)*Y))
exp_Y2<-num_Y/den_Y

#compare - TODO should look at variances
exp_Y2
alpha/beta

#check that our estimate of Z_p/Z_q is good
den_Y
(gamma(alpha)/gamma(a))*(b^a/beta^alpha)
```

Surprisingly, simulation studies show that importance sampling using method II gives more stable estimates of $E_p(f)$ than does method I - some authors recommend using II even when you know and can easily calculate all the normalizing constants. This is surprising as we have an extra quantity to estimate in method II. In my experience it is easy to make programming errors, even for simple normalizing functions, when using method I, so I tend to use method II.

8. Markov Chain Monte Carlo

Our purpose is to estimate $E_p(f(X))$ for $X \sim p$ for $p(x)$ some pmf (or pdf) defined for $x \in \Omega$. Up to this point we have based our estimates on iid draws from either p itself, or some proposal distribution with pmf q. In MCMC we simulate a correlated sequence $X_0, X_1, X_2, ...$ which satisfies $X_t \sim p$ (or at least X_t converges to p in distribution) and rely on the usual estimate $\hat{f} = n^{-1} \sum_i f(X_i)$. We need to review and extend some of the Markov chain material from Part A Probability.

In the following discussion we will suppose the space of states of X is finite (and therefore discrete). We do not promise a satisfactory proof that all this works in the case X a continuous rv: we simply observe that all our work is to be on a computer, with finite precision, and that all the probability densities we will consider are in fact represented by some closely approximating pmf on the computer.

You can find other presentations of parts of the material in this section in Section 5.5 of Norris and Chapter 10 of Ross.

8.1. **Markov chains.** Let $\{X_t\}_{t=0}^{\infty}$ be a homogeneous Markov chain of random variables on Ω with starting distribution $X_0 \sim p^{(0)}$ and transition probability

$$P_{i,j} = \Pr(X_{t+1} = j | X_t = i).$$

Denote by $P_{i,j}^{(n)}$ the n-step transition probabilities

$$P_{i,j}^{(n)} = \Pr(X_{t+n} = j | X_t = i)$$

and by $p^{(n)}(i) = \Pr(X_n = i)$. Recall that P is *irreducible* if and only if, for each pair of states $i, j \in \Omega$ there is n such that $P_{i,j}^{(n)} > 0$. The Markov chain is *aperiodic* if $P_{i,j}^{(n)}$ is non zero for all sufficiently large n.

Example 8.1. Here is an example of a period chain: $\Omega = \{1, 2, 3, 4\}, p^{(0)} = (1, 0, 0, 0)$, and transition matrix

$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{pmatrix},$$

since $P_{1,1}^{(n)} = 0$ for n odd.

Exercise: show that if P is irreducible and $P_{i,i} > 0$ for some $i \in \Omega$ then P is aperiodic.

Recall that the pmf $\pi(i), i \in \Omega$, $\sum_{i\in\Omega} \pi(i) = 1$ is a stationary distribution of P if $\pi P = \pi$. If $p^{(0)} = \pi$ then

$$p^{(1)}(j) = \sum_{i\in\Omega} p^{(0)}(i) P_{i,j},$$

so $p^{(1)}(j) = \pi(j)$ also. Iterating, $p^{(t)} = \pi$ for each $t = 1, 2, ...$ in the chain, so the distribution of $X_t \sim p^{(t)}$ doesn't change with t, it is stationary.

We now introduce reversible Markov chains (see also Norris 1.9). In a reversible chain we cannot distinguish the direction of simulation from inspection of a realization of the chain (so, you simulate a piece of the chain, toss a coin and reverse the order of states if the coin comes up heads; now you present me the sequence of states; I cannot tell whether or not you have reversed the sequence, though I know the transition matrix of the chain).

Denote by $P'_{i,j} = \Pr(X_{t-1} = j | X_t = i)$ the transition matrix for the time-reversed chain. It seems clear that a Markov chain will be reversible if and only if $P = P'$, so that any particular transition occurs with equal probability in forward and reverse directions.

Theorem 8.1. *(i) If there is a probability mass function $\pi(i), i \in \Omega$ satisfying $\pi(i) \geq 0$, $\sum_{i\in\Omega} \pi(i) = 1$ and*

$$\pi(i)P_{i,j} = \pi(j)P_{j,i} \qquad \text{for all pairs } i, j \in \Omega, \tag{8.1}$$

then $\pi = \pi P$ so π is stationary for P. (ii) If in addition $p^{(0)} = \pi$ then $P' = P$ and the chain is reversible with respect to π.

The relations in Eqn. 8.1 are called the *detailed balance* relations (and in this sense, $\pi = \pi P$ would be global balance).

Proof of (i): sum both sides of Eqn. 8.1 over $i \in \Omega$. Now $\sum_i P_{j,i} = 1$ so $\sum_i \pi(i)P_{i,j} = \pi(j)$ and we are done.

Proof of (ii), we have π a stationary distribution of P so $\Pr(X_t = i) = \pi(i)$ for all $t = 1, 2, ...$ along the chain. Then

$$\begin{aligned} P_{i,j} &= \Pr(X_{t+1} = j | X_t = i) \\ &= \Pr(X_t = i | X_{t+1} = j) \frac{\Pr(X_{t+1} = j)}{\Pr(X_t = i)} \\ &= P'_{j,i}\pi(j)/\pi(i) \end{aligned}$$

but $P_{i,j} = P_{j,i}\pi(j)/\pi(i)$ by detailed balance so $P = P'$.

Why bother with reversibility? It is (i) of Theorem 8.1 that will be useful to us. If we can find a transition matrix P satisfying $p(i)P_{i,j} = p(j)P_{j,i}$ then $pP = p$ so 'our' p is a stationary distribution. Given P it is far easier to verify detailed balance, than to check $p = pP$ directly.

We will be interested in using simulation of $\{X_t\}_{t=0}^{n-1}$ in order to estimate $E_p(f(X))$. The idea will be to arrange things so that the stationary distribution of the chain is $\pi = p$: if $X_0 \sim p$ (*ie* start the chain in its stationary distribution) then $X_t \sim p$ for all t and we get some useful samples.

The 'obvious' estimator is

$$\hat{f} = \frac{1}{n} \sum_{t=0}^{n-1} f(X_t),$$

but we may be concerned that we are averaging correlated quantities. Also, if we always start the Markov chain in some fixed state $X_0 = x^{(0)}$ say then we certainly dont have $X_0 \sim p$. In your study of Markov chains last term you saw just the two theorems we need. The convergence theorem (Theorem 1.8.3 page 41 of Norris) gives us conditions for $p^{(t)} \to \pi$, from any start, so if $\pi = p$ we will get $X_t \to p$ in distribution.

However, this mode of convergence is not enough for us. The problem is that our estimator $\hat{f}$ is getting 'polluted' by samples drawn in the early phase of the chain, when the distribution of X_t is still converging. The Ergodic theorem (Norris Section 1.10) covers this aspect.

Theorem 8.2. *If $\{X_t\}_{t=0}^{\infty}$ is an irreducible and aperiodic Markov chain on a finite space of states Ω, with stationary distribution p then, as $n \to \infty$, for any bounded*

function $f : \Omega \to R$,

$$\frac{1}{n}\sum_{t=0}^{n-1} f(X_t) \;\to\; E_p(f(X)).$$

The convergence is almost surely, which is stronger than, and implies, convergence in probability. In Part A probability the Ergodic theorem asks for positive recurrent $X_0, X_1, X_2, ...$, and the stated conditions are simpler here because we are assuming a finite state space for the Markov chain.

We would really like to have a CLT for $\hat{f}$ formed from the Markov chain output, so we have confidence intervals $\pm\sqrt{\mathrm{var}(\hat{f})}$ as well as the central point estimate $\hat{f}$ itself. It is in fact the case that under mild conditions, which hold for almost all the MCMC simulations I have done, there is a CLT for $\hat{f}$. However, these results are a little beyond us at this point.

So, we can form estimates of $E_p(f)$ by averaging along states of a Markov chain which has p as its equilibrium distribution. There remains the problem of how large n must be for the guaranteed convergence to give a usefully accurate estimate. This problem, of assessing when the Markov chain simulation length is sufficiently large, does not have a simple honest answer, though there are some obvious necessary conditions we can check (*eg* repeat the entire simulation and check that independent estimates $\hat{f}$ have an acceptably small scatter). We should check also that 'most' of the samples are not biased in any obvious way by the choice of X_0, the initial state.

8.2. **Metropolis Hastings Markov chain Monte Carlo.** The Metropolis Hastings Markov Chain Monte Carlo algorithm (or MCMC, for short) is an algorithm for simulating a Markov Chain with any given equilibrium distribution.

If we are given a pdf of pmf p then we may be able to simulate an iid sequence $X_1, X_2, ..., X_n$ of r.v. satisfying $n^{-1}\sum_i f(X_i) \to E_p(f(X))$ as $n \to \infty$, using the Rejection algorithm.

In a similar way, if we are given a pdf of pmf p then we may be able to simulate an correlated sequence $X_1, X_2, ..., X_n$ of r.v. (*ie*, a Markov chain) satisfying $n^{-1}\sum_i f(X_i) \to E_p(f(X))$ as $n \to \infty$, using the MCMC algorithm.

In each case convergence in probability is 'easily' established, whilst the more useful CLT 'usually' applies, but is harder to verify, at least in the MCMC case.

We will start with simulation of rv X on a finite state space. Let $p(x) = \tilde{p}(x)/Z_p$ be the pmf on finite state space $\Omega = \{1, 2, ..., m\}$. We will call p the (pmf of the) target distribution. This is the one we want to sample. Fix a 'proposal' transition matrix $q(y|x)$. We will use the notation $Y \sim q(\cdot|x)$ to mean $\Pr(Y = y|X = x) = q(y|x)$.

Theorem 8.3. *The transition matrix P of the Markov chain generated by the following Metropolis-Hastings MCMC algorithm satisfies $p = pP$.*

Algorithm 8.1. If $X_t = x$, then X_{t+1} is determined in the following way.

1. Let $Y \sim q(\cdot|x)$ and $U \sim U(0, 1)$. Simulate $Y = y$ and $U = u$.

2. If
$$u \leq \min\left\{1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right\}$$
set $X_{t+1} = y$, otherwise set $X_{t+1} = x$.

Proof: Look at the conditions in Theorem 8.1. Since p is a pmf, it is enough to check that the detailed balance relations, Eqn 8.1, are satisfied for all $x, y \in \Omega$. The case $x = y$ is trivial. Let
$$\alpha(y|x) = \min\left\{1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right\}.$$
If we enter a MCMC update with $X_t = x$, then the probability to come out with $X_{t+1} = y$ for $y \neq x$ is the probability to propose y at step 1 times the probability to accept it at step 2. The transition matrix $P_{x,y} = \Pr(X_{t+1} = y|X_t = x)$ for the Markov chain simulated by the algorithm is therefore
$$P_{x,y} = q(y|x)\alpha(y|x).$$
Assume without loss of generality that $\tilde{p}(x)q(y|x) \geq \tilde{p}(y)q(x|y)$ (x and y can change labels if this is false). Now,
$$\begin{aligned} p(x)P_{x,y} &= p(x)q(y|x)\,\alpha(y|x) \\ &= p(x)q(y|x) \times \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)} \\ &= p(y)q(x|y). \\ p(y)P_{y,x} &= p(y)q(x|y)\,\alpha(x|y) \\ &= p(y)q(x|y) \times 1. \end{aligned}$$
I used $p(x)\tilde{p}(y)/\tilde{p}(x) = p(y)$ between the 2nd and 3rd lines and $\tilde{p}(x)q(y|x) \geq \tilde{p}(y)q(x|y)$ to get the 2nd and final lines. Eqns 8.1 (and the conditions of Theorem 8.1) are thus satisfied, and so p is a stationary distribution of the Markov Chain simulated by algorithm 8.1.

Corollary 8.4. *If the Markov chain $X_0, X_1, X_2, ...$ simulated by Algorithm 8.1 is irreducible and aperiodic, then, for any bounded function $f : \Omega \to R$*
$$\frac{1}{n}\sum_{t=0}^{n-1} f(X_t) \;\to\; E_p(f(X)).$$
with convergence as for the Ergodic Theorem.

Proof: we have seen that p is stationary for $\{X_t\}_{t=0}^{\infty}$ so the conditions of Theorem 8.2 are satisfied if in addition the $\{X_t\}_{t=0}^{\infty}$ are irreducible and aperiodic.

In order to run this Markov chain simulation we need to specify a start state $X_0 = x_0$ and a proposal mechanism $q(y|x)$. We then repeat steps 1 and 2 of algorithm 8.1 to generate a sequence $X_0, X_1, X_2, ..., X_n$, and these are our correlated samples distributed according to p (at least for large n when $p^{(n)}$ has converged to p). Notice that when we come to compute the acceptance probability $\alpha(y|x)$, we do not need the normalized expression for the pmf's: α depends only on the ratio $p(y)/p(x) = \tilde{p}(y)/\tilde{p}(x)$.

We are left, by Corollary 8.4, to verify irreducibility and aperiodicity. The latter is usually straightforward, since the MCMC algorithm may reject the candidate state y, so that the transition matrix $P_{x,y}$ for the Markov chain satisfies $P_{x,x} > 0$ for at least some states $x \in \Omega$. In order to check irreducibility we need to check that the proposal mechanism q can take us anywhere in Ω (so q itself is an irreducible transition matrix), and then that the acceptance step doesn't trap the chain (as might happen if $\alpha(y|x)$ is zero too often).

Example 8.2. We will use MCMC to simulate $X \sim p$ with $p \propto (1, 2, ..., m)$. The normalising constant is $Z = \sum_i i = m(m+1)/2$, the unnormalised mass function $\tilde{p}(x) = x$ and $p(x) = \tilde{p}(x)/Z$. One simple proposal distribution is $Y \sim q$, $q = (1, 1, ..., 1)/m$ the uniform distribution on 1 to m, which we write $Y \sim U\{1, 2, ..., m\}$. This proposal scheme is clearly irreducible (we can get from A to B in a single hop). Here is a MCMC algorithm simulating $X_t \sim p$. Start with some arbitrary starting point, for example, $X_t = 1$.

Algorithm 8.2. If $X_t = x$ then X_{t+1} is determined in the following way.

- Simulate $Y \sim U\{1, 2, 3, ...\}$. Simulate $U \sim U(0, 1)$.
- If $U \leq \alpha(y|x)$ set $X_{t+1} = y$ and otherwise set $X_{t+1} = x$.

Compute α.

$$\begin{aligned} \alpha(y|x) &= \min\left(1, \frac{p(y)q(x|y)}{p(x)q(y|x)}\right) \\ &= \min\left(1, \frac{y \times 1/m}{x \times 1/m}\right) \\ &= \min\left(1, \frac{y}{x}\right). \end{aligned}$$

Does this work on the computer?

```
#MCMC simulate X_t according to p=[1:m]/sum(1:m).
m<-30
pt<-1:m

n<-10000
X<-rep(NA,n)
X[1]<-1
for (t in 1:(n-1)) {
   x<-X[t]
   y<-ceiling(m*runif(1))
   a<-min(1,pt[y]/pt[x])
   U<-runif(1)
   if (U<=a) {
      X[t+1]<-y
   } else {
      X[t+1]<-x
   }
}
plot(X[1:200],type="l",ann=F)
hist(X,-1:m+0.5,freq=F,ann=F); lines(1:m,pt/sum(pt),ann=F)
```

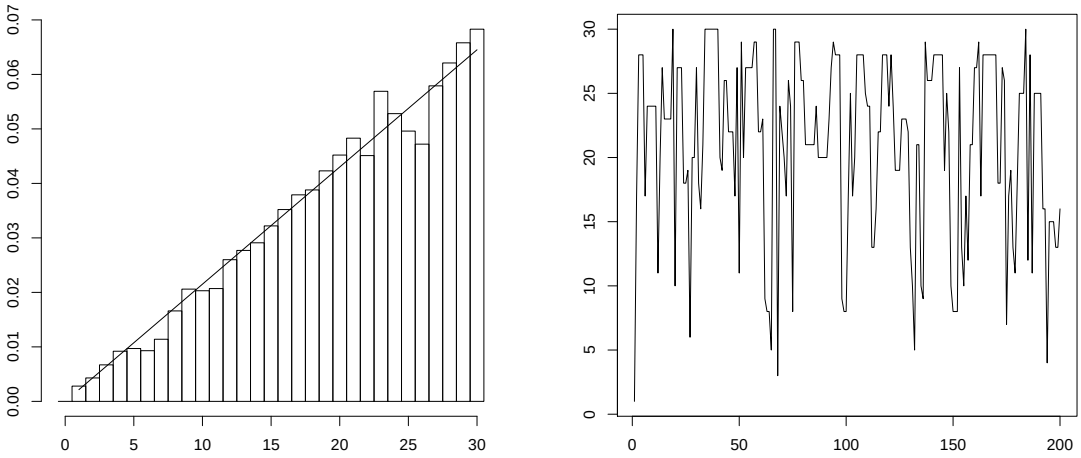


FIGURE 2. The figure at right shows the first 200 states visited by the MCMC algorithm. The figure at left is a histogram of the 10000 sampled states.

The code example gives MCMC simulation of a Markov chain with eqilibrium distribution $p = (1, 2, ..., 30)/465$. The output is plotted in Figure 8.2.

Example 8.3. We will use MCMC to simulate Poisson variates. Let $p(x) = \exp(-\lambda)\lambda^x/x!$ and $X \sim p$. We need a proposal mechanism which will take us around the space $\Omega = \{0, 1, 2, ...\}$ of X. Anything irreducible will do. I will use the simplest thing I can think of,

$$q(y|x) = \begin{cases} 1/2 & \text{if } y = x \pm 1, \\ 0 & \text{otherwise.} \end{cases}$$

Given x we generate y by tossing a coin and adding or subtracting one. Here is a MCMC algorithm simulating $X_t \sim$Poisson(λ). Start with some arbitrary starting point, for example, $X_t = 0$.

Algorithm 8.3. If $X_t = x$ then X_{t+1} is determined in the following way.

- Simulate $V \sim U(0, 1)$, and set $y = x+1$ if $V > 1/2$ and otherwise $y = x-1$. Simulate $U \sim U(0, 1)$.
- If $U \leq \alpha(y|x)$ set $X_{t+1} = y$ and otherwise set $X_{t+1} = x$.

Compute α. There are two main cases. If $y = x + 1$ then

$$\begin{aligned}\alpha(x+1|x) &= \min\left(1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right) \\ &= \min\left(1, \frac{\exp(-\lambda)\lambda^{x+1}/(x+1)!}{\exp(-\lambda)\lambda^{x}/x!} \times \frac{1/2}{1/2}\right) \\ &= \min\left(1, \frac{\lambda}{x+1}\right).\end{aligned}$$

If $y = x - 1$ then

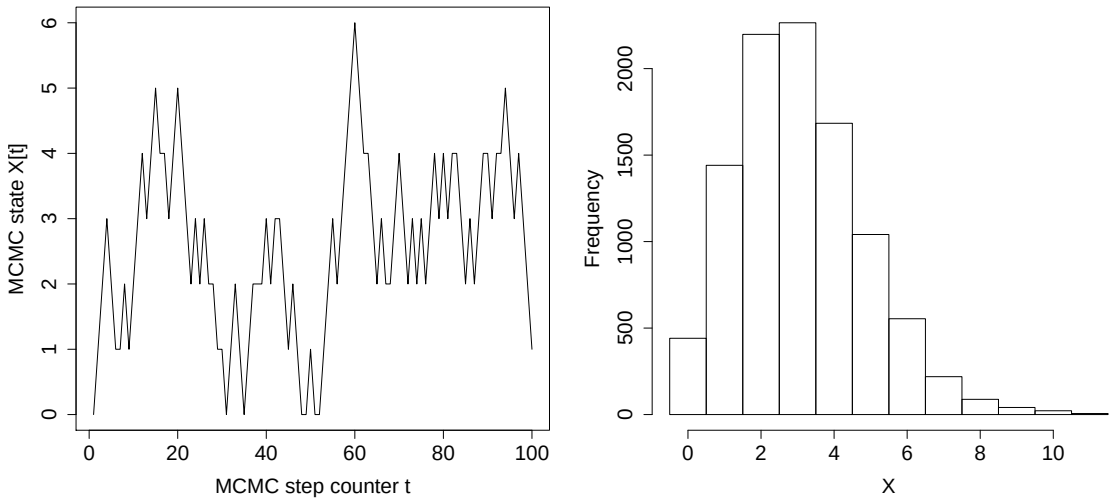
$$\alpha(x-1|x) = \min\left(1, \frac{x}{\lambda}\right).$$

We must always check the behavior at the boundary of Ω. What happens if we step outside the set of allowed states? This would happen if $x = 0$ and we propose $y = -1$. The simplest way to deal with this is to define $p(-1) = 0$ (or in general $p(y) = 0$ for $y \notin \Omega$). Then

$$\begin{aligned}\alpha(-1|0) &= \min\left(1, \frac{\tilde{p}(-1)q(0|-1)}{\tilde{p}(0)q(-1|0)}\right) \\ &= \min(1, 0) \\ &= 0,\end{aligned}$$

so if we propose to leave the space, the acceptance probability is zero, so we reject the candidate and stay where we are. Does this work on the computer?

```
#MCMC simulate X_t according to a Poisson dbn of mean lambda=3.
lambda<-3
n<-10000
X<-rep(NA,n)
X[1]<-0
for (t in 1:(n-1)) {
    x<-X[t]
    y<-x+2*(runif(1)<=0.5)-1
    if (y>x) {
        a<-lambda/(x+1)
    }
    if (y<x & y>=0) {
        a<-x/lambda
    }
    if (y<0) {
        a<-0
    }
    U<-runif(1)
    if (U<=a) {
        X[t+1]<-y
    }
    else {
        X[t+1]<-x
    }
}
> X[1:10] #first 10 states visited
```




```
 [1] 0 1 2 3 2 1 1 2 1 2
> mean(X) #expect = 3 (but no easy way to get a sd() as X[t] correlated)
[1] 3.0789
> var(X)  #expect = 3 (same comment)
[1] 3.134788
```

8.3. **MCMC for state spaces which are not finite.** What of using MCMC for a target distribution on a space which is discrete but not finite, or using MCMC for a random variable which is continuous, and therefore has a pdf rather than a pmf? The theory we have covered doesn't treat these cases. Our Poisson example had an infinite state space. Of course, if I had truncated the distribution at the largest integer I can represent exactly on the computer, there would be no detectable change to the samples: since the MCMC never visited that boundary, it doesn't know it was there. I get the same samples I would have had anyway. This kind of approximation is implicit in almost all numerical work.

The question remains, how to treat continuous random variables? It is easy to see that the condition for irreducibility must take some other form for Markov chains on continuous spaces (the probability of hitting any particular state will be zero). A rigorous treatment of these issues is beyond us here. However, in some respects such an analysis must be irrelevant. In numerical work on the computer, continuous and unbounded random variables are approximated by discrete analogues, on a finite space. The real number line is broken up into cells, with x in the cell δx say. Because the precision is fixed at around 15 decimal places, the length $|\delta x|$ of a cell depends on the x value, $|\delta x|/x \simeq 10^{-15}$ for $|x| \gg |\delta x|$. If $\pi(x)$ and $q(y|x)$ are densities then they are approximated on the computer by distributions $\Pr(X \in \delta x) \simeq \pi(x)|\delta x|$ and $\Pr(Y \in \delta y | X = x) \simeq q(y|x)|\delta y|$. This is still approximate, as the return values of the functions $\pi()$ and $q()$ are rounded, and we are approximating integrals over

the sets δx and δy, assuming the densities are constant to machine precision over these small sets. The Hastings ratio is then evaluated as

$$\begin{aligned}\frac{\Pr(Y \in \delta y)\Pr(X \in \delta x|Y=y)}{\Pr(X \in \delta x)\Pr(Y \in \delta y|X=x)} &\simeq \frac{p(y)|\delta y|q(x|y)|\delta x|}{p(x)|\delta x|q(y|x)|\delta y|} \\ &= \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}.\end{aligned}$$

If we apply the Metropolis Hastings ratio to densities, for continuous random variables, we expect to simulate the approximate distribution, in which the density is lumped into the rounding-sets δx. For this reason our discussion of Markov chains on finite spaces remains relevant.

Example 8.4. MCMC for a standard Normal distribution. Suppose want to simulate the standard normal distribution $X \sim N(0,1)$. The target density is

$$\tilde{p}(x) \propto \exp(-x^2/2).$$

I left off the factor $1/2\pi$, since it will cancel in the Hastings ratio. We will use a proposal density which allows us to tour $\mathbb{R}$. We have alot of freedom here, so this is simply something that 'will do'. Fix a constant $a > 0$ and choose a new point uniformly at random in a window of length $2a$ centred at x. The proposal density is

$$q(y|x) = \frac{1}{2a}\mathbb{I}_{x-a<y<x+a}$$

Now $q(y|x) = q(x|y)$: the probability density to propose y given x is equal to the probability density to propose x given y.

Here is a Metropolis Hastings Markov Chain Monte Carlo algorithm simulating $X_t \sim N(0,1)$.

Algorithm 8.4. Start with with an arbitrary point $X_0 = 0$ say. If $X_t = x$ then X_{t+1} is determined in the following way.

- Simulate $V \sim U(0,1)$ iid and set $Y = x + (2V-1)a$. Simulate $U \sim U(0,1)$.
- If $U \leq \alpha(y|x)$ set $X_{t+1} = y$ and otherwise set $X_{t+1} = x$. Here

$$\begin{aligned}\alpha(y|x) &= \min\left(1, \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right) \\ &= \min\left(1, \exp(-y^2/2 + x^2/2)\right).\end{aligned}$$

We dont have to worry about the process leaving the state space, since the state space is the whole of $\mathbb{R}$.

```
#MCMC simulate X~N(0,1)
n<-10000
X<-matrix(NA,1,n)
X[1]<-0
a<-3

for (t in 1:(n-1)) {
    x<-X[t]
    y<-x+(2*runif(1)-1)*a
    U<-runif(1)
```

```
    MHR<-exp( (x^2-y^2)/2 )
    alpha<-min(1,MHR)
    if (U<=alpha) {
        X[t+1]<-y
    }
    else {
        X[t+1]<-x
    }
}
> mean(X)

> var(X)
```

Note that I chose $a = 3$ to generate graphs (a) and (b) in Figure 8.5. Any positive value would be technically correct (irreducible in the discrete sense) but values much less than, or much greater than, the standard deviation of X (which is one) would be inefficient. In both cases the chain moves very slowly through the space. The ergodic theorem still applies, but convergence to equilibrium is slow. In the first case (illustrated with $a = 0.3$ in Figure 8.5(c)) the chain nearly always accepts proposals, but moves just a short distance. In the latter case ($a = 30$ in Figure 8.5(d)), the chain hardly ever accepts, so it stays in the same state for a very long time.

Example 8.5. Here is an example of MCMC for a distribution with more than one variable: MCMC for a bivariate Normal distribution. Suppose want to simulate a bivariate normal distribution $X \sim N(\mu, \Sigma)$, $\mu = (1, 1)^T$ and $\Sigma_{ii} = \text{var}(X_i) = 3$, $\Sigma_{1,2} = \text{cov}(X_1, X_2) = -2$. The target density is

$$\tilde{p}(x_1, x_2) \propto \exp\left(-\frac{1}{2}\left[(x_1, x_2) - (1, 1)\right]\begin{pmatrix} 3 & -2 \\ -2 & 3 \end{pmatrix}^{-1}\left[\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right]\right)$$

where I have omitted the factor $1/(2\pi\sqrt{\det(\Sigma)})$, since it will cancel in the Hastings ratio. This time we need to tour $\mathbb{R}^2$. Inspired by the last example, fix a constant $a > 0$ and make random jumps of size up to a along the two axes:

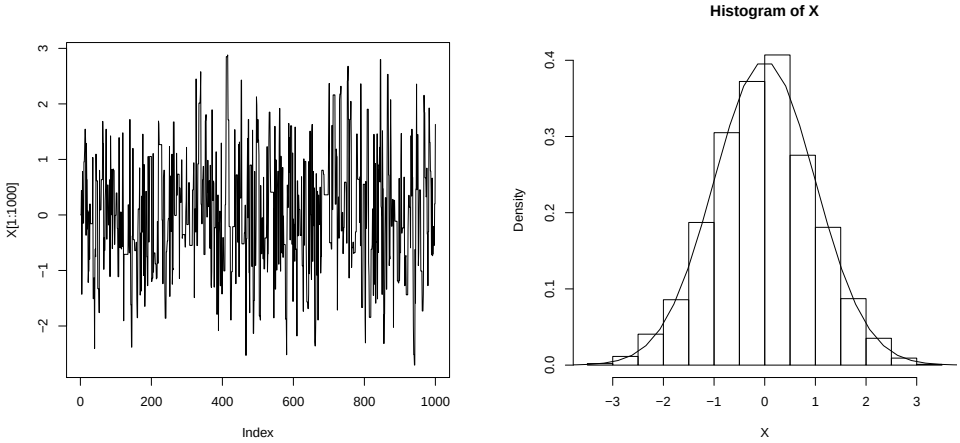
$$q(y|x) = \frac{1}{4a^2}\mathbb{I}_{x_1-a\le y_1\le x_1+a, x_2-a\le y_2\le x_2+a}$$

ie the uniform density in a box of side $2a$ centred at x with sides aligned to the axes. Now $q(y|x) = q(x|y)$: the probability density to propose y given x is equal to the probability density to propose x given y (if y is in reach of x then x is in reach of y so both densities equal $1/4a^2$).

Here is a Metropolis Hastings Markov Chain Monte Carlo algorithm simulating $X_t \sim N(\mu, \Sigma)$.

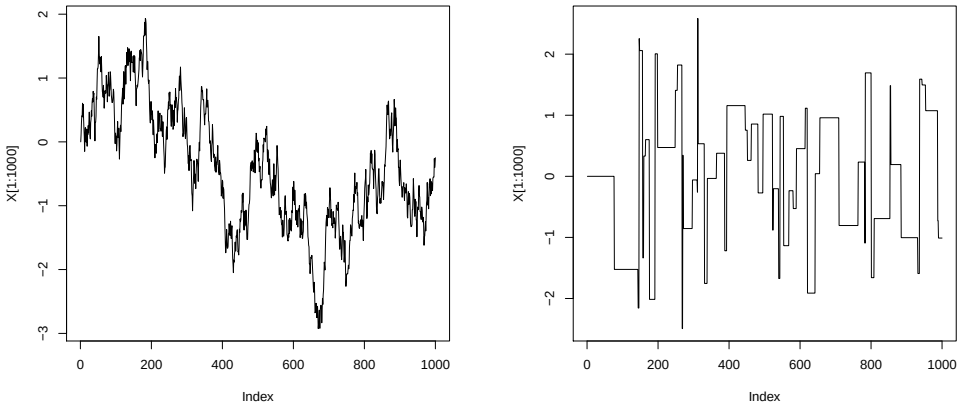
Algorithm 8.5. Start with with an arbitrary point $X_0 = (0, 0)^T$ say in $\mathbb{R}^2$. If $X_t = x$ then X_{t+1} is determined in the following way.

- Simulate $V_1, V_2 \sim U(0, 1)$ iid and set $Y_1 = x_1 + (2V_1 - 1)a$ and $Y_2 = x_2 + (2V_2 - 1)a$. Simulate $U \sim U(0, 1)$.



(a) The first 1000 steps, $a = 3$.

(b) A histogram of the samples.



(c) as (a) but $a = 0.3$.

(d) as (a) but $a = 30$.

FIGURE 3. MCMC for the standard normal distribution, Example 8.5.

- If $U \leq \alpha(y|x)$ set $X_{t+1} = y$ and otherwise set $X_{t+1} = x$. Here

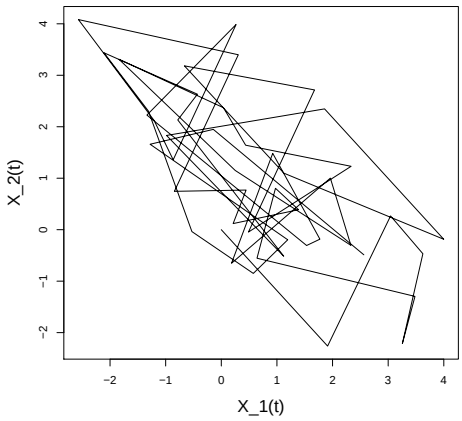
$$\begin{aligned}\alpha(y|x) &= \min\left(1,\ \frac{\tilde{p}(y)q(x|y)}{\tilde{p}(x)q(y|x)}\right)\\ &= \min\left(1, \exp\left(-(y-\mu)^T\Sigma^{-1}(y-\mu)/2 + (x-\mu)^T\Sigma^{-1}(x-\mu)/2\right)\right).\end{aligned}$$

Again, the process cant leave the state space, as the state space is all of $\mathbb{R}^2$.

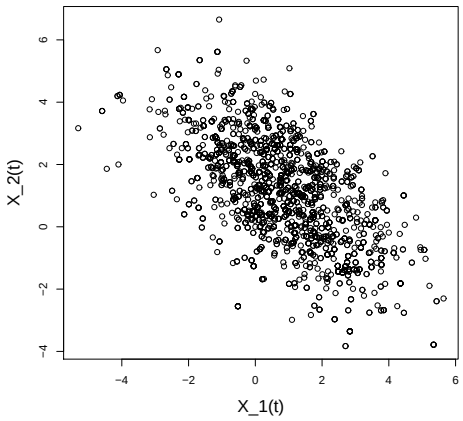
```
#MCMC simulate bivariate Normal
mu<-c(1,1)
(S<-matrix(c(3,-2,-2,3),2,2))
(Si<-solve(S))
```

```
n<-10000
X<-matrix(c(NA,NA),2,n)
X[,1]<-c(0,0)
a<-3

for (t in 1:(n-1)) {
    x<-X[,t]
    y<-x+(2*runif(2)-1)*a
    U<-runif(1)
    MHR<-exp( -t(y-mu)%*%Si%*%(y-mu)/2+t(x-mu)%*%Si%*%(x-mu)/2 )
    alpha<-min(1,MHR)
    if (U<=alpha) {
        X[,t+1]<-y
    }
    else {
        X[,t+1]<-x
    }
}
> apply(X,1,mean)
[1] 1.037171 0.963198
> cov(t(X))
          [,1]      [,2]
[1,]  3.064267 -2.006450
[2,] -2.006450  2.971347
```



The first 100 steps



The first 2000 points visited by the MCMC

Note that I chose $a = 3$ for similar reasons to the choice in Example 8.5.

8.4. **MCMC and conditional distributions.** How about simulating conditional distributions? This is a real strength of MCMC. The condition density $p(x|X \in B)$

for some set B a subset of the state space of X is $p(x|X \in B) = p(x)/\Pr(X \in B)$ for $x \in B$ and 0 otherwise. It follows that $\tilde{p}(x|X \in B) = \tilde{p}(x)$ for $X \in B$ and $\tilde{p}(x|X \in B) = 0$ for $x \notin B$. In other words, the MH ratio of the chain simulating the conditional distribution has just the same MH ratio as the unconditioned chain, for x, y both in B, and if y falls outside B we simply reject it. We should check that rejecting candidates in this way does not cost us irreducibility within B, but otherwise, things are straightforward.

Example 8.6. Suppose want to simulate the bivariate normal distribution $X \sim N(\mu, \Sigma)$ of Example 8.4 but conditioned on $|X-(3,3)| < 1$. The MCMC algorithms is

Algorithm 8.6. Start with $X_0 = (3,3)^T$ so the start state satisfies $|X-(3,3)| < 1$. If $X_t = x$ then X_{t+1} is determined in the following way.

- Simulate $V_1, V_2 \sim U(0,1)$ iid and set $Y_1 = x_1 + (2V_1 - 1)a$ and $Y_2 = x_2 + (2V_2 - 1)a$. Simulate $U \sim U(0,1)$.
- If $|Y-(3,3)| < 1$ and $U \leq \alpha(y|x)$ set $X_{t+1} = y$ and otherwise set $X_{t+1} = x$. Here
$$\alpha(y|x) = \min\left(1,\ \frac{\exp(-(y-\mu)^T\Sigma^{-1}(y-\mu)/2)}{\exp(-(x-\mu)^T\Sigma^{-1}(x-\mu)/2)}\right),$$
is unchanged from Example 8.4. We could have put $\tilde{p}(y)\mathbb{I}_{|y-(3,3)|<1}$ in the numerator of the acceptance probability itself. However, what we have done amounts to the same thing.

The code is the same as before but with a different initialization (X_0 inside the circle) and a very slightly altered test condition:

```
.
.
.
X[,1]<-c(3,3) #X_0 at center of allowed region
a<-1          #reduce proposal size - less fall outside circle
.
.
.
for (t in 1:(n-1)) {
    x<-X[,t]
    y<-x+(2*runif(2)-1)*a
    U<-runif(1)
    MHR<-exp( -t(y-mu)%*%Si%*%(y-mu)/2+t(x-mu)%*%Si%*%(x-mu)/2 )
    alpha<-min(1,MHR)
    if (U<=alpha & sum((y-c(3,3))^2)<1 ) {
        X[,t+1]<-y
    }
    else {
        X[,t+1]<-x
    }
}
```

The samples are constrained to lie within 1 unit of $(3, 3)$, but are otherwise distributed according to the bivariate normal. The results are illustrated in Fig 8.6. Exercise: How would use samples distributed $X \sim N(\mu, \Sigma)$ in order to simulate

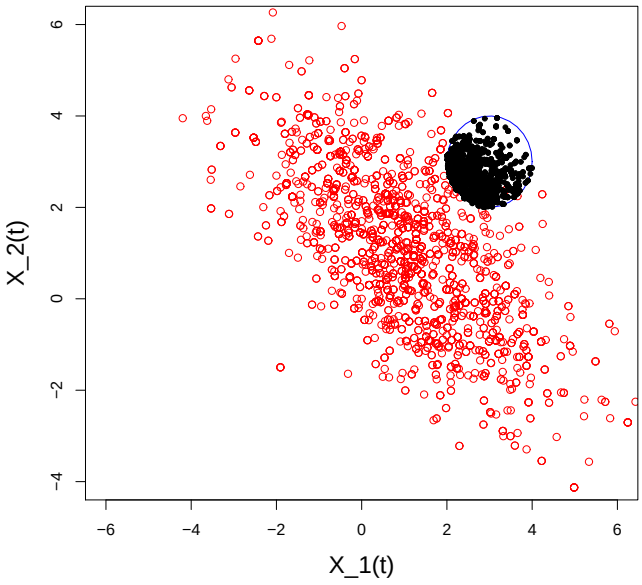


FIGURE 4. Open circles: $X \sim N(\mu, \Sigma)$, Dots: $X \,|\, (X - (3, 3))^2 < 1$.

$X \,|\, (X - (3, 3))^2 < 1$ via rejection? What weakness does this approach have?

STATISTICS DEPARTMENT

E-mail address: nicholls@stats.ox.ac.uk