Simulations Using Markov Chain Monte Carlo Methods

Giang Nguyen

February 12, 2016

Abstract

Markov chain Monte Carlo is a powerful method to estimate parameters that involve a mathematically intractable normalization constant. We discuss the Metropolis-Hastings algorithm in problems with one unknown and two unknown parameters. In the one-parameter case, we discuss the Beta-Binomial model. Due to conjugate prior distribution, the actual posterior distribution is known, providing an opportunity to compare our sampler output to the actual posterior distribution. We illustrate the two-parameter case with an example of the normal distribution with unknown mean and variance. In the last section, we carry out simulations of Straussian spatial point patterns, conditioning on the number of points in the pattern.

Contents

List of Figures

1 Introduction to the Metropolis-Hastings Algorithm

In this section, we give a brief overview of Markov chain Monte Carlo (MCMC) and the Metropolis-Hastings algorithm from a Bayesian point of view.

1.1 The Problem

Let D denote the observed data, and θ denote model parameters, which are considered random in the Bayesian paradigm. Let $f(\theta)$ denote the prior distribution, and $f(D|\theta)$ denote the likelihood. The posterior distribution of θ is

$$f(\theta|D) = \frac{f(\theta)f(D|\theta)}{\int f(\theta)f(D|\theta)d\theta}.$$

We commonly seek to find the posterior expectation of a function $g(\theta)$

$$E[g(\theta)|D] = \int g(\theta)f(\theta|D)d\theta = \frac{\int g(\theta)f(\theta)f(D|\theta)d\theta}{\int f(\theta)f(D|\theta)d\theta}. \tag{1}$$

The integrations in (1) can cause difficulties, especially if θ is of high dimension. MCMC is a technique to approximate these integrations using simulation.

1.2 Markov chain Monte Carlo

In order to estimate the expectation in (1), we can draw independent samples from the posterior distribution $f(\theta|D)$, say $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(T)}$. Then, as a result of the strong law of large numbers, the population mean can be approximated by

$$\hat{E}[g(\theta)|D] = \frac{1}{T}\sum_{i=1}^{T} g(\theta^{(i)}). \tag{2}$$

However, drawing independent samples from the posterior distribution $f(\theta|D)$ is not usually feasible since it is typically not possible to sample directly from the posterior. This problem is overcome by constructing a Markov chain which has the posterior distribution as its stationary (or limiting) distribution. This technique is called MCMC.

1.3 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a method to construct a Markov chain as described in the previous subsection. The algorithm can be described as follows:

1. Initialize $\theta^{(i)}$ where $i = 0$.
2. Generate a candidate value θ' from some proposal distribution $q(\cdot|\theta^{(i)})$.
3. Accept θ' (i.e. let $\theta^{(i+1)} = \theta'$) with probability

$$\alpha(\theta^{(i)}, \theta') = min\left[1, \frac{f(D|\theta')f(\theta')q(\theta^{(i)}|\theta')}{f(D|\theta^{(i)})f(\theta^{(i)})q(\theta'|\theta^{(i)})}\right].$$

   Otherwise reject θ' (i.e. let $\theta^{(i+1)} = \theta^{(i)}$).
4. Increment i and proceed to step 2. Repeat T times.

After an approximate burn-in period, the $\theta^{(1)}, \ldots, \theta^{(T)}$ can be treated as a correlated sample from the posterior $f(\theta|D)$. We can use $\theta^{(1)}, \ldots, \theta^{(T)}$ in (2) to estimate (1).

2 The Beta-Binomial Model

Suppose $Y|\theta \sim Bin(n, \theta)$, where n is specified, and θ is unknown, $0 < \theta < 1$. It is often of interest to estimate θ from the results of a sequence of Bernoulli trials (Gelman et al., 1995). In this section, we use MCMC in the context of Bayesian inference to make the desired estimation.

2.1 Bayesian Inference for the Beta-Binomial Model

In the Beta-Binomial model, the prior distribution for θ is $Beta(\alpha, \beta)$, since the beta density is conjugate with respect to a binomial likelihood. Conjugacy ensures that the posterior distribution follows the same parametric form as the prior distribution. As a result, the posterior distribution is mathematically tractable, and we can easily perform inference for θ (Gelman et al., 1995). The posterior density is derived as follows. We have the prior density:

$$f(\theta) = \frac{1}{B(\alpha, \beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}.$$

Likelihood:

$$f(y|\theta) = \binom{n}{y}\theta^y(1-\theta)^{n-y}.$$

Posterior density:

$$\begin{aligned} f(\theta|y) &= \frac{f(y|\theta)f(\theta)}{f(y)} \\ &\propto f(y|\theta)f(\theta) \\ &\propto \binom{n}{y}\theta^y(1-\theta)^{n-y}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &\propto \theta^{y+\alpha-1}(1-\theta)^{n-y+\beta-1}, \end{aligned}$$

which is the kernel of a beta density, thus $\theta|y \sim Beta(y+\alpha, n-y+\beta)$.

2.2 MCMC for the Beta-Binomial Model

We illustrate MCMC in the context of the Beta-Binomial model with the following example.

Suppose we are interested in estimating the proportion of Californians who support the death penalty, denoted by θ. For illustration suppose the prior distribution for θ is

$$\theta \sim Beta(1, \frac{2}{3}).$$

A random sample of 100 Californians is taken, and 65% support the death penalty (Gelman et al., 1995). We use MCMC in the context of Bayesian inference to do posterior inference for θ. Since $n = 100$ and $y = 65$, by the result derived above, the posterior distribution:

$$\theta|y \sim Beta(66, 35.67).$$

The posterior mean is $\mathrm{E}(\theta|y) = 0.6492$, and the posterior variance is $\mathrm{Var}(\theta|y) = 0.00222$.

Since the beta density is conjugate with respect to a binomial likelihood, the posterior density has a closed form, and we easily obtained the posterior mean and variance. MCMC provides a method to perform inference for θ, even in complicated problems that lack conjugate priors. In the rest of the section, we use the Metropolis-Hastings algorithm to approximate the posterior mean and variance in this example.

We apply the Metropolis-Hastings algorithm using two proposal distributions: a) $Beta(3, 2)$, and b) uniform random walk with maximum step size of 0.1. We choose 0.5 as the starting value for θ. After $100,000$ iterations (and burn-in of $1,000$ iterations), the estimated means and variances from these proposal distributions are summarized in the following table.

Table 1: Summary of the Estimated Means and Variances

Proposal	Estimated Mean	Estimated Variance	95% Credible Set	Acceptance Rate
Beta(3,2)	0.6490	0.00221	(0.55, 0.74)	0.27
Random walk (step 0.1)	0.6493	0.00223	(0.55, 0.74)	0.62

The proposals $Beta(3, 2)$ and uniform random walk ± 0.1 result in estimates that are close to the actual mean and variance. We investigate these proposals by plotting the mixing behaviors of the Markov chain and comparing the prior, actual posterior and estimated posterior distributions.

Using the proposals $Beta(3, 2)$ and uniform random walk ± 0.1, convergence is quick, and the estimated posterior distributions are almost identical to the actual posterior distribution (Figure 1 and Figure 2).

A question that often needs addressing in implementing MCMC algorithms is: How many iterations are needed for the burn-in period? A popular approach is applying convergence diagnostic tools to the MCMC output. In this paper, we choose the convergence diagnostic method proposed by Geweke (1992). (For an in-depth review of MCMC convergence diagnostics, see Cowles and Carlin (1996).) Geweke's method is generally applicable to a single chain. The method is especially applicable if the researcher is interested in estimating the mean of some function g of the simulated parameters θ. In order to assess convergence, the sample mean from an early segment of the chain is compared to the sample mean from a later segment. Geweke proposed using the first 10% and the last 50% of the chain.

We use the R package boa to implement Geweke's diagnostic. Values returned include a Z-Score for the test of equality between the sample mean from the early

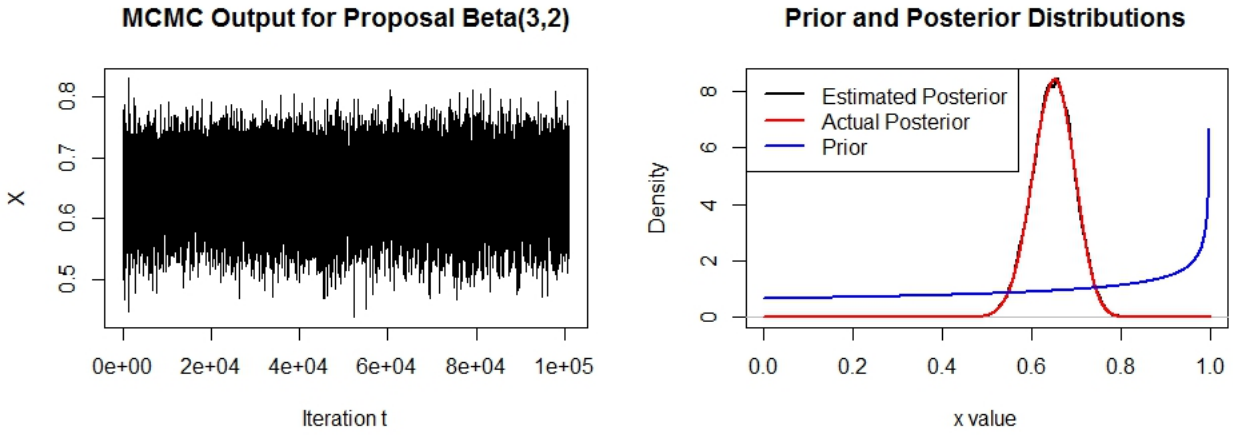

Figure 1: 100,000 Iterations with Beta(3,2) Proposal

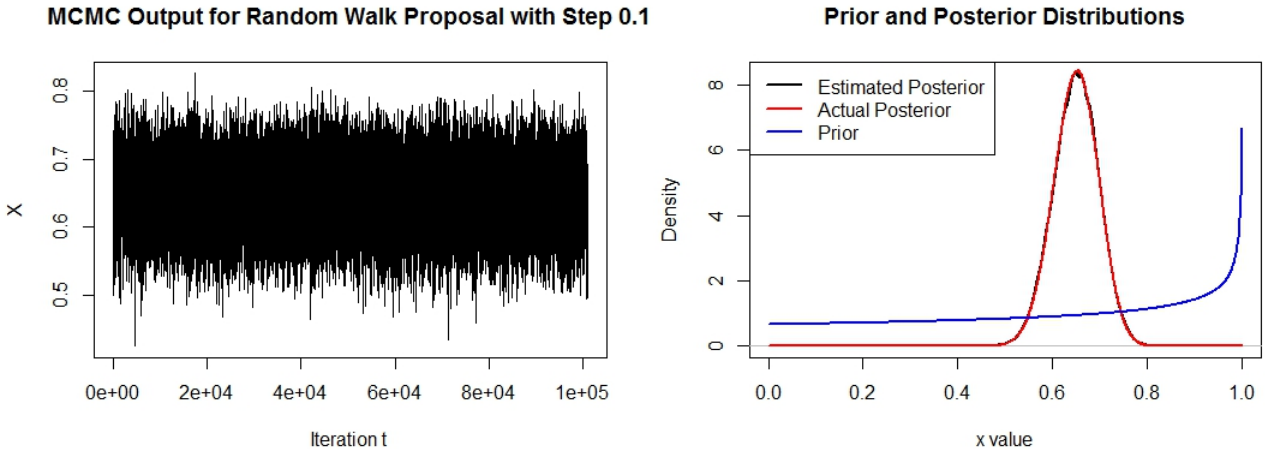

Figure 2: 100,000 Iterations with Uniform Random Walk ±0.1 Proposal

segment and the sample mean from the later segment of the chain. A frequentist two-sided p-value is also computed for this Z-Score, against the null hypothesis that the two sequences are from the same stationary distribution (Smith, 2007). We compare the sample mean of the first 10% and the sample mean of the last 50% of the chain, as suggested. The result is given below.

Table 2: Result of Geweke's Convergence Diagnostic

Proposal	Z-Score	p-value
Beta(3,2)	0.09	0.93
Random walk (step 0.1)	0.28	0.78

If the significance level is at 5%, the p-values suggest that the results are not significant. Therefore, there is no evidence of non-convergence, using Geweke's diagnostic.

3 Two-Parameter Case: An Example with the Normal Distribution

In this section, we extend our discussion of MCMC to the case when we wish to estimate two unknown parameters. Suppose

$$Y_i|\mu,\sigma \sim N(\mu,\sigma), i = 1, 2, ..., n,$$

where Y_i's are independent. Suppose further that the prior distributions are

$$\mu \sim N(0, 100),$$

and

$$\sigma \sim \chi^2(2),$$

where μ and σ are independent. As we are using non-conjugate prior distributions, the actual posterior distribution is unknown.

We simulate 10 observations from the $N(5, 2)$ distribution and use these observations to estimate the posterior means of μ and σ. As the data are sampled from a

$N(5, 2)$ distribution, and because the priors are not terribly informative, we expect that the estimated posterior means of μ and σ are (roughly speaking) close to 5 and 2, respectively.

We apply the Metropolis-Hastings algorithm with initial values $\mu = 0$ and $\sigma = 1$, $10,000$ iterations and $2,000$ iterations of burn-in. In each iteration, we first choose either μ or σ at random, then draw from a proposal distribution for the chosen parameter. The value of the other parameter (which was not chosen) stays the same for that iteration. This is sometimes called a "random scan" algorithm. We use two different combinations of proposals for μ and σ: a) uniform random walk ± 5 (for μ) and uniform random walk ± 1 (for σ), and b) $N(\mu^{(i)}, 1)$ (for μ) and uniform random walk ± 1 (for σ). The estimated posterior means are summarized in the following table.

Table 3: Summary of the Estimated Posterior Means

Proposal for μ	RW ± 5	$N(\mu^{(i)}, 1)$
Proposal for σ	RW± 1	RW ± 1
Estimated E($\mu|y$)	3.9097	3.9380
Estimated E($\sigma|y$)	2.1841	2.1749
95% credible set for μ	(2.44, 5.39)	(2.51, 5.36)
95% credible set for σ	(1.41, 3.44)	(1.42, 3.51)
Acceptance rate for μ	0.22	0.58
Acceptance rate for σ	0.59	0.59
Geweke's convergence diagnostic		
Z-Score (p-value) for μ	-0.03 (0.98)	0.46 (0.65)
Z-Score (p-value) for σ	-0.56 (0.58)	1.30 (0.19)

Both combinations of proposals lead to rapid convergence and mixing (Figure 3 and Figure 4). At the 5% significance level, Geweke's convergence diagnostic does not provide evidence of non-convergence for either set of proposals. Not surprisingly, therefore, posterior inference for μ and σ are similar for both proposals. This highlights the generality of the Metropolis-Hastings algorithm.

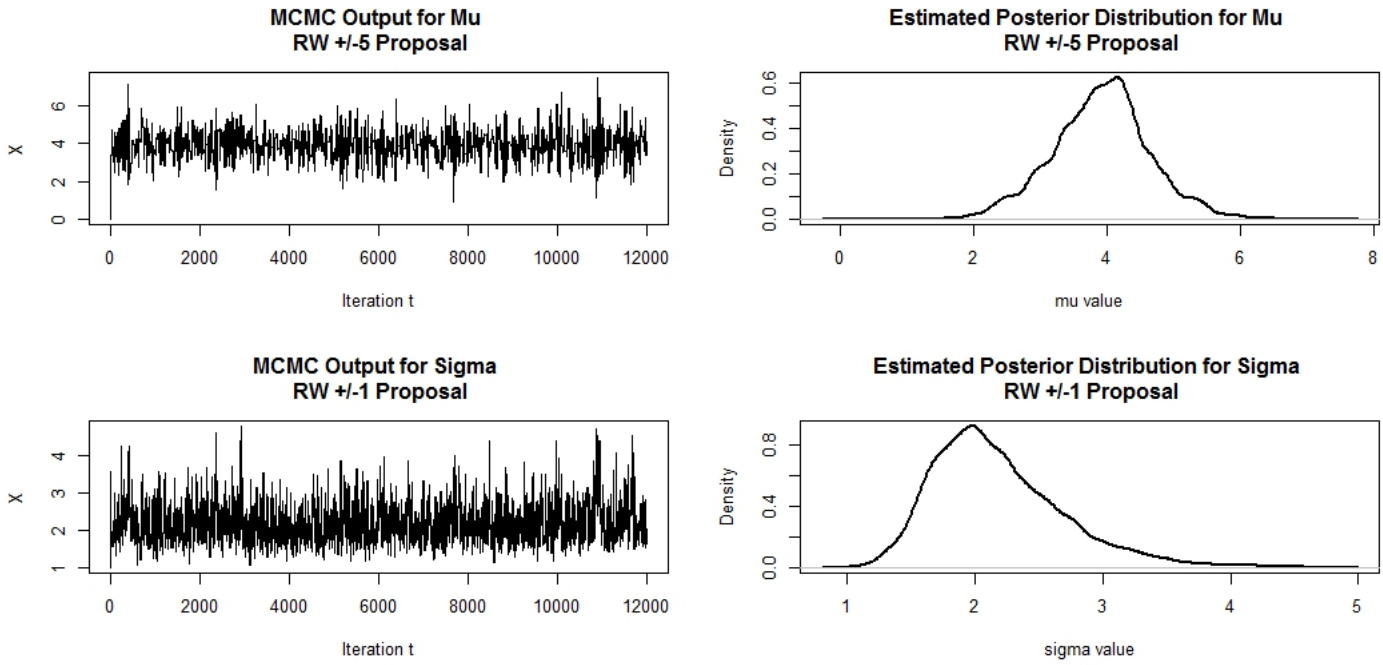

Figure 3: 10,000 Iterations with RW ± 5 and RW ± 1 Proposals

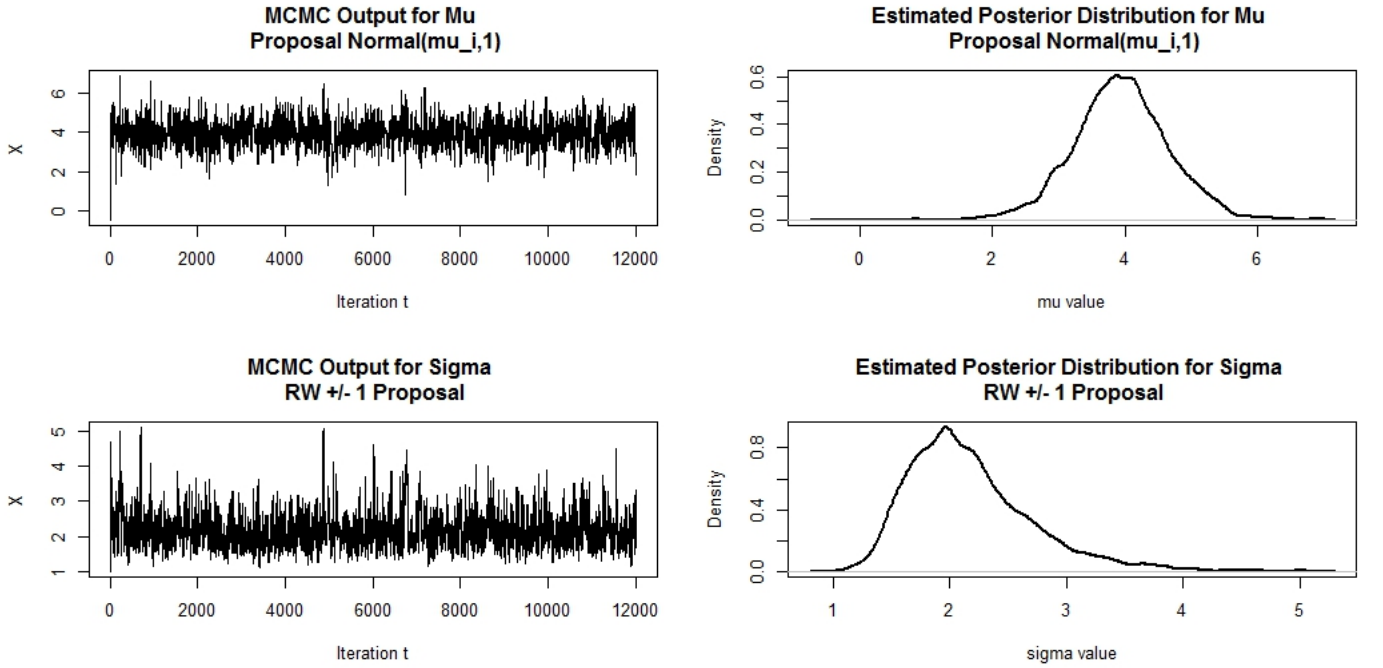

Figure 4: 10,000 Iterations with $\mathrm{N}(\mu^{(i)}, 1)$ and RW ± 1 Proposals

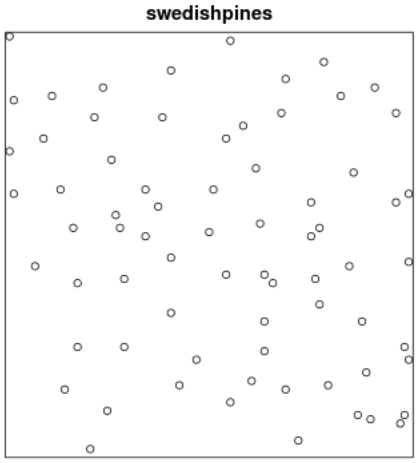

Figure 5: Positions of 71 Trees in a 9.6x10.0 Meter Forest

4 Pairwise Interacting Point Processes

4.1 An Introduction to Pairwise Interacting Point Processes

A spatial point pattern (SPP) dataset depicts the spatial locations of events occurring in a region of interest. For example, the events could represent ant nests, trees, or disease outbreaks. Events are visualized as dots in a bounded region. Figure 5 is a visualization of the dataset "Swedish Pines", a standard point pattern dataset installed in the package spatstat in R. The points represent the positions of 71 trees in a forest plot 9.6 by 10.0 metres.

The spatial positions of events in spatial point pattern datasets can sometimes exhibit regularity (inhibition), i.e., the points are more evenly spaced than complete spatial randomness. Our goal is to model the spatial inhibition in such datasets.

One class of models for spatial point patterns in a bounded region is pairwise interacting point processes (PIPPs). In a PIPP, the interaction between two points is a function of their interpoint distance, and is described by a *pair potential function.*

Suppose a point pattern $\mathbf{x} = \{x_i, i = 1, \ldots, n\}$ is observed in some bounded region V. A common pair potential function is due to Strauss (1975)

$$\phi_\theta(s) = \begin{cases} h & \text{if } s \leq b \\ 0 & \text{if } s > b \end{cases}$$

where $\theta = (b, h)$ and s is the interpoint Euclidean distance between two points.

Conditioning on n, the likelihood is

$$f(\mathbf{x}|\theta) = \frac{\exp[-\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\phi_\theta(\|x_i - x_j\|)]}{Z_n(\theta)}$$

where $\|.\|$ denotes Euclidean distance and

$$Z_n(\theta) = \int_{V^n} \exp\Big[-\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\phi_\theta(\|x_i - x_j\|)\Big] dx_1 \ldots dx_n$$

is an intractable function of θ (Bognar and Cowles, 2004).

In the literature, b is called the *interaction distance*, and h is known as the *Straussian* parameter. We observe that for positive values of the Straussian parameter, it is unlikely to have points that are close together. Hence, the Straussian parameter signals the penalty for close distance, while the interaction distance is the value at which this penalty ceases to take place.

4.2 Using the Metropolis-Hastings Algorithm in a Pairwise Interacting Point Process

Suppose we have n points in a bounded region $W \subset \mathbb{R}^2$. Conditioning on n, we can simulate a PIPP as follows:

1. Generate n points uniformly in W, say $x_1, \ldots, x_n$. Let $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$, where $i = 0$.

2. Randomly choose a point out of the n points, say, x_k. We will update the coordinates of x_k.

3. Generate a candidate point, say x_k' uniformly in W.

Let $\mathbf{x}' = (x_1^{(i)}, \ldots, x_{k-1}^{(i)}, x_k', x_{k+1}^{(i)}, \ldots, x_n^{(i)})$.

4. Accept $\mathbf{x}'$ (i.e. let $\mathbf{x}^{(i+1)} = \mathbf{x}'$) with probability

$$\alpha = min\left[1, \frac{f(\mathbf{x}'|\theta)q(\mathbf{x}^{(i)}|\mathbf{x}')}{f(\mathbf{x}^{(i)}|\theta)q(\mathbf{x}'|\mathbf{x}^{(i)})}\right]. \tag{3}$$

   Otherwise reject $\mathbf{x}'$ (i.e. let $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)}$).

5. Increment i and proceed to step 2. Repeat a large number of times, say, T.

   Note that the intractable normalizer $Z_n(\theta)$ cancels in (3).

In the rest of the section, we describe our simulations of spatial point patterns, conditioning on the number of points in the pattern.

In the initial SPP, 50 points were generated uniformly in the unit square $W = [0,1]$x$[0,1]$. Next, choosing $h = 2$ and $b = 0.1$, we carried out 2000 iterations to simulate a Straussian SPP (Figure 6). We observe that in the Straussian SPP, there are fewer pairs of points that are close together than in the original pattern (complete spatial randomness). In other words, the Straussian SPP exhibits spatial inhibition.

Next, we simulated two other Straussian SPPs, keeping $h = 2$ while decreasing b to 0.05 and 0.01 (Figure 7). The inhibition is the same, but the interaction distance b is smaller. When $b = 0.01$, the interaction distance has such a small value that the SPP looks not unlike complete spatial randomness.

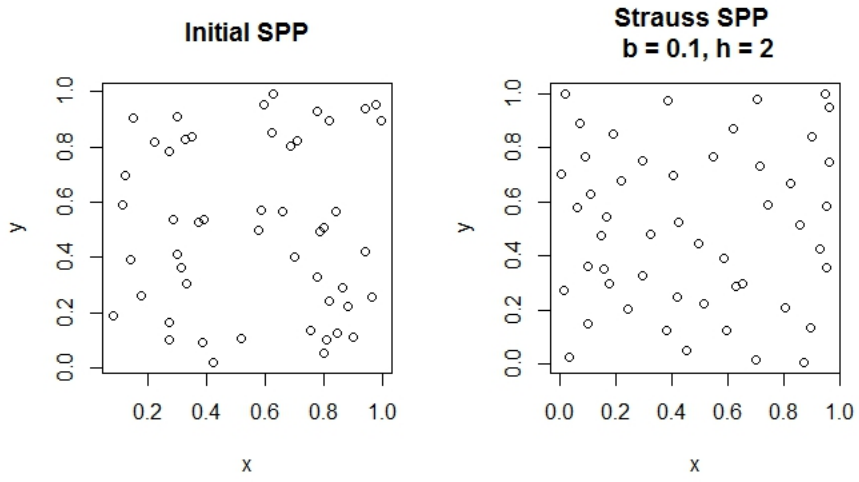

Figure 6: Initial and Straussian SPPs with $h = 2$ and $b = 0.1$

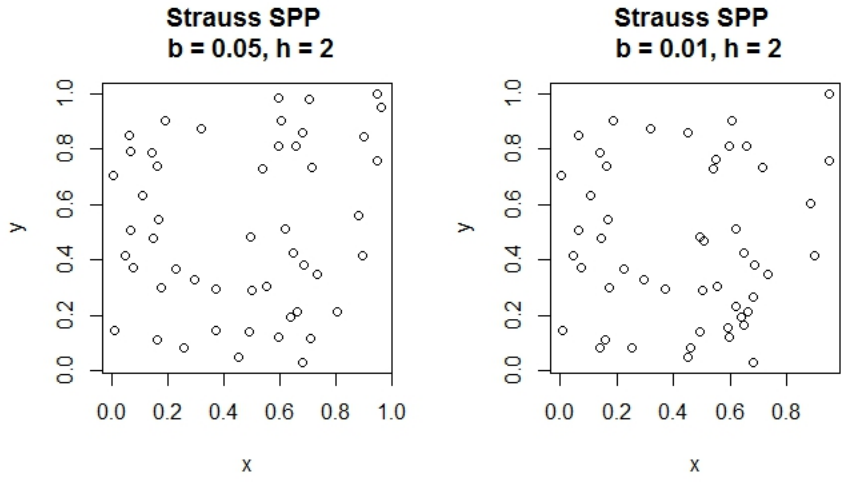

Figure 7: Straussian SPPs with $b = 0.05$, $b = 0.01$ and $h = 2$

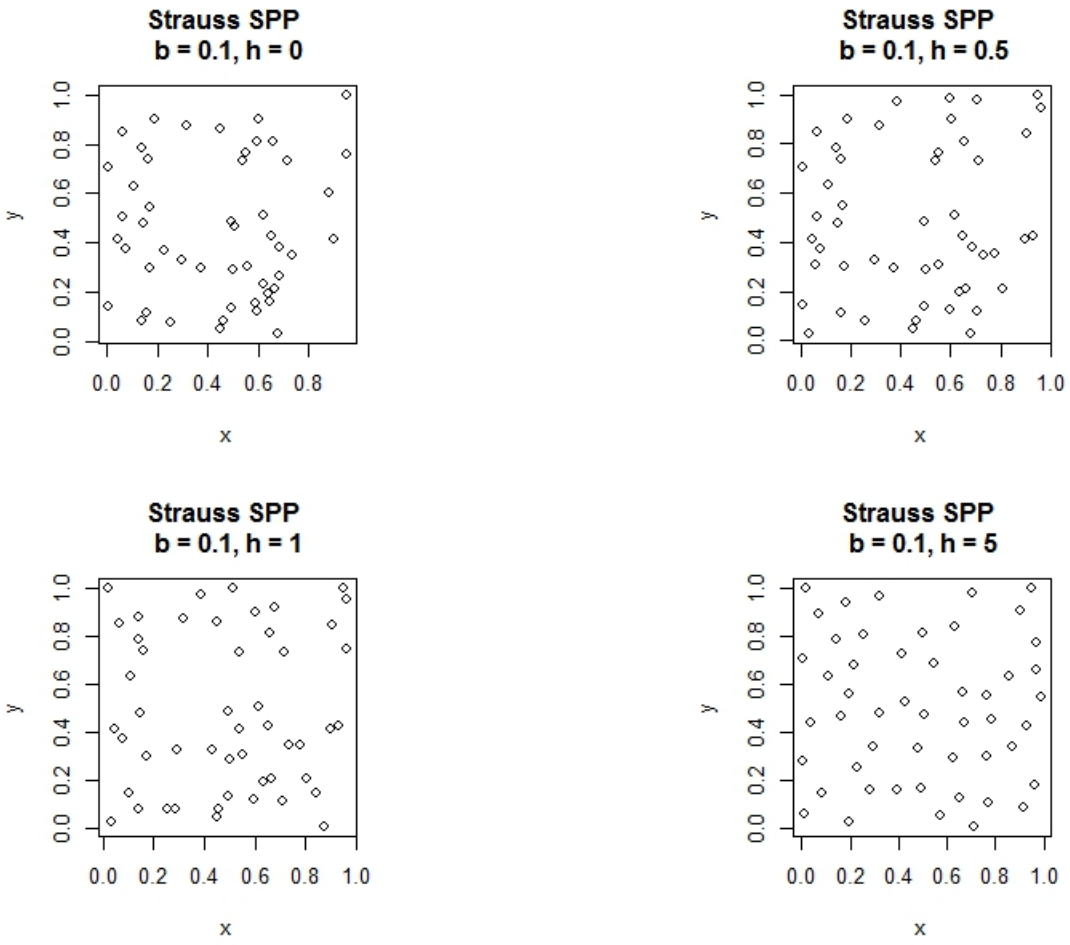

Figure 8: Straussian SPPs with $b = 0.1$, $h = 0$, $h = 0.5$, $h = 1$, and $h = 5$

Finally, we held $b = 0.1$ and simulated Straussian SPPs with different values of h (Figure 8). When $h = 0$, there is no penalty for closeness between two points. Therefore, we observe many pairs of points close together. The case in which $h = 0$ is called a *binomial process*. In addition, the SPP realized by a binomial process (in Figure 8) exhibits a pattern that looks similar to the SPP simulated with $b = 0.01$ and $h = 2$ (Figure 7). As h increases, the penalty for closeness also increases. At $h = 5$, the SPP displays very strong repulsion between points.

5 Conclusion

In this paper, we discussed the Metropolis-Hastings algorithm in problems with one unknown and two unknown parameters. We also applied the algorithm to simulate SPPs, conditioning on the number of points in the pattern. To extend the paper, researchers could carry out simulation of SPPs without conditioning on n, the number of points in the pattern. Reversible jump MCMC, or the birth-death algorithm, could be used to simulate SPP's without conditioning on n.

6 Appendix A: R Code for the Beta-Binomial Problem

```
# Y|theta ~ binomial(n, theta)
# theta ~ beta(alpha = 1, beta = 2/3)
# n = 100, y = 65

# Actual posterior:
# theta|y ~ beta(alpha + y, beta + n - y) = beta(66, 35.667)
# E(theta|y) = 0.6492
# Var(theta|y) = 0.00222

set.seed(23)

n <- 100
y <- 65

theta <- c(0.5) # starting value

burnin <- 1000
iter <- 100000
accept <- 0
```

```
for(i in 1:(burnin + iter)) {

    # proposal ratio
    ################

    # beta proposal
    thetaprime <- rbeta(1, 3, 2)
    prop_ratio <- dbeta(theta[i], 3, 2) / dbeta(thetaprime, 3, 2)

    # Random Walk
#   thetaprime <- runif(1, theta[i] - 0.1, theta[i] + 0.1)
#   if (thetaprime > 0 && thetaprime < 1) prop_ratio <- 1
#   else prop_ratio <- 0


    # prior ratio
    ##############
    prior_ratio <- dbeta(thetaprime, 1, 2/3) / dbeta(theta[i], 1, 2/3)


    # likelihood ratio
    ###################
    like_ratio <- dbinom(y, n, thetaprime) / dbinom(y, n, theta[i])


    # acceptance probability
    ########################
    acc_prob <- min(1, like_ratio * prior_ratio * prop_ratio)


    # accept/reject
    ###############
```

```
    if (runif(1) < acc_prob) { # accept
        theta[i+1] <- thetaprime
        accept = accept + 1
    }
    else { # reject
        theta[i+1] <- theta[i]
    }
}

# Report posterior mean and variance
cat("posterior mean:     ", mean(theta[-burnin]), "\n")
cat("posterior variance: ", var(theta[-burnin]), "\n")

# Acceptance rate
acc_rate <- accept / (burnin + iter)

# 95% credible set
quantile(theta, 0.025)
quantile(theta, 0.975)

par(mfrow = c(1, 2))
# Plot MCMC output
plot(theta, type='l', main = 'MCMC Output for Proposal Beta(3,2)',
    xlab = 'Iteration t', ylab = 'X')

# Plot prior and posterior distributions
colors <- c('black', 'red', 'blue')
labels <- c('Estimated Posterior', 'Actual Posterior', 'Prior')
plot(density(theta[-burnin]), lwd=2, xlim = c(0, 1),
    main = 'Prior and Posterior Distributions', xlab = 'x value', ylab = 'Density')
grd <- seq(0, 1, 0.001)
lines(grd, dbeta(grd, 66, 35.6667), col = "red", lwd=2)
```

```
lines(grd, dbeta(grd, 1, 2/3), col = "blue", lwd=2)
legend('topleft', lwd = 2, labels, col = colors)

# Geweke's convergence diagnostic
install.packages("boa")
library(boa)
theta_matrix <- as.matrix(theta)
rownames(theta_matrix) <- c(1:length(theta))
colnames(theta_matrix) <- c("theta")
boa.geweke(theta_matrix, 0.1, 0.5)
```

7 Appendix B: R Code for the Two-Parameter Normal Problem

```
# Data
######
n = 10
y <- c(3.659798, 6.203945, 4.200632, 6.837608, 1.433226,
6.449043, 3.906239, 1.211519, 2.543834, 2.785943)

# Starting values
mu <- c(0)
sigma <- c(1)
sd <- 1

burnin <- 2000
iter <- 10000
move_mu <- 0
move_sigma <- 0
accept_mu <- 0
accept_sigma <- 0
```

```
set.seed(23)

for (i in 1:(burnin + iter)) {

    # proposal ratio
    ################

    # choose move type at random
    u <- runif(1)
    if (u <= 0.5) {
        movetype = "mu"
        move_mu = move_mu + 1
    }
    else {
        movetype = "sigma"
        move_sigma = move_sigma + 1
    }

    # RW proposal (1-at-a-time)
      # RW proposal for mu
    if (movetype == "mu") {
        muprime <- runif(1, mu[i] - 5, mu[i] + 5)
        sigmaprime <- sigma[i]
        prop_ratio <- 1
    }
      #Normal RW proposal for mu
#   if (movetype == 'mu'){
#     muprime <- rnorm(1, mu[i], sd)
#     sigmaprime <- sigma[i]
#     prop_ratio <- dnorm(mu[i], muprime, sd) / dnorm(muprime, mu[i], sd)
#   }
```

```
if (movetype == "sigma") {
    muprime <- mu[i]
    sigmaprime <- runif(1, sigma[i] - 1, sigma[i] + 1)
    if (sigmaprime > 0) prop_ratio <- 1
    else prop_ratio <- 0
}



# prior ratio
##############
if (sigmaprime > 0) {
    prior_ratio <- (dnorm(muprime, 0, 100) * dchisq(sigmaprime, 2)) /
                   (dnorm(mu[i], 0, 100) * dchisq(sigma[i], 2))
}
else prior_ratio <- 0

# likelihood ratio
##################
if (sigmaprime > 0) {
    like_ratio <- 1
    for (j in 1:n) {
        like_ratio <- like_ratio * dnorm(y[j], muprime, sigmaprime) /
                      dnorm(y[j], mu[i], sigma[i])
    }
}
else like_ratio <- 0

# acceptance probability
########################
acc_prob <- min(1, like_ratio * prior_ratio * prop_ratio)
```

```
    # accept/reject
    ###############
    if (runif(1) < acc_prob) { # accept
        mu[i+1] <- muprime
        sigma[i+1] <- sigmaprime
        if (movetype == 'mu') accept_mu = accept_mu + 1
        if (movetype == 'sigma') accept_sigma = accept_sigma + 1
    }
    else { # reject
        mu[i+1] <- mu[i]
        sigma[i+1] <- sigma[i]
    }
}

# Report estimated posterior means
cat("posterior mean of mu:    ", mean(mu[-burnin]), "\n")
cat("posterior mean of sigma: ", mean(sigma[-burnin]), "\n")

# Plot MCMC output and estimated posterior distribution
par(mfrow = c(2, 2))
grd <- seq(0, 15, 0.001)
plot(mu, type='l', main = 'MCMC Output for Mu \n RW +/-5 Proposal',
    xlab = 'Iteration t', ylab = 'X')
plot(density(mu[-burnin]), lwd=2,
    main = 'Estimated Posterior Distribution for Mu \n RW +/-5 Proposal',
    xlab = 'mu value', ylab = 'Density')
grd <- seq(0, 5, 0.001)
plot(sigma, type='l', main = 'MCMC Output for Sigma \n RW +/-1 Proposal',
    xlab = 'Iteration t', ylab = 'X')
plot(density(sigma[-burnin]), lwd=2,
    main = 'Estimated Posterior Distribution for Sigma \n RW +/-1 Proposal',
```

```
    xlab = 'sigma value', ylab = 'Density')

# 95% credible set
quantile(mu, 0.025)
quantile(mu, 0.975)
quantile(sigma, 0.025)
quantile(sigma, 0.975)

# Acceptance rate
accept_mu / move_mu
accept_sigma / move_sigma

# Geweke's convergence diagnostic
theta_matrix <- cbind(mu, sigma)
rownames(theta_matrix) <- c(1:length(mu))
boa.geweke(theta_matrix, 0.1, 0.5)
```

8 Appendix C: R Code for Simulation of Spatial Point Pattern

```
# Sampling window W = [0,1]x[0,1]
# n = 50
# Straussian pair potential function phi(s) = h if ||xi-xj|| < b (0 otherwise)

set.seed(23)
n <- 50
x.curr <- runif(n, 0, 1)
y.curr <- runif(n, 0, 1)

b <- 0.1
```

```
h <- 2

x.prop <- x.curr
y.prop <- y.curr

# Plot initial spatial point pattern
par(pty = "s")
par(mfrow = c(1, 2))
plot(x.curr, y.curr, main = "Initial SPP", xlab = 'x', ylab = 'y')

# Compute total potential energy (exponent in exponential function in likelihood)
tpe <- function(x, y, b, h) {
    tpe.tmp <- 0
    for (i in 1:(n-1)) {
        for (j in (i+1):n) {
            if (sqrt((x[i] - x[j])^2 + (y[i] - y[j])^2) < b) {
                tpe.tmp <- tpe.tmp + h
            }
        }
    }
    return(tpe.tmp)
}

# Main mcmc loop
for (i in 1:2000) {

    # choose point to move
    i <- sample(1:n, 1)

    # propose to move i^th point uniformly in W
    x.prop[i] <- runif(1, 0, 1)
    y.prop[i] <- runif(1, 0, 1)
```

```
    proposal.ratio <- 1

    # likelihood ratio
    tpe.prop <- tpe(x.prop, y.prop, b, h)
    tpe.curr <- tpe(x.curr, y.curr, b, h)
    likelihood.ratio <- exp(-tpe.prop + tpe.curr)

    # accept/reject
    acc.prob <- min(1, likelihood.ratio * proposal.ratio)
    if (runif(1, 0, 1) < acc.prob) { # accept
        x.curr <- x.prop
        y.curr <- y.prop
    }
    else { # reject (i.e. reset x.prop and y.prop)
        x.prop <- x.curr
        y.prop <- y.curr
    }
}

# Plot resulting spatial point pattern
plot(x.curr, y.curr, main = "Strauss SPP \n b = 0.1, h = 2", xlab = 'x', ylab = 'y')
```

References

Matthew A. Bognar and Mary Kathryn Cowles. Bayesian inference for pairwise interacting point processes. *Stat. Comput.*, 14(2):109–117, 2004. ISSN 0960-3174. doi: 10.1023/B:STCO.0000021409.73461.b9. URL `http://dx.doi.org.proxy.lib.uiowa.edu/10.1023/B:STCO.0000021409.73461.b9`.

Mary Kathryn Cowles and Bradley P. Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *J. Amer. Statist. Assoc.*, 91(434):883–904, 1996. ISSN 0162-1459. doi: 10.2307/2291683. URL `http://dx.doi.org/10.2307/2291683`.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis*. Texts in Statistical Science Series. Chapman & Hall, London, 1995. ISBN 0-412-03991-5.

John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *Bayesian statistics, 4 (Peñíscola, 1991)*, pages 169–193. Oxford Univ. Press, New York, 1992.

Brian J. Smith. boa: An R package for MCMC output convergence assessment and posterior inference. *Journal of Statistical Software*, 21(11):1–37, 2007.

David J. Strauss. A model for clustering. *Biometrika*, 62(2):467–475, 1975. ISSN 0006-3444.