A Bayesian model for hurricane trajectories

Yimeng Shang, Weijia Xiong, Ruoyuan Qian

5/8/2020

Introduction

Backgroud

Due to the disasters caused by hurricanes in the U.S., there is an increasing desire to predict the performance of hurricane, such as its location, speed and so on.

In this report, a simplified Bayesian model with component-wise M-H algorithm is implemented for the prediction of hurricane speed based on recorded data.

Data

`hurrican356.csv` collected the track data of 356 hurricanes in the North Atlantic area since 1989. For all the storms, their location (longitude & latitude) and maximum wind speed were recorded every 6 hours. The data includes the following variables

1. **ID**: ID of the hurricans
2. **Season**: In which **year** the hurricane occurred
3. **Month**: In which **month** the hurricane occurred
4. **Nature**: Nature of the hurricane

- ET: Extra Tropical
- DS: Disturbance
- NR: Not Rated
- SS: Sub Tropical
- TS: Tropical Storm

5. **time**: dates and time of the record
6. **Latitude** and **Longitude**: The location of a hurricane check point
7. **Wind.kt** Maximum wind speed (in Knot) at each check point

First of all, we break the **time** variable into two new variables: **date**, **hour**. **date** denotes the number of days between the first day in a year. **hour** denotes the number of hours between the first hour in a day. Then we filter the hour to be 0, 6, 12, 18. And we rename the **Season** as **year**.

Moreover, we remove the first and last record for each hurricane because they don't have enough information for calculation.

Finally, we scale the **date** and **year**.

Method

Here we mainly use a simplified Bayesian model to model hurricane trajectories. Specificly, we use componentwise M-H algorithm to develop an MCMC process. Let t be time (in hours) since a hurricane began, and For each hurrican i, we denote $Y_i(t)$ be the wind speed at time t. The following Baysian model was suggested.

$$Y_i(t+6) = \mu_i(t) + \rho_j Y_i(t) + \epsilon_i(t)$$

where $\mu_i(t)$ is the funtional mean, and the errors $\epsilon_i(t)$ follows a normal distributions with mean zero and variance σ^2, independent across t.

So that the

$$\epsilon_i(t) = Y_i(t+6) - (\mu_i(t) + \rho_j Y_i(t)) \overset{\text{i.i.d}}{\sim} N(0, sigma^2)$$

The likelihood is:

$$f_{Y_i(t+6)}(Y_i|\rho_j, \beta, \sigma) = \frac{exp[-\frac{1}{2}(Y_i(t+6) - [\mu_i(t) + \rho_j Y_i(t))]^2 \frac{1}{\sigma^2}]}{\sqrt{(2\pi)}\sigma}$$

The joint likelihood is:

$$f_{Y_{(t+6)}}(Y|\rho_j, \beta, \sigma) = \prod_{n \times m_n} \frac{exp[-\frac{1}{2}(Y_i(t+6) - [\mu_i(t) + \rho_j Y_i(t))]^2 \frac{1}{\sigma^2}]}{\sqrt{(2\pi)}\sigma}$$

We further assume that the mean functions $\mu_i(t)$ can be written as

$$\mu_i(t) = \beta_0 + x_{i,1}(t)\beta_1 + x_{i,2}\beta_2 + x_{i,3}\beta_3 + \sum_{k=1}^{3} \beta_{3+k}\Delta_{i,k}(t-6)$$

where $x_{i,1}(t)$, ranging from 0 to 365, is the day of year at time t, $x_{i,2}$ is the calenda year of the hurrican, and $x_{i,3}$ is the type of hurrican, and $\Delta_{i,k}(t-6) = Y_{i,k}(t) - Y_{i,k}(t-6), k = 1, 2, 3$ are the change of latitude, longitude, and wind speed between $t-6$ and t.

We assume the following prior distributions:

For $\boldsymbol{\beta} = (\beta_k)_{k=0,...,6}$, we assume $\pi(\boldsymbol{\beta})$ is jointly normal with mean 0 and variance $diag(1, 7)$.

We assume that $\pi(\rho)$ follows a trucated normal $N_{[0,1]}(0.5, 1/5)$

$\pi(\frac{1}{\sigma^2})$ follows a inverse-gamma $(0.001, 0.001)$

We denote:

$$\theta = (\boldsymbol{\beta}, \rho, \sigma) = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \rho, \sigma)$$

So the posterior is:

$$posterior = \pi(\theta) \propto f_{Y_{(t+6)}}(Y|\rho_j, \beta, \sigma) \times \pi(\boldsymbol{\beta}) \times \pi(\rho) \times \pi(\frac{1}{\sigma^2})$$

For computation convenience, we take the logorithm of posterior:

$$\pi(\theta) = \sum \frac{exp[-\frac{1}{2}(Y_i(t+6) - [\mu_i(t) + \rho_j Y_i(t))]^2 \frac{1}{\sigma^2}]}{\sqrt{(2\pi)}\sigma} + \pi(\boldsymbol{\beta}) + \pi(\rho) + \pi(\frac{1}{\sigma^2})$$

In this report, we use coordiantewise MH sampling method, so we update one single parameter at a time:

If at step k, we have $\boldsymbol{X}^{(k)} = (x_1^{(k)}, ..., x_p^{(k)})'$ and we denote $\boldsymbol{X}_{-i}^{(k)} = (x_1^{(k)}, ..., x_{i-1}^{(k)}, x_{i+1}^{(k)}, ..., x_p^{(k)})'$. The algorithm is:

Generate a proposed value y_1 from $q_1(y|x_1^{(k)}, \boldsymbol{X}_{-1}^{(k)})$

- Set $x_1^{(k+1)} = y_1$ if a random uniform is less than the acceptance probability

$$\alpha_i(x_1^{(k)}, y1) = min\left\{1, \frac{\pi(y_1, X_{-1}^{(k)})q_1(x_1^{(k)}|y_1, X_{-1}^{(k)})}{\pi(x_1^{(k)}, X_{-1}^{(k)})q_1(y_1|y_1, X_{-1}^{(k)})}\right\}$$

where $X_{-1}^{(k)} = (x_2^{(k)}, ..., x_p^{(k)})$

- Otherwise, set $x_1^{(k+1)} = x_1^{(k)}$

Do this procedure for all components of X is called one step.

We randomly select a start value and update the parameters for 10000 times for the markov chain to be stable. And we take the average of the last 5000 times as the mean. Because each step is correlated with the last step, we further approximated it as a AR(1) process so that the numerical standard error is:

$$nse(\bar{h}_N) \approx \sqrt{\frac{\sigma^2(1+\rho)}{N(1-\rho)}}$$

Then the 95% confidence interval is: $mean \pm se \times 1.96$

Results

Parameter Estimation

We randomly select 80% of the hurricanes and design a MCMC algorithm to estiamte the posterior distributions of the model parameters. We iterate for 10000 times. Figure1 and Figure2 show that all parameters are converged in the last 5000 steps. We discard the first 5000 steps and base our estimates on the last 5000.

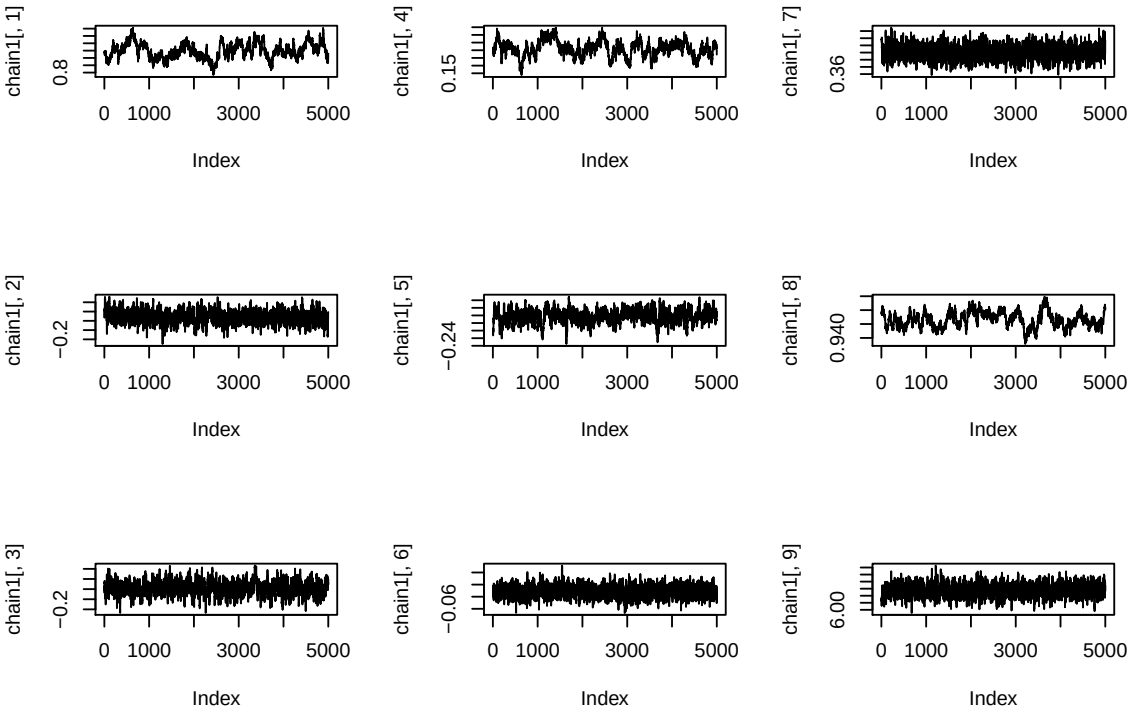

Figure 1: Plots for of the last 5000 MCMC values

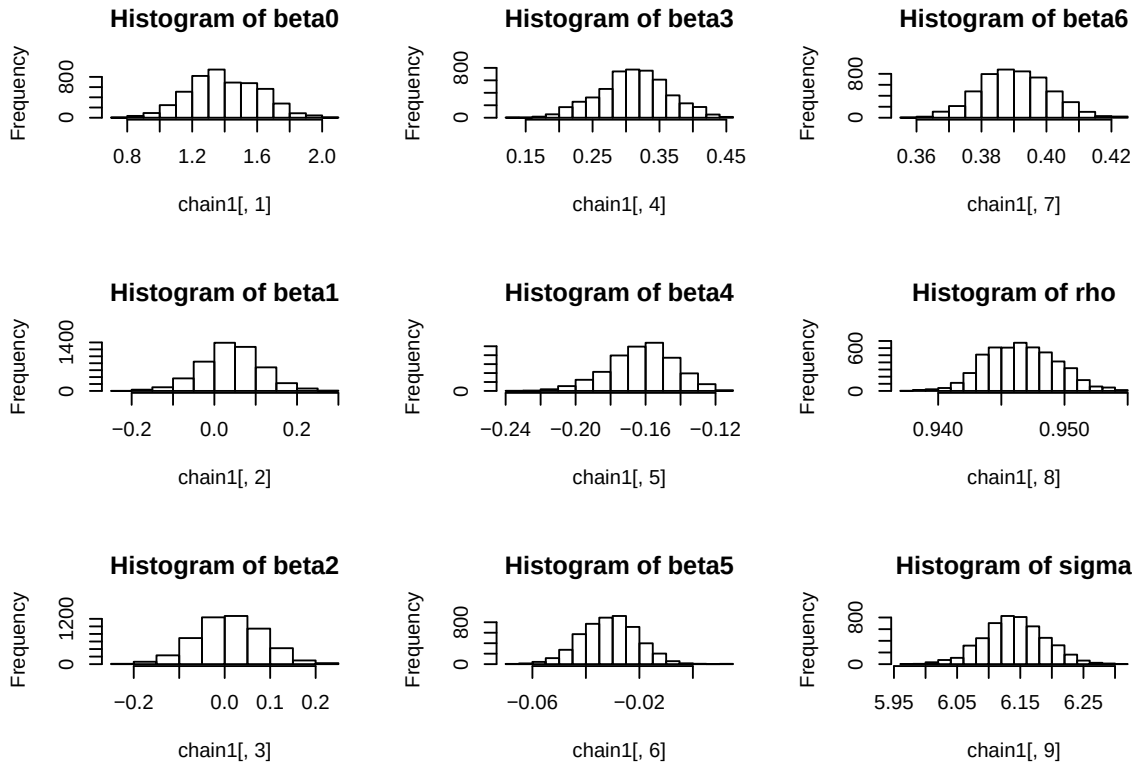

Figure 2: Histograms of the last 5000 MCMC values

Then we estimate the model parameters using posterior means and the numerical standard error is estimated to construct their 95% credibility intervals.

Table 1: Estimates and Credibility Intervals

Parameters	Estimates	CI_{lower}	CI_{upper}
β_0	1.3988	1.3630	1.4346
β_1	0.0408	0.0288	0.0529
β_2	0.0104	-0.0017	0.0225
β_3	0.3108	0.3021	0.3194
β_4	-0.1623	-0.1654	-0.1591
β_5	-0.0312	-0.0330	-0.0295
β_6	0.3903	0.3886	0.3921
ρ	0.9466	0.9461	0.9470
σ	6.1405	6.1324	6.1485

Prediction Results

We apply this model to estimate the speed of the remaining 20% hurricans. And we draw a plot the compare the true wind speed and the estimated speed.

From Figure 3, we can find that two curves are overlapped, which indicates the predicted values are similar with the original values.

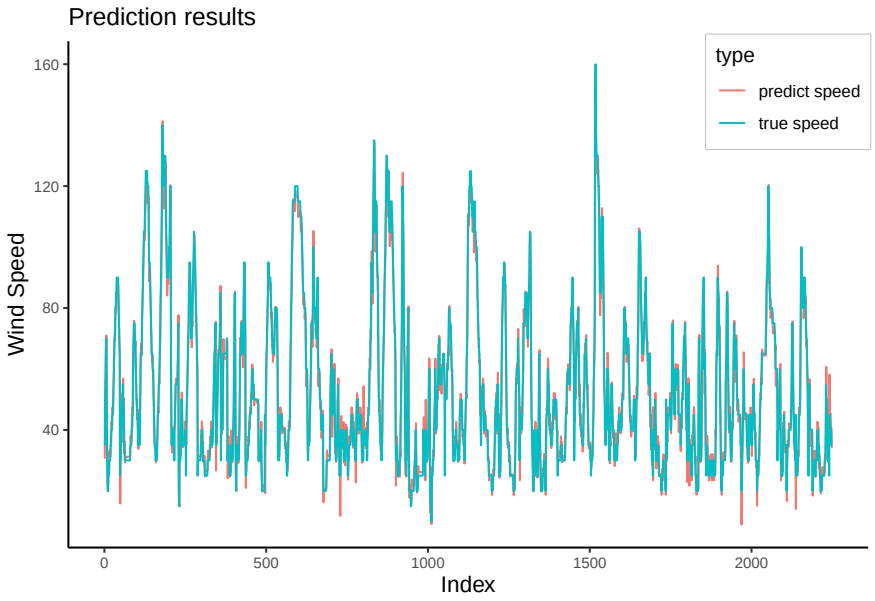

Figure 3: Comparison of true speed and predict speed

Also, we evaluate the performance of predictive ability by calculating the MSE, which is 34.397. Therefore, we may conclude that our model performs well when tracking the wind speed of other hurricanes.

Discussion

In this report, we use a simplified Bayesian model only tracking the wind speed for hurricane trajectories. In future, we may apply other more complicated Bayesian model tracking longitude and latitude along with wind speed to deal with this problem. In addition, here we use componentwise M-H algorithm to develop an MCMC process, which still costs a long time. So further efforts can be done to improve the efficiency of the algorithm.

Appendix

```
library(tidyverse)
library(caret)
library(ModelMetrics)
library(lubridate)
library(stats)
library(invgamma)
library(mvtnorm)
library(truncnorm)
```

```
shift = function(x, n = 1){
  c(x[-(seq(n))], rep(NA, n))
}
```

```
# theta = beta,rho,sigma
beta_ind = 1:7
rho_ind = 8
sigma_ind = 9
```

Data Manipulation

```
raw_data = read_csv("./hurrican356.csv") %>% janitor::clean_names() %>% dplyr::select(-x1)
data = raw_data %>%
  mutate(year = season) %>%
  separate(time, into = c("date", "hour"), sep = " ") %>%
  mutate(date = str_replace(date, "\\(", ""),
         hour = str_replace(hour, "\\)", ""),
         hour = as.integer(str_replace(hour, ":00:00", "")),
         date = yday(date)) %>%
  filter(hour == 0 | hour == 6 | hour == 12 | hour == 18) %>%
  group_by(id,date) %>%
  mutate(sum_hour = sum(hour),
         count_hour = length(hour)) %>%
  filter(sum_hour == 36 & count_hour == 4) %>%
  ungroup(id,date) %>%
  mutate(type = as.numeric(as.factor(nature)),
         d_lati = c(NA, diff(latitude)),
         d_longti = c(NA, diff(longitude)),
         d_wind = c(NA, diff(wind_kt)),
         latitude_shift = shift(latitude),
         longitude_shift = shift(longitude),
         windkt_shift = shift(wind_kt)
         ) %>%  na.omit() %>%
  mutate(
    date = scale(date, center = T, scale = T),
    year = scale(year, center = T, scale = T)
  ) %>%
  dplyr::select(id, date, year, type, d_lati, d_longti, d_wind,
         latitude, longitude, wind_kt,hour,
         latitude_shift, longitude_shift, windkt_shift)
```

```r
set.seed(1)
id = unique(data$id)
num_id = length(id)
train_id = sample(id, 0.8*num_id)

train_data = data[which(data$id %in% train_id),]
test_data = data[which(!(data$id %in% train_id)),]
```

Function Preparation

```r
componentwiseMHstep = function(x, a, logp, pars) {
  # theta: beta, rho, sigma. stuff we need to sample ~ logp
  # a: search window
  # pars: constants needed in logp i.e. data
  p = length(x)
  res = x
  for (i in 1:p) {
    prop = res
    prop[i] = x[i] + 2 * (runif(1) - 0.5) * a[i]
    if (log(runif(1)) < logp(prop, pars) - logp(res, pars))
      res[i] = prop[i] # update res if accepted
  }
  return(res)
}
```

```r
## log multivariate density  ... log likelihood
loglikeli= function(theta,data){
  beta = theta[beta_ind] # 1*7
  rho = theta[rho_ind] # 1*1
  sigma = theta[sigma_ind] # 1*1
  x0 = rep(1,nrow(data))
  x = bind_cols(x0 = x0, data[,2:7]) %>% as.matrix()
  y = data$wind_kt
  ustar= x %*% beta + rho * y
  yshift = data$windkt_shift
  logy = -(yshift - ustar)^2/(2*sigma^2) - log((sqrt(2*pi))*sigma)

  data$logy = logy
  loglikelihoodsum = sum(data$logy)
  return(loglikelihoodsum)
}
```

```r
## The log of the prior:
logprior= function(theta) {
  beta = theta[beta_ind] # 1*7
  rho = theta[rho_ind] # 1*1
  sigma = theta[sigma_ind] # 1*1

  pi_beta = log(dmvnorm(beta,rep(0,length(beta)),diag(1,length(beta))))
  pi_rho = log(dtruncnorm(rho, a=0, b=1, mean = 0.5, sd = 0.2))
  pi_sigma = dinvgamma(1/sigma^2, 0.001, 0.001, log = TRUE)
  return(pi_beta + pi_rho + pi_sigma)
}
```

```
## The log of the posterior:
logposterior = function(theta,data) {

  beta = theta[beta_ind] # 1*7
  rho = theta[rho_ind] # 1*1
  sigma = theta[sigma_ind] # 1*1

  logpost = loglikeli(theta,data) + logprior(theta)
  if(sigma <= 0) {
    return(-Inf)
  } else
    return(logpost)
}
```

```
numunique = function(mat) {
  for (i in 1:ncol(mat)) {
    cat(i, "\t", length(unique(mat[,i])), "\n")
  }
}
```

```
nse = function(x){
  var_h = apply(x, 2, var)
  pho = mean(x[,8])
  N = length(x[,8])
  nes = sqrt((var_h * (1+pho))/(N * (1-pho)))
  return(nes)
}
```

```
predict_new = function(theta, data){
  beta = theta[beta_ind] # 1*7
  rho = theta[rho_ind] # 1*1
  sigma = theta[sigma_ind] # 1*1
  x0 = rep(1,nrow(data))
  x = bind_cols(x0 = x0, data[,2:7]) %>% as.matrix()
  y = data$wind_kt
  ustar = x %*% beta + rho * y
  return(ustar)
}
```

Run MH sampling

```
MH_sample = function(theta,search_window,data,nrep){
    chain = matrix(NA, nrep, length(theta))
    x = theta
    for (j in 1:nrep) {
      newx = componentwiseMHstep(x, a = search_window,
                                 logp = logposterior,
                                 pars = data)
      chain[j,] = newx
      x = newx
#      if (j %% 100 == 0) {
#        print(j)
#      }
```

```
    }
    # print(k)
  return(chain)
}
```

Test

```
theta1 = c(rep(1,7), 1, 10)
search_window1 = c(0.09, 0.1, 0.09, 0.03, 0.015, 0.03, 0.02, 0.001, 0.08)
```

```
result1 = MH_sample(theta = theta1, data = train_data,
                    search_window = search_window1, nrep = 10000)
```

```
numunique(result1)
chain1 = result1
```

Visualization for test case

```
chain1 = result1[5001:10000,]

layout(matrix(c(1:9),3,3))
plot(chain1[,1],type = "l", main="trace plot of beta0")
plot(chain1[,2],type = "l", main="trace plot of beta1")
plot(chain1[,3],type = "l", main="trace plot of beta2")
plot(chain1[,4],type = "l", main="trace plot of beta3")
plot(chain1[,5],type = "l", main="trace plot of beta4")
plot(chain1[,6],type = "l", main="trace plot of beta5")
plot(chain1[,7],type = "l", main="trace plot of beta6")
plot(chain1[,8],type = "l", main="trace plot of rho")
plot(chain1[,9],type = "l", main="trace plot of sigma")
```

```
layout(matrix(c(1:9),3,3))
hist(chain1[,1], main="Histogram of beta0")
hist(chain1[,2], main="Histogram of beta1")
hist(chain1[,3], main="Histogram of beta2")
hist(chain1[,4], main="Histogram of beta3")
hist(chain1[,5], main="Histogram of beta4")
hist(chain1[,6], main="Histogram of beta5")
hist(chain1[,7], main="Histogram of beta6")
hist(chain1[,8], main="Histogram of rho")
hist(chain1[,9], main="Histogram of sigma")
```

CI

```
est_theta = apply(chain1, 2, mean)
nse_theta = nse(chain1)

CI = data.frame(cbind(qnorm(0.975)* nse_theta %*% t(c(-1,1)) + est_theta, est_theta),
                row.names = c("beta_0","beta_1","beta_2","beta_3","beta_4","beta_5","beta_6","pho","sig
colnames(CI) = c("CI_lower","CI_upper","pont_est")
```

```r
CI %>% knitr::kable(caption = "CI for parameters")
```

Prediction

```r
predict_wind_point = predict_new(est_theta, test_data)
predict_wind_lower = predict_new(CI$CI_lower, test_data)
predict_wind_upper = predict_new(CI$CI_upper, test_data)

test_y = test_data$windkt_shift
index = seq(1:length(test_y))

pred_res = data.frame(index, test_y, predict_wind_point, predict_wind_lower, predict_wind_upper)

test_mse = mse(test_y,predict_wind_point)
test_mse
```

```r
plot_df = pred_res %>% select(-predict_wind_lower, -predict_wind_upper) %>% pivot_longer(
  2:3,
  values_to = "data",
  names_to = "type"
)

ggplot(plot_df) + geom_line(aes(x = index, y = data, color = type)) +
  scale_color_discrete(labels = c("predict speed", "true speed")) +
  theme_classic()+
  theme(axis.text.x = element_text(size = 8), axis.title.x = element_text(size = 12),
        axis.text.y = element_text(size = 8), axis.title.y = element_text(size = 12),
        plot.title = element_text(size = 13),
        legend.position = c(0.9, 0.9),
        legend.background = element_rect(
                                  size=0.2, linetype="solid",
                                  colour ="grey"))+
  labs(x = "Index", y = "Wind Speed", title = "Prediction results")
```

Another start value

```r
theta2 = c(rep(0.05,7), 0.1, 10)
search_window2 = c(0.3, 0.02, 0.3, 0.3, 0.15, 0.3, 0.2, 0.005, 0.2)
```

```r
result2 = MH_sample(theta = theta2, data = train_data,
                    search_window = search_window2, nrep = 10000)
```

```r
# dim(chain1)
layout(matrix(c(1:9),3,3))
chain2 = result1[5000:10000,]
plot(chain1[,1],type = "l")
plot(chain1[,2],type = "l")
plot(chain1[,3],type = "l")
plot(chain1[,4],type = "l")
plot(chain1[,5],type = "l")
plot(chain1[,6],type = "l")
```

```r
plot(chain1[,7],type = "l")
plot(chain1[,8],type = "l")
plot(chain1[,9],type = "l")
```

```r
est_theta = apply(chain2, 2, mean)
## same value
```