Hw06

Chunlei Liu

10/23/2018

Q1:

The target distribution is $\pi(x) \propto e^{-x^4-x} = l(x)$, at time 0, start with $x^{(0)} = 0$, in general in the state x^t, draw y from the proposal distribution $T(y|x^{(t)}) \sim N(x^{(t)}, 2)$, the proposal distribution is a symmetric distribution.

Draw $u \sim Uniform(0,1), r(y, x^t) = min\left\{1, \frac{\pi(y)T(x|y)}{\pi(x)T(y|x)}\right\} = min\left\{1, \frac{\pi(x)}{\pi(y)}\right\}$

$$x^{(t+1)} \begin{cases} y, u \leqq r(x^{(t)}, y) \\ x^{(t)}, otherwise \end{cases} .$$

Follow this procedure , we can generate $x^{(1)}...x^{(T)} \sim l(x)$.

Finally, I choose the last 5000 samples. T = 5000

The times of $x^{(t+1)} = y$ divided by the total times is the acceptance rate. $\frac{\sum_{i=1}^{T} x^{(i)}}{T}$ is the mean. $\frac{\sigma^2(1+2\sum_{j=1}^{\infty}\rho_j)}{T}$ is variance.

The acceptance rate is 0.31498.

The mean of the above density function is -0.3200664.

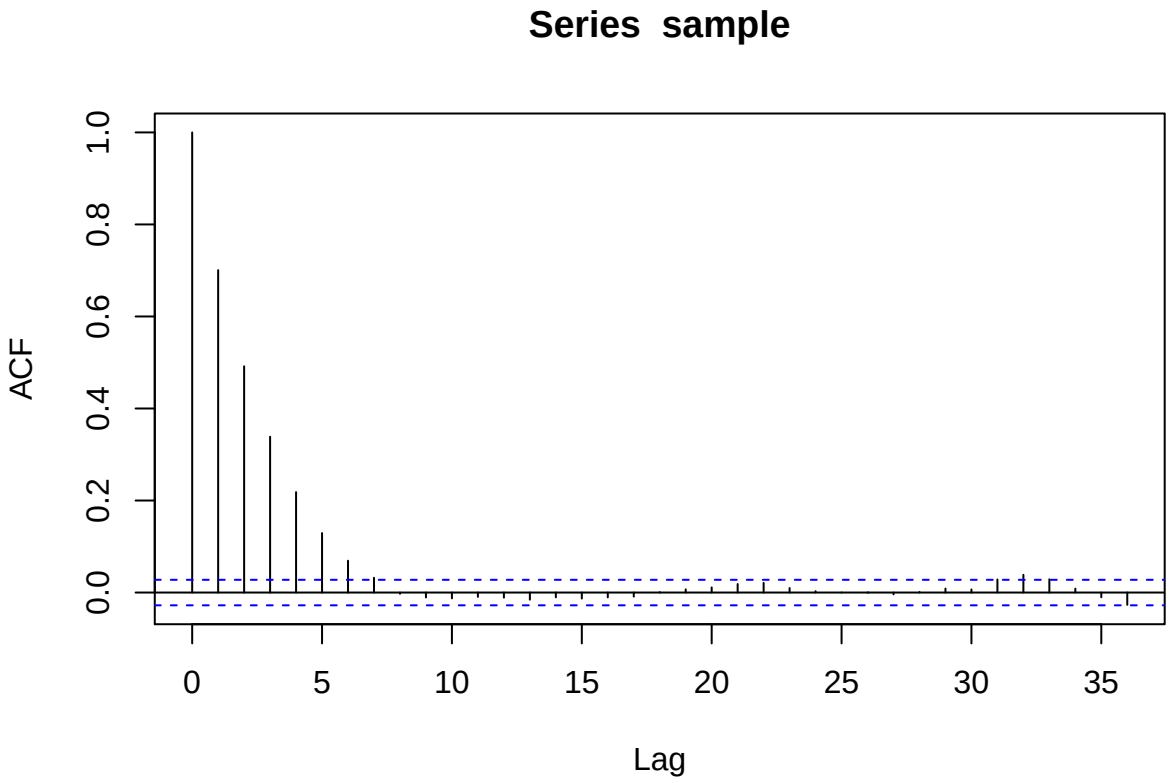
The standart error is 0.01689385.

```
rm(list = ls())
set.seed(1)
T = 50000
x = rep(0, T)
x0 = 0
count = 0
for (t in 1:T){
  if (t == 1) {
    y = rnorm(1, x0, 2)
    u = runif(1, 0, 1)
    r = min(1, exp(-y^4-y)/exp(-x0^4-x0))
    if (u <= r) {
      x[t] = y
      count = count + 1
    }
    else{
      x[t] = x0
    }
  }
  else{
    y = rnorm(1, x[t-1], 2)
    u = runif(1, 0, 1)
    r = min(1, exp(-y^4-y)/exp(-x[t-1]^4-x[t-1]))
    if (u <= r) {
      x[t] = y
      count = count + 1
    }
```

```
    else{
      x[t] = x[t-1]
    }
  }
}
accept_rate = count/T
accept_rate
```

```
## [1] 0.31498
```

```
sample = tail(x, 5000)
acf(sample)
```




```
mean(sample)
```

```
## [1] -0.3200664
```

```
sum_acf = sum(acf(sample, plot = FALSE)$acf) - 1
variance = var(sample)*(1+2*sum_acf)/length(sample)
sqrt(variance)
```

```
## [1] 0.01689385
```

Q2:

The target distribution: $P(T) \propto \frac{1}{\prod_{i=1}^{12}\prod_{j=1}^{12} t_{ij}!} = l(T)$.

At t = 0, start with current table $T^{(0)}$. In general, draw y from the proposal distribution $G(y|T^{(t)})$, at each step, pick two rows and columns uniformly, add or subtract one in the four entries at the interaction of the two rows and columns with two patterns with equal probability. Draw $u \sim Uniform(0,1)$, because the proposal distribution is symmetric, $r(T^{(t)}, y) = min\left\{1, \frac{l(y)}{l(T^{(t)})}\right\}$.

$$T^{(t+1)} \begin{cases} y, u \leqq r(T^{(t)}, y) \\ T^{(t)}, otherwise \end{cases} .$$

Follow this procedure , we can generate $T^{(1)}...T^{(T)} \sim l(T)$.

Finally, I choose the last 20000 samples.

The p value is 0.766, which is the mean of the vector.

The standard error is 0.024623355, similar to the formula in Q1.

```
set.seed(23)
rm(list = ls())
T0 <- read.delim("xid-36891322_1.txt", row.names=1, sep="", stringsAsFactors=FALSE)[1:12, 1:12]
T = 50000
T_matrix = list()
for (i in 1:T) {
  row_index = sample.int(dim(T0)[1], 2)
  col_index = sample.int(dim(T0)[2], 2)
  u1 = runif(1)
  if (i == 1){
    if (u1 <= 0.5) {
      y = T0[,]
      y[row_index[1], col_index[1]] = T0[row_index[1], col_index[1]] + 1
      y[row_index[1], col_index[2]] = T0[row_index[1], col_index[2]] - 1
      y[row_index[2], col_index[1]] = T0[row_index[2], col_index[1]] - 1
      y[row_index[2], col_index[2]] = T0[row_index[2], col_index[2]] + 1
      u2 = runif(1)
      ratio = (1/prod(factorial(y)))/(1/prod(factorial(T0)))
      if (is.na(ratio) == TRUE){
        r = min(1, 0)
      }
      else{
        r = min(1, ratio)
      }
      if (u2 <= r){
        T_matrix[[i]] = y
      }
      else{
        T_matrix[[i]] = T0
      }
    }
    else{
      y = T0[,]
      y[row_index[1], col_index[1]] = T0[row_index[1], col_index[1]] - 1
      y[row_index[1], col_index[2]] = T0[row_index[1], col_index[2]] + 1
      y[row_index[2], col_index[1]] = T0[row_index[2], col_index[1]] + 1
      y[row_index[2], col_index[2]] = T0[row_index[2], col_index[2]] - 1
      u2 = runif(1)
      ratio = (1/prod(factorial(y)))/(1/prod(factorial(T0)))
      if (is.na(ratio) == TRUE){
        r = min(1, 0)
      }
      else{
        r = min(1, ratio)
      }
```

```
      if (u2 <= r){
        T_matrix[[i]] = y
      }
      else{
        T_matrix[[i]] = T0
      }
    }
  }
  else{
    if (u1 <= 0.5) {
      y = T_matrix[[i-1]]
      y[row_index[1], col_index[1]] = T_matrix[[i-1]][row_index[1], col_index[1]] + 1
      y[row_index[1], col_index[2]] = T_matrix[[i-1]][row_index[1], col_index[2]] - 1
      y[row_index[2], col_index[1]] = T_matrix[[i-1]][row_index[2], col_index[1]] - 1
      y[row_index[2], col_index[2]] = T_matrix[[i-1]][row_index[2], col_index[2]] + 1
      u2 = runif(1)
      ratio = (1/prod(factorial(y)))/(1/prod(factorial(T_matrix[[i-1]])))
      if (is.na(ratio) == TRUE){
        r = min(1, 0)
      }
      else{
        r = min(1, ratio)
      }
      if (u2 <= r){
        T_matrix[[i]] = y
      }
      else{
        T_matrix[[i]] = T_matrix[[i-1]]
      }
    }
    else{
      y = T_matrix[[i-1]]
      y[row_index[1], col_index[1]] = T_matrix[[i-1]][row_index[1], col_index[1]] - 1
      y[row_index[1], col_index[2]] = T_matrix[[i-1]][row_index[1], col_index[2]] + 1
      y[row_index[2], col_index[1]] = T_matrix[[i-1]][row_index[2], col_index[1]] + 1
      y[row_index[2], col_index[2]] = T_matrix[[i-1]][row_index[2], col_index[2]] - 1
      u2 = runif(1)
      ratio = (1/prod(factorial(y)))/(1/prod(factorial(T_matrix[[i-1]])))
      if (is.na(ratio) == TRUE){
        r = min(1, 0)
      }
      else{
        r = min(1, ratio)
      }
      if (u2 <= r){
        T_matrix[[i]] = y
      }
      else{
        T_matrix[[i]] = T_matrix[[i-1]]
      }
    }
  }
}
```

```r
p_T0 = 1/(prod(factorial(T0)))
sample = tail(T_matrix, 20000)
res = rep(0, length(sample))
for (i in 1:length(sample)) {
  if (1/prod(factorial(sample[[i]])) < p_T0) {
    res[i] = 1
  }
}
mean(res)
```

```
## [1] 0.766
```

```r
sqrt(var(res)*(1+2*(sum(acf(res, plot = FALSE)$acf) - 1))/length(sample))
```

```
## [1] 0.02462335
```