statComp_hw7

凌浩东

12/7/2021

9.1

```
set.seed(42)

# the Rayleigh distribution pdf
f <- function(x, sigma) {
  if (any(x < 0)) return (0)
  stopifnot(sigma > 0)
  return ((x / sigma^2) * exp(-x^2 / (2*sigma^2)))
}

MHsampler <- function(sigma) {
  # length of loop
  m <- 10000
  # initiate the MC
  x <- numeric(m)
  # chi-square as proposal
  x[1] <- rchisq(1, df=1)
  # count of the rejected sample
  k <- 0
  # generate uniform numbers
  u <- runif(m)

  for (i in 2:m) {
    xt <- x[i-1]
    y <- rchisq(1, df = xt)
    # numerator
    num <- f(y, sigma) * dchisq(xt, df = y)
    # denominator
    den <- f(xt, sigma) * dchisq(y, df = xt)
    if (u[i] <= num/den) x[i] <- y else {
      x[i] <- xt
      k <- k + 1
    }
  }
  print(k)
  return (x)
}

x1 <- MHsampler(4)
```

```
## [1] 4091
```

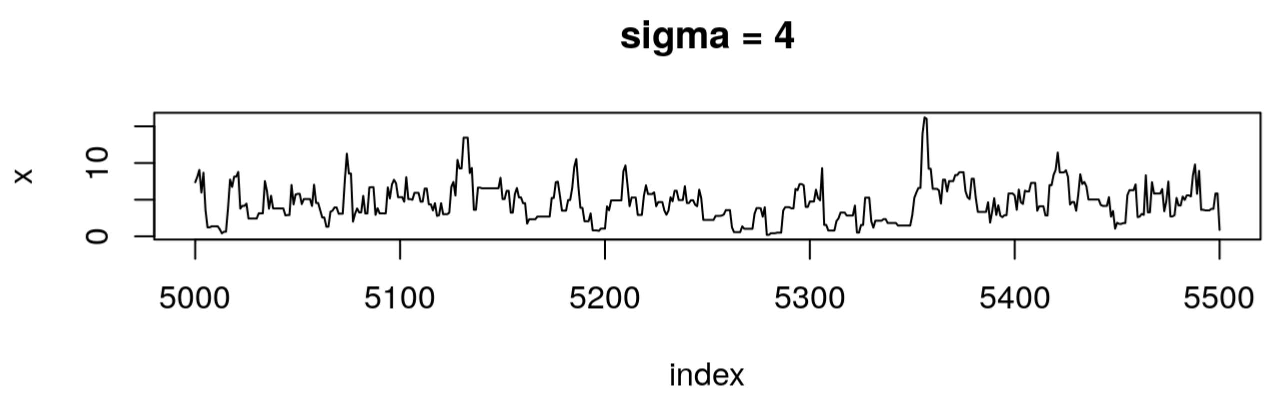
```
x2 <- MHsampler(2)
```

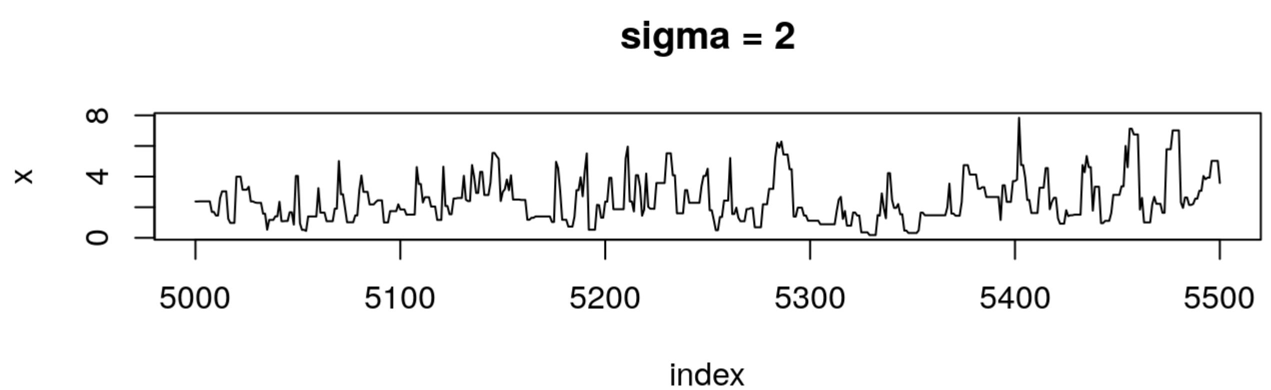
```
## [1] 5193
```

For $\sigma = 4$, approximately 40% of candidate points are rejected, while for $\sigma = 2$, approximately 50% of candidate points are rejected.

Known that $E[X] = \sigma\sqrt{\frac{\pi}{2}}$ and $Var(X) = \sigma^2\frac{4-\pi}{2}$, we can tell that the for $\sigma = 4$, the expected value is 5.013 and variance is 6.867; for $sigma = 2$, the expected value is 2.507 and variance is 1.617.

```
index <- 5000:5500
par(mfrow = c(2,1))
y1 <- x1[index]
plot(index, y1, type = "l", main = "sigma = 4", ylab = "x")
y2 <- x2[index]
plot(index, y2, type = "l", main = "sigma = 2", ylab = "x")
```





In the above figure, we can see that, when $\sigma = 4$, the sample fluctuate around 5, within the range of 0 to 15; when $\sigma = 2$, the sample fluctuate around 4, within the range of 0 to 8. It's clear that the sample mean for $\sigma = 2$ is smaller and sample variance smaller.

9.3

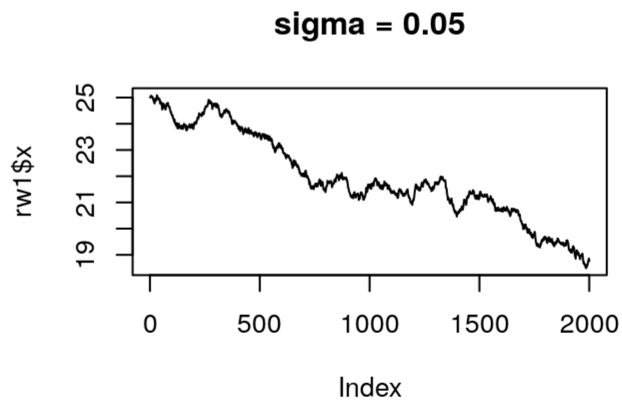
```
set.seed(42)
f <- function(x) {
  1/(pi * (1+x^2))
}


# length of loop
m <- 10000
# initiate the MC
x <- numeric(m)
# normal distribution as proposal
x[1] <- rnorm(1)
# count of the rejected sample
k <- 0
# generate uniform numbers
u <- runif(m)

for (i in 2:m) {
  xt <- x[i-1]
  y <- rnorm(1, xt)
  # numerator
  num <- f(y) * dnorm(xt, y)
  # denominator
  den <- f(xt) * dnorm(y, xt)
  if (u[i] <= num/den) x[i] <- y else {
    x[i] <- xt
    k <- k + 1
  }
}

deciles.MHsampler <- quantile(x[1001:m], probs = seq(.1, .9, by = .1))
deciles.qcauchy <- qt(p = seq(.1, .9, by = .1), df=1)
df <- data.frame(deciles.MHsampler)
df$deciles.qcauchy <- deciles.qcauchy
df
```

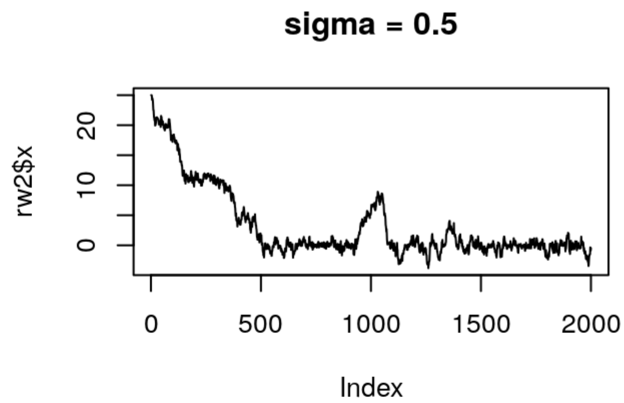
```
##     deciles.MHsampler deciles.qcauchy
## 10%       -3.21230424      -3.0776835
## 20%       -1.53810774      -1.3763819
## 30%       -0.81416347      -0.7265425
## 40%       -0.42000168      -0.3249197
## 50%       -0.08829822       0.0000000
## 60%        0.21668118       0.3249197
## 70%        0.55532343       0.7265425
## 80%        1.12030899       1.3763819
## 90%        2.39592442       3.0776835
```

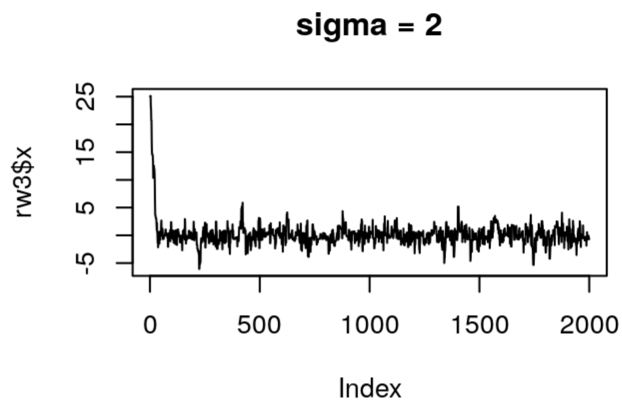
9.4

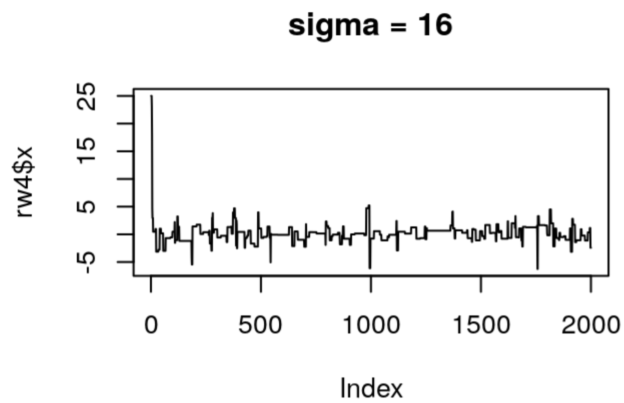
The standard Laplace distribution has the form $f(x) = \frac{1}{2}e^{-|x|}$.

```r
f <- function(x) {
  0.5 * exp(-abs(x))
}

rw.Metropolis <- function(sigma, x0, N) {
  x <- numeric(N)
  x[1] <- x0
  u <- runif(N)
  k <- 0
  for (i in 2:N) {
    y <- rnorm(1, x[i-1], sigma)
    if (u[i] <= (f(y) / f(x[i-1])))
      x[i] <- y
    else {
      x[i] <- x[i-1]
      k <- k + 1
    }
  }
  return(list(x=x, k=k))
}

N <- 2000
sigma <- c(.05, .5, 2, 16)
x0 <- 25
rw1 <- rw.Metropolis(sigma[1], x0, N)
rw2 <- rw.Metropolis(sigma[2], x0, N)
rw3 <- rw.Metropolis(sigma[3], x0, N)
rw4 <- rw.Metropolis(sigma[4], x0, N)

par(mfrow=c(2,2))
plot(rw1$x, type = "l", main = "sigma = 0.05")
plot(rw2$x, type = "l", main = "sigma = 0.5")
plot(rw3$x, type = "l", main = "sigma = 2")
plot(rw4$x, type = "l", main = "sigma = 16")
```









The acceptance rates of each chain is shown below.

```
#proportion of candidate points accepted
print(c(1-rw1$k/N, 1-rw2$k/N, 1-rw3$k/N, 1-rw4$k/N))
```

```
## [1] 0.9735 0.8260 0.5585 0.0950
```

2

Denote $x_{-j} = (x_1, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n)$

In the j-th step of iteration t, the proposal in the is

$$g\left(y_j | x^{(t-1)}\right) = \begin{cases} f\left(y_j | x_{-j}^{(t-1)}\right) & if\ y_{-j} = x_{-j}^{(t-1)} \\ 0 & else \end{cases}$$

Thus the ratio at the j-th step of iteration t is

$$
\begin{aligned}
r &= \frac{f(y)g\left(x^{(t-1)}|y_j\right)}{f\left(x^{(t-1)}\right)g\left(y_j|x^{(t-1)}\right)} \\
&= \frac{f(y)f\left(x_{-j}^{(t-1)}|y_j\right)}{f\left(x^{(t-1)}\right)f\left(y_j|x_{-j}^{(t-1)}\right)} \\
&= \frac{f(y_j, y_{-j})f\left(x_{-j}^{(t-1)}|y_j\right)}{f\left(x_j^{(t-1)}, x_{-j}^{(t-1)}\right)f\left(y_j|x_{-j}^{(t-1)}\right)} \\
&= \frac{f(y_j|y_{-j})f(y_{-j})f\left(x_{-j}^{(t-1)}|y_j\right)}{f\left(x_j^{(t-1)}|x_{-j}^{(t-1)}\right)f\left(x_{-j}^{(t-1)}\right)f\left(y_j|x_{-j}^{(t-1)}\right)} \\
&= \frac{f(y_j|y_{-j})f(y_{-j})f\left(x_{-j}^{(t-1)}|y_j\right)}{f\left(x_j^{(t-1)}|x_{-j}^{(t-1)}\right)f\left(x_{-j}^{(t-1)}\right)f\left(y_j|x_{-j}^{(t-1)}\right)} \\
&= 1
\end{aligned}
$$

The factors are eliminated because $y_{-j} = x_{-j}^{(t-1)}$.

Thus Gibbs sampling can be viewed as a special case of the Metropolis-Hastings algorithm. In Gibbs sampler, every newly proposed sample is accepted with probability one