Chapter 13: Monte Carlo Methods

Alex Chi

Exercises

13.1 (Late to class?).

Suppose the travel times for a particular student from home to school are normally distributed with mean 20 minutes and standard deviation 4 minutes. Each day during a five-day school week she leaves home 30 minutes before class. For each of the following problems, write a short Monte Carlo simulation function to compute the probability or expectation of interest.

a. Find the expected total traveling time of the student to school for a five- day week. Find the simulation estimate and give the standard error for the simulation estimate.

```
sim.travel <- function() sum(rnorm(5, mean=20, sd=4))
result = replicate(1000, sim.travel())
mean(result)
```

```
## [1] 99.89034
```

```
sd(result)
```

```
## [1] 8.780764
```

b. Find the probability that the student is late for at least one class in the five-day week. Find the simulation estimate of the probability and the corresponding standard error.

```
sim.travel <- function() ifelse(sum(rnorm(5, mean=20, sd=4) >= 30) >= 1, 1, 0)
result = replicate(1000, sim.travel())
mean(result)
```

```
## [1] 0.028
```

```
sd(result)
```

```
## [1] 0.1650553
```

c. On average, what will be the longest travel time to school during the five- day week? Again find the simulation estimate and the standard error.

```
sim.travel <- function() max(rnorm(5, mean=20, sd=4))
result = replicate(1000, sim.travel())
mean(result)
```

```
## [1] 24.5515
```

```
sd(result)
```

```
## [1] 2.636612
```

13.2 (Confidence interval for a normal mean based on sample quan- tiles).

Suppose one obtains a normally distributed sample of size n = 20 but only records values of the sample median M and the first and third quartiles Q1and Q3.

a. Using a sample of size n = 20 from the standard normal distribution, sim- ulate the sampling distribution of the statistic S = M Q3Q1. Store the simulated values of S in a vector.

```
sim.distribution <- function() {
  y = quantile(rnorm(20), c(0.25, 0.5, 0.75))
  M = y[2]; Q3 = y[3]; Q1 = y[1]
  M / (Q3 - Q1)
}
s = replicate(100, sim.distribution())
```

b. Find two values, s1,s2, that bracket the middle 90% probability of the distribution of S.

```
quantile(s, c(0.05, 0.95))
```

```
##         5%        95% 
## -0.3670022  0.4225112
```

c. For a sample of size n = 20 from a normal distribution with mean μ and standard deviation σ, it can be shown that P ? s1< M μQ3Q1 < s2 ? = 0.90.

Using this result, construct a 90% confidence interval for the mean μ

Answer: 和上面一样。

d. In a sample of 20, we observe (Q1,M,Q3) = (37.8,51.3,58.2). Using your work in parts (b) and (c), find a 90% confidence interval for the mean μ.

13.3 (Comparing variance estimators).

Suppose one is taking a sample y1,...,ynfrom a normal distribution with mean μ and variance σ2.

a. It is well known that the sample variance is an unbiased estimator of σ2. To confirm this, assume n = 5 and perform a simulation experiment to compute the bias of the sample variance S.

```
sim.var <- function() var(rnorm(5))
result = replicate(1000, sim.var())
x = c(mean(result) - 1, sd(result) / sqrt(1000))
c(x[1] - 2 * x[2], x[1] + 2 * x[2])
```

```
## [1] -0.03178022  0.05717307
```

b. Consider the alternative variance estimator Sc, where c is a constant. Suppose one is interested in finding the estimator Scthat makes the mean squared error MSE = E?(Scσ2)2? as small as possible. Again assume n = 5 and use a simulation experiment to compute the mean squared error of the estimators S3,S5,S7,S9 and find the choice of c (among {3, 5, 7, 9}) that minimizes the MSE.

```r
sim.sc <- function(c) {
  x = rnorm(5)
  sum((x - mean(x)) ^ 2) / c
}
mean(replicate(100, sim.sc(3))^2)
```

```
## [1] 2.610384
```

```r
mean(replicate(100, sim.sc(5))^2)
```

```
## [1] 0.8933126
```

```r
mean(replicate(100, sim.sc(7))^2)
```

```
## [1] 0.6297634
```

```r
mean(replicate(100, sim.sc(9))^2)
```

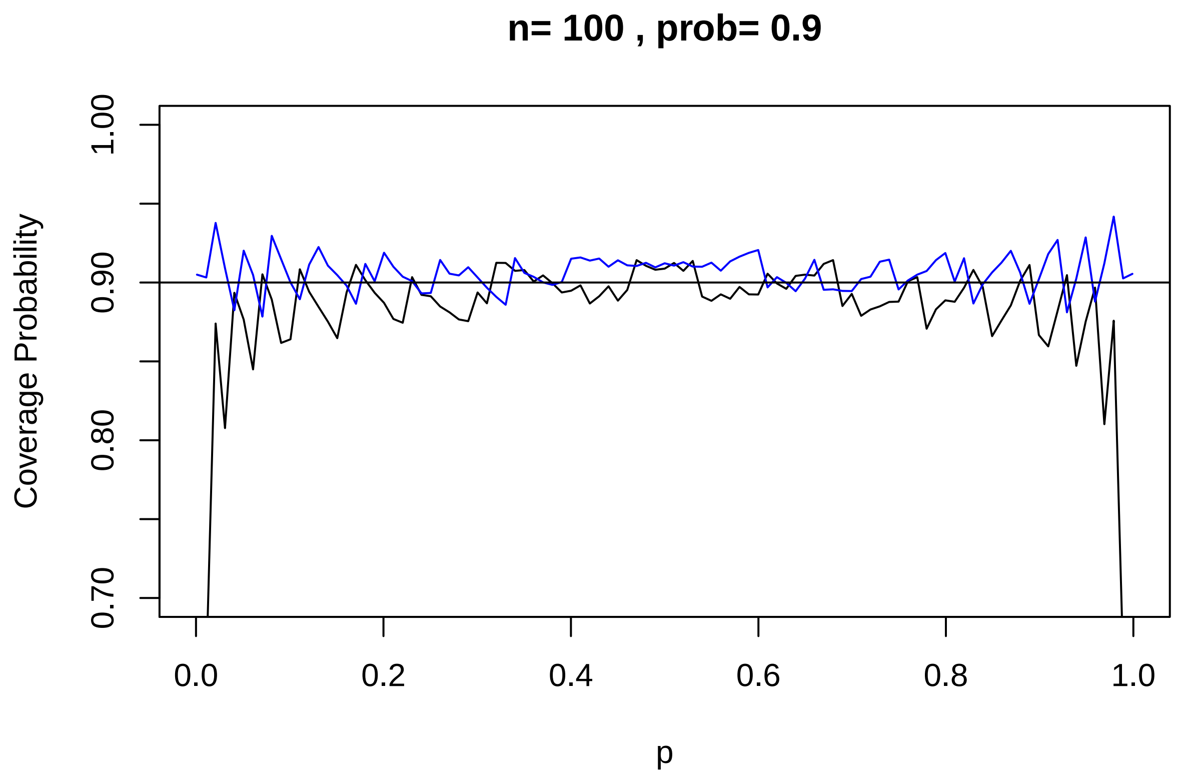
```
## [1] 0.3656897
```

Therefore, select c = 9.

13.4 (Evaluating the "plus four" confidence interval).

A modern method for a confidence interval for a proportion is the "plus-four" interval described in Agresti and Coull [2]. One first adds 4 imaginary observations to the data, two successes and two failures, and then apply the Wald interval to the adjusted sample. Let n = n + 4 denoted the adjusted sample size and p denotes the adjusted sample proportion. Then the "plus-four"interval is given by INT lus□four, where z denote the corresponding 1 □ (1 □ γ)/2 percentile for a standard normal variable. By a Monte Carlo simulation, compute the probability of coverage of the plus-four interval for values of the proportion p between 0.001 and 0.999. Contrast the probability of coverage of the plus-four interval with the Wald interval when the nominal coverage level is γ = 0.90. Does the plus-four in- terval have a 90% coverage probability for all values of p?

```r
wald_plus_4 = function(y ,n, prob) {
  n = n + 4
  p = (y + 2) / n
  z = qnorm(1 - (1 - prob) / 2)
  lb = p - z * sqrt(p * (1 - p) / n)
  ub = p + z * sqrt(p * (1 - p) / n)
  cbind(lb, ub)
}
wald = function(y ,n, prob) {
  n = n
  p = y / n
  z = qnorm(1 - (1 - prob) / 2)
  lb = p - z * sqrt(p * (1 - p) / n)
  ub = p + z * sqrt(p * (1 - p) / n)
  cbind(lb, ub)
}
mc.coverage = function(p, n, prob, iter=10000){
  y = rbinom(iter, n, p)
  c.interval = wald(y, n, prob)
  mean((c.interval[ ,1] < p) & (p < c.interval[ ,2]))
}
mc.coverage_p4 = function(p, n, prob, iter=10000){
  y = rbinom(iter, n, p)
  c.interval = wald_plus_4(y, n, prob)
  mean((c.interval[ ,1] < p) & (p < c.interval[ ,2]))
}
many.mc.coverage = function(p.vector, n, prob)
  sapply(p.vector, mc.coverage, n, prob)
many.mc.coverage_p4 = function(p.vector, n, prob)
  sapply(p.vector, mc.coverage_p4, n, prob)
curve(many.mc.coverage(x, 100, 0.90), from=0.001, to=0.999,
      xlab="p", ylab="Coverage Probability",
      main=paste("n=", 100, ", prob=", 0.90),
      ylim=c(0.7, 1))
curve(many.mc.coverage_p4(x, 100, 0.90), add=TRUE, col="blue")
abline(h=.9)
```



13.5 (Metropolis-Hastings algorithm for the poly-Cauchy distribu- tion).

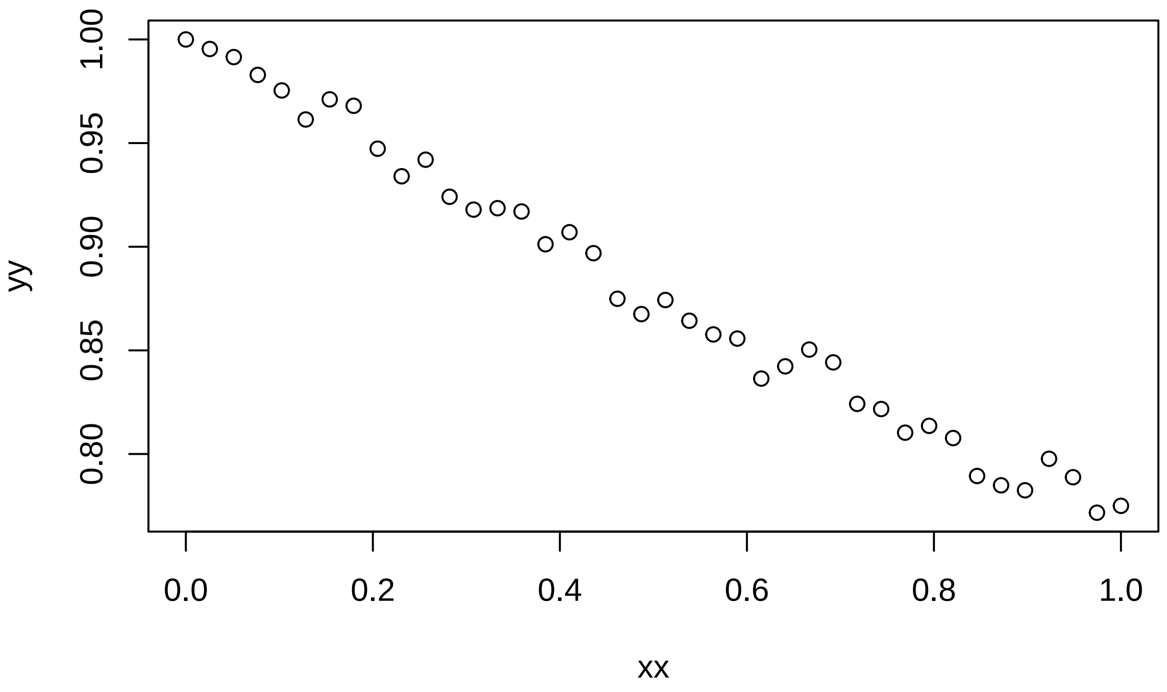
Suppose that a random variable y is distributed according to the poly-Cauchy density, where a = (a1,…,an) is a vector of real-valued parameters. Suppose that n = 6 and a = (1,2,2,6,7,8).

a. Write a function to compute the log density of y. (It may be helpful to use the function dcauchy that computes the Cauchy density.)

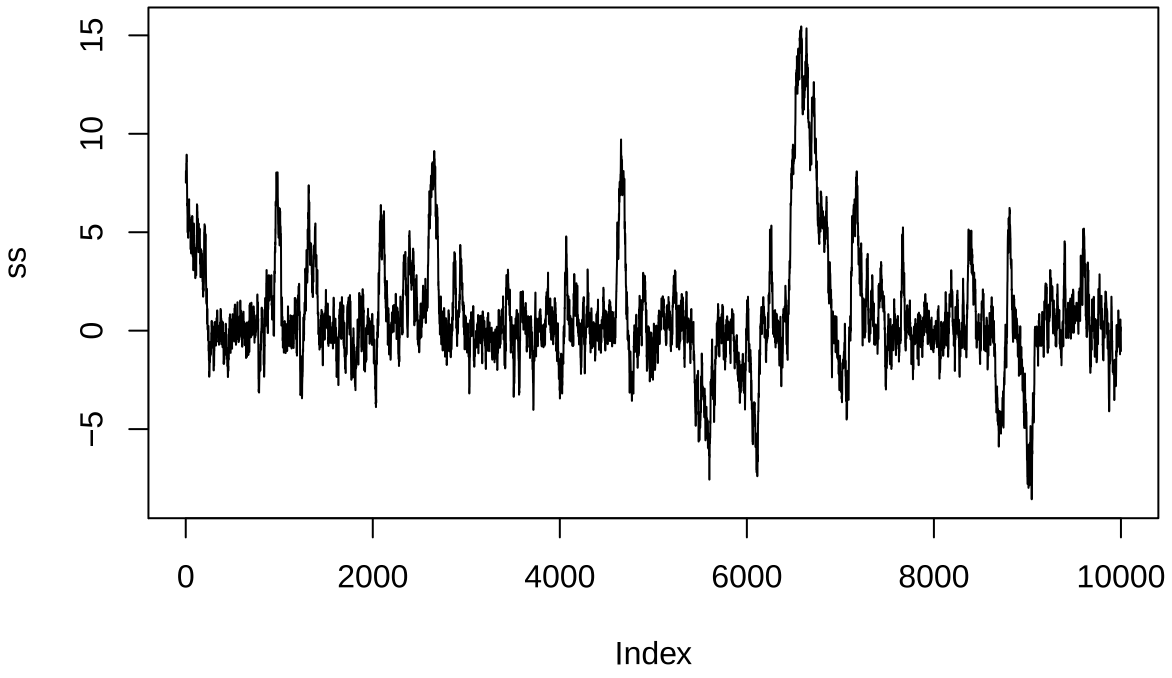
```
log.cauchy = function(x) dcauchy(x, log=TRUE)
```

b. Use the function metrop.hasting.rw to take a simulated sample of size 10,000 from the density of y. Experiment with different choices of the standard deviation C. Investigate the effect of the choice of C on the acceptance rate, and the mixing of the chain over the probability density.

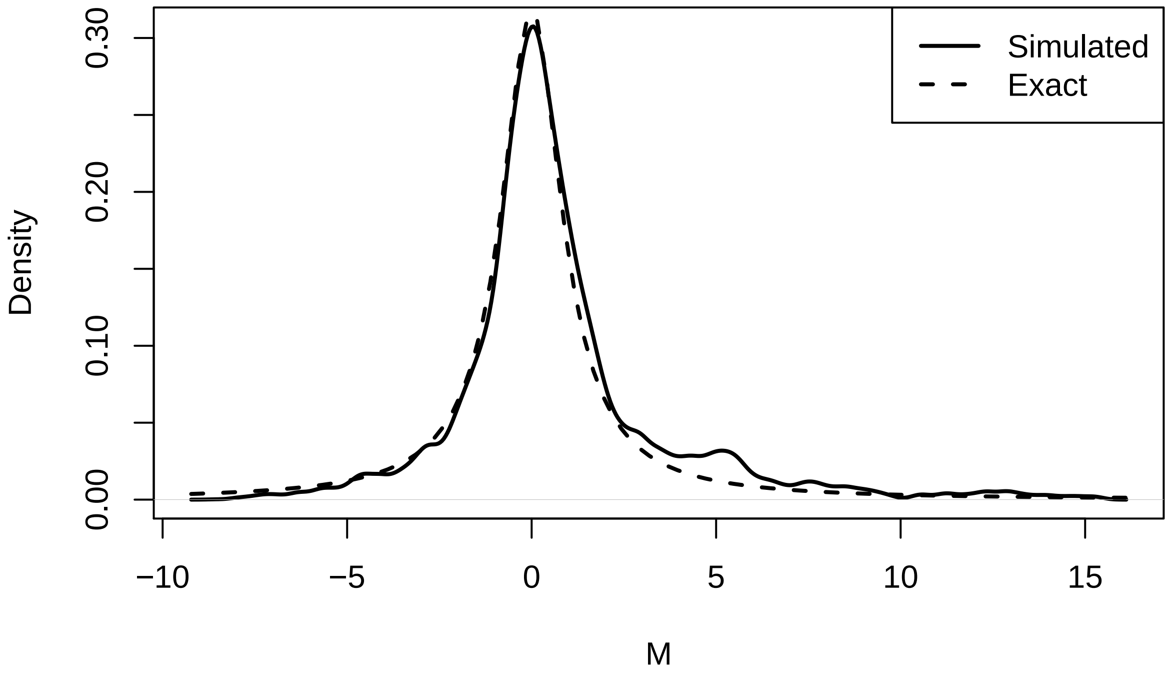
```
metrop.hasting.rw = function(logf, current, C, iter, ... ){
  S = rep(0, iter); n.accept = 0
  for(j in 1:iter){
    candidate = rnorm(1, mean=current, sd=C)
    prob = exp(logf(candidate, ... ) - logf(current, ... ))
    accept = ifelse(runif(1) < prob, "yes", "no")
    current = ifelse(accept == "yes", candidate, current)
    S[j] = current; n.accept = n.accept + (accept == "yes")
  }
  list(S=S, accept.rate=n.accept / iter)
}
xx = seq(0, 1, length.out=40)
yy = sapply(xx, function(C) metrop.hasting.rw(log.cauchy, 7, C, 10000)$accept.rate)
plot(xx, yy)
```



```r
ss = metrop.hasting.rw(log.cauchy, 7, 0.5, 10000)$S
plot(ss, type="l")
```



```r
plot(density(ss), lwd=2, main="", xlab="M")
curve(dcauchy(x), lwd=2, lty=2, add=TRUE)
legend("topright", c("Simulated", "Exact"), lty=c(1, 2),lwd=c(2, 2))
```



c. Using the simulated sample from a "good" choice of C, approximate the probability P(6 < Y < 8).

```r
sum(6 <= ss & ss <= 8)/length(ss)
```

```
## [1] 0.0233
```

13.6 (Gibbs sampling for a Poisson/gamma model).

Suppose the vec- tor of random variables (X,Y) has the joint density function f(x,y) =xa+y□1e□(1+b)xba y!Γ(a) , x > 0,y = 0,1,2,... and we wish to simulate from this joint density.

a. Show that the conditional density f(x|y) has a gamma density and identify the shape and rate parameters of this density.

$$f(x|y) \propto x^{a+y-1}e^{-(1+b)x}$$

$$\alpha = a + y, \beta = 1 + b$$

b. Show that the conditional density f(y|x) has a Poisson density.

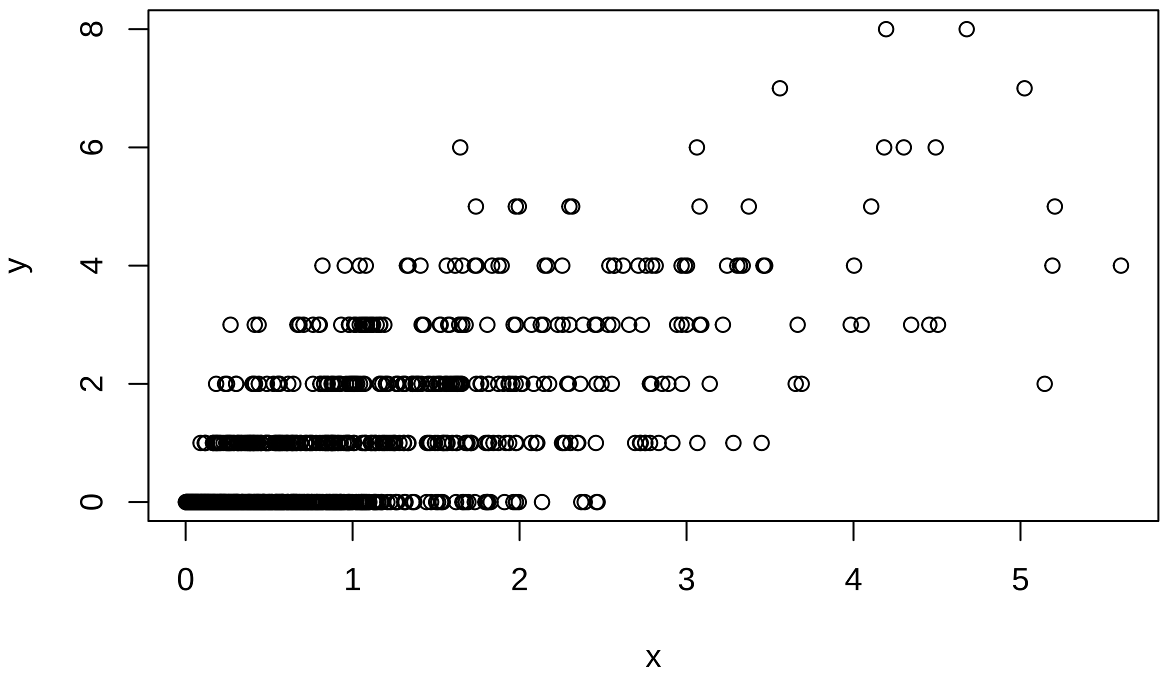
$$f(y|x) \propto \frac{x^y}{y!} = \frac{x^y}{y!}e^{-x}$$
$$\lambda = x$$

c. Write a R function to implement Gibbs sampling when the constants are given by a = 1 and b = 1.

```r
random.gibbs = function(m=1000, a=1, b=1){
  S = matrix(0, m, 2)
  dimnames(S)[[2]] = c(”x”, ”y”)
  for(j in 1:m){
    y = rpois(1, x)
    x = rgamma(1, shape=a+y, rate=1+b)
    S[j, ] = c(x, y)
  }
  return(S)
}
```

d. Using your R function, run 1000 cycles of the Gibbs sampler and from the output, display (say, by a histogram) the marginal probability mass function of Y and compute E(Y).

```r
sim.values=random.gibbs()
plot(sim.values)
```



```r
table(sim.values[ ,"y"])
```

```
##
##   0   1   2   3   4   5   6   7   8
## 554 205 117  67  39   9   5   2   2
```

```r
mean(sim.values[ ,"y"])
```

```
## [1] 0.901
```