STAT 525 — Spring 2020 — Section A1

Homework 6 Solution

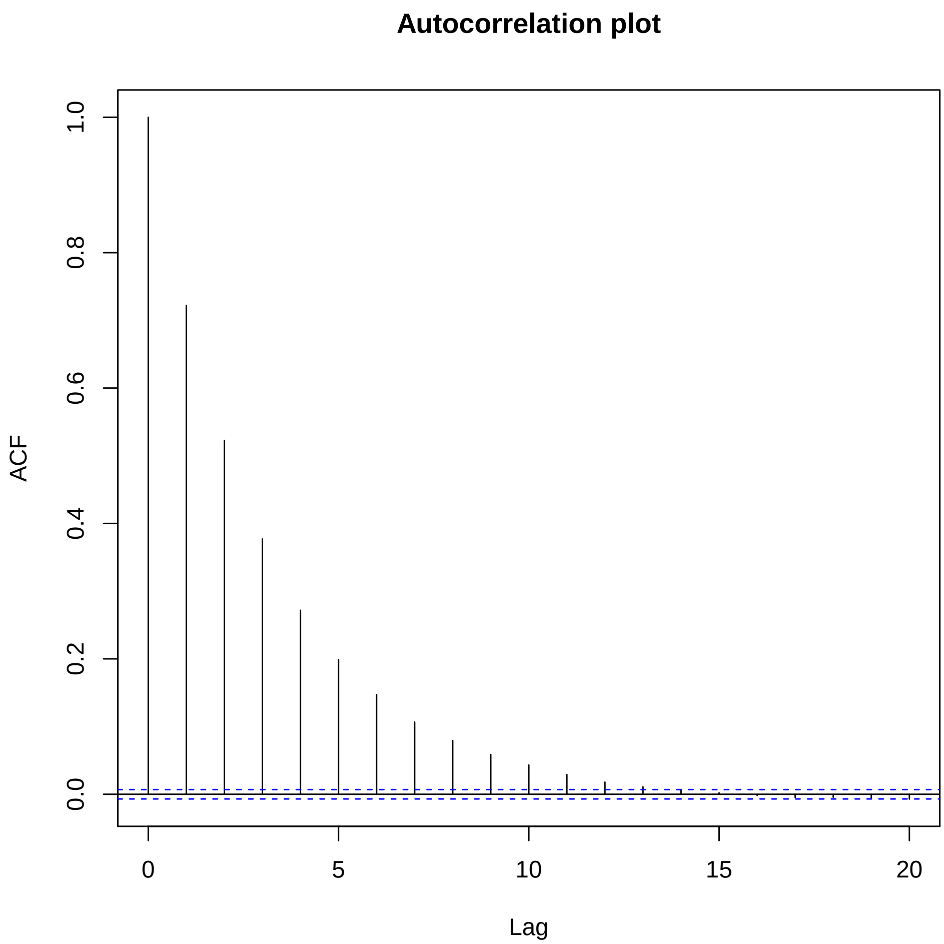
1. We use the following Metropolis-Hastings algorithm to estimate the mean of the target distribution $\pi(x)$:

   - Set initial value $x^{(0)} = 0$
   - Given current state $x^{(t)}$:
     - Draw y from the proposal distribution $y = x^{(t)} + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, 2.5^2)$
     - Calculate
$$r = \min\left\{1, \frac{\pi(y)}{\pi(x^{(t)})}\right\} = \min\left\{1, \frac{e^{-y^6-y}}{e^{-(x^{(t)})^6-x^{(t)}}}\right\}$$
     - Draw $U \sim \text{Uniform}[0, 1]$ and update
$$x^{(t+1)} = \begin{cases} y & \text{if } U \le r \\ x^{(t)} & \text{otherwise} \end{cases}$$
   - Remove the burn-in period: $x^{(0)}, \ldots, x^{(b)}$
   - Estimate $\mu = E_\pi(X)$ by
$$\hat{\mu} = \frac{1}{m-b}\left\{x^{(b+1)} + \cdots + x^{(m)}\right\}$$
   - The standard error of the estimate is
$$se(\hat{\mu}) = \sqrt{\frac{\hat{\sigma}^2}{m-b}\left[1 + 2\sum_{j=1}^{\infty}\rho_j\right]}$$

   By implementing the following code, we find that the acceptance rate is 29.50%, the estimated mean is -0.3038, and the standard error of the estimate is 0.0039.

```
m = 100000; b = m/5;
sigma = 2.5
x = rep(0,m)
n_acc = 0
for(i in 2:m){
        y = x[i-1] + rnorm(1,0,sigma)
        U = runif(1)
        r = min(1,exp(-y^6-y)/exp(-x[i-1]^6-x[i-1]))
        if(U<r) {x[i] = y; n_acc=n_acc+1}
```

```
        else {x[i] = x[i-1]}
}
acc_rate = n_acc/m
mean(x[(b+1):m])
sqrt(var(x[(b+1):m])*(sum(acf(x[(b+1):m])$acf))/(m-b))
acf(x[(b+1):m], 20, main = 'Autocorrelation plot')
```



2. Under the null hypothesis $H_0 : p_{ij} = p_{i\cdot} p_{\cdot j}$, the target distribution is

$$p(T) \propto \frac{1}{\prod_{i=1}^{12} \prod_{j=1}^{12} n_{ij}!}.$$

The exact p-value is given by

$$\mu = \sum_{T \in \Omega} I_{\{p(T) \leq p(T_0)\}} p(T) = \mathbb{E}\left[I_{\{p(T) \leq p(T_0)\}}\right].$$

We use the following Metropolis-Hastings algorithm to estimate the p-value:

- Set initial table $T^{(0)} = T_0$
- Given current table $T^{(t)}$:
  - Calculate $h(T^{(t)}) = I_{\{p(T^{(t)}) \le p(T_0)\}}$
  - Pick two twos and columns uniformly at random, and add or subtract the elements in the four entries according to the following

$$\begin{pmatrix} + & - \\ - & + \end{pmatrix} \quad \text{or} \begin{pmatrix} - & + \\ + & - \end{pmatrix}$$

    and get a new table T^{new}
  - Calculate

$$r = \min\left\{1, \frac{p(T^{\text{new}})}{p(T^{(t)})}\right\}$$

  - Draw $U \sim \text{Uniform}[0,1]$ and update

$$T^{(t+1)} = \begin{cases} T^{\text{new}} & \text{if } U \le r \\ T^{(t)} & \text{otherwise} \end{cases}$$

- Remove the burn-in period: $T^{(0)}, \ldots, T^{(b)}$
- Estimate the p-value by

$$\hat{p} = \frac{1}{m-b}\left\{h(T^{(b+1)}) + \cdots + h(T^{(m)})\right\}$$

- The standard error of the estimate is

$$se(\hat{p}) = \sqrt{\frac{\hat{\sigma}^2}{m-b}\left[1 + 2\sum_{j=1}^{\infty} \rho_j\right]}$$

By implementing the following R code, we find that the acceptance rate is 12.59%, the estimated p-value is 0.8215, and the standard error is 0.0081.

```
N = 100000;
T0 = matrix(c(1,0,0,0,1,2,0,0,1,0,1,0,
              1,0,0,1,0,0,0,0,0,1,0,2,
              1,0,0,0,2,1,0,0,0,0,0,1,
              3,0,2,0,0,0,1,0,1,3,1,1,
              2,1,1,1,1,1,1,1,1,1,1,0,
              2,0,0,0,1,0,0,0,0,0,0,0,
              2,0,2,1,0,0,0,0,1,1,1,2,
              0,0,0,3,0,0,1,0,0,1,0,2,
              0,0,0,1,1,0,0,0,0,0,1,0,
              1,1,0,2,0,0,1,0,0,1,1,0,
```

```
                                  0,1,1,1,2,0,0,2,0,1,1,0,
                                  0,1,1,0,0,0,1,0,0,0,0,0), 12, 12, byrow = TRUE
ind = rep(0, N)
p_T0 = 1/prod(factorial(T0))
T = T0; i = 1; n = 1
m1 = matrix(c(1,-1,-1,1), 2, 2)
m2 = matrix(c(-1,1,1,-1), 2, 2)
while(i < N){
      a = sample(12, 2)
      b = sample(12, 2)
      tmp = T
      h = rbinom(1, 1, 0.5)
      if (h == 1) {tmp[a,b] = tmp[a,b] + m1}
      else {tmp[a,b] = tmp[a,b] + m2}
      if (sum(tmp[a,b] < 0) == 0){
            r = min(1, (1/prod(factorial(tmp[a,b]))) / (1/prod(factorial(T
            u = runif(1)
            if (u < r) {T=tmp; n=n+1}
      }
      ind[i] = as.numeric(p_T0 >= (1/prod(factorial(T))))
      i = i + 1
}
acc_rate = n/N
mean(ind[(N/5+1):N])
sqrt(var(ind[(N/5+1):N])*(sum(acf(ind[(N/5+1):N])$acf))/(N-N/5))
```