STAT525 HW6

Yuxuan Zhang

3/18/2020

1

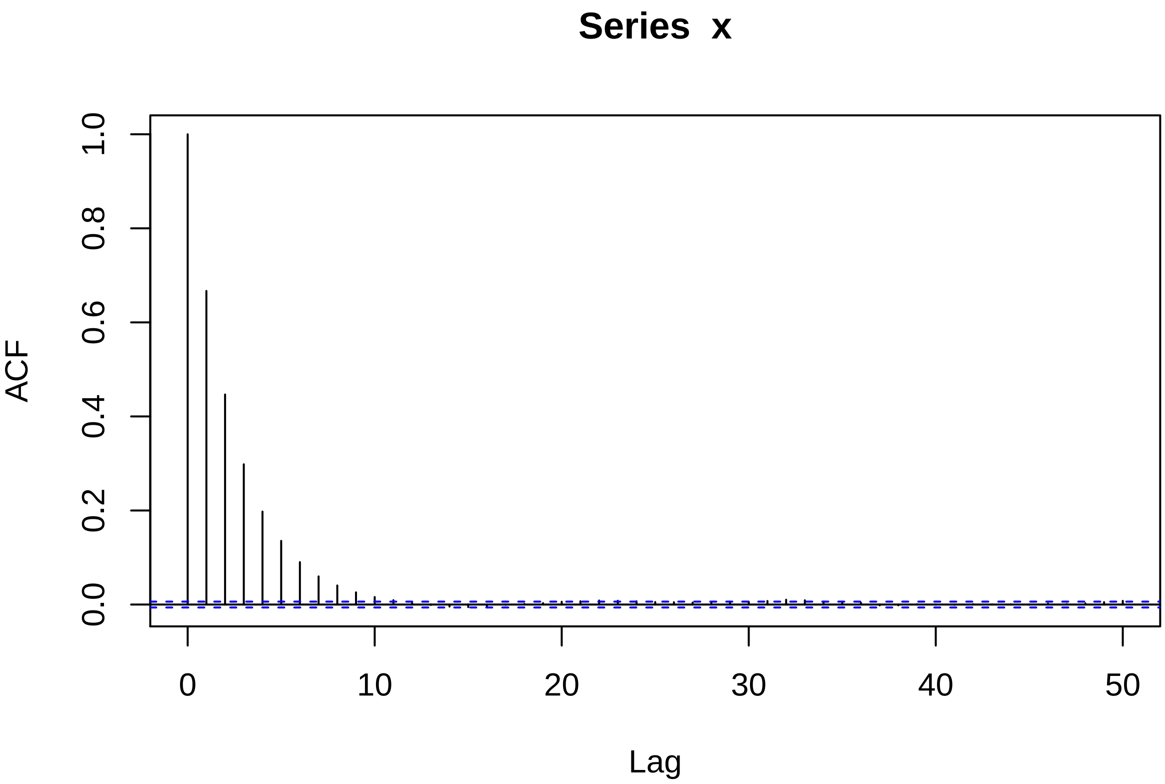
Design a Metropolis-Hastings algorithm to simulate samples from a distribution with the following density: $\pi(x) \propto e^{-x^6-x}, -\infty < x < \infty$.

Estimate the acceptance rate of your algorithm. Show the autocorrelation plot of your samples. Estimate the mean of the above density function and give the standard error of your estimate.Attach your code and results.

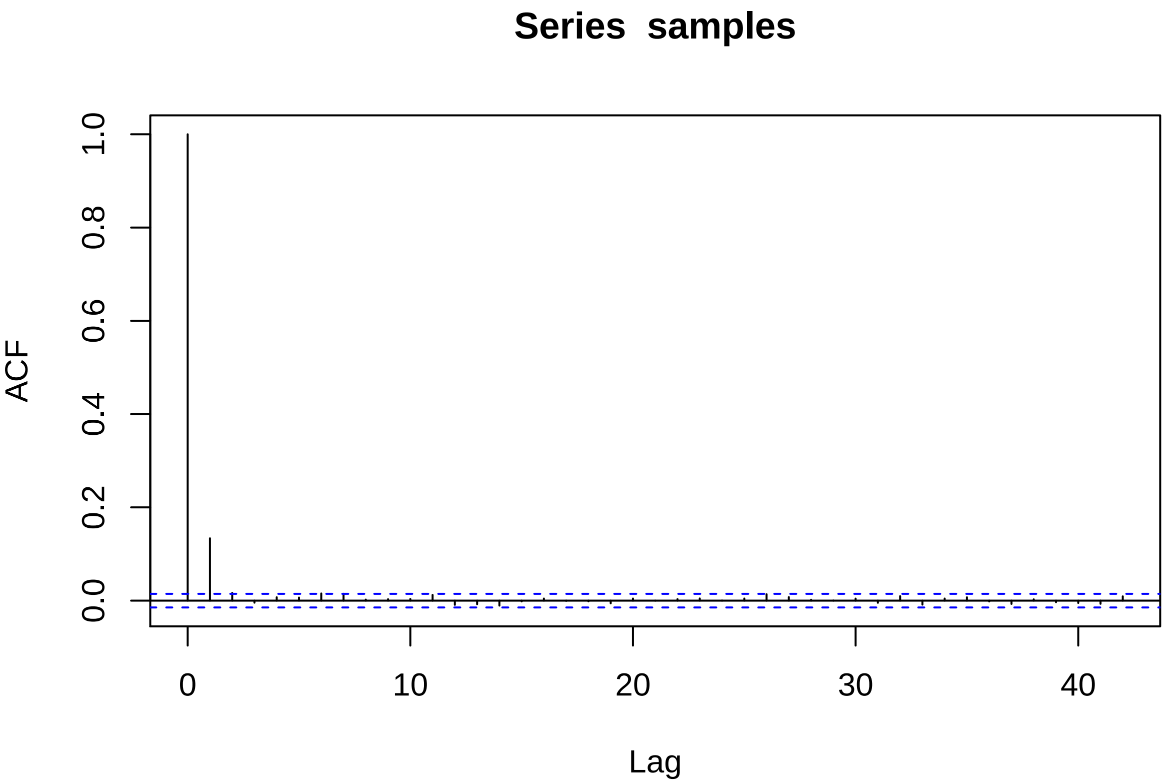
```
pi = function(x) {
  return (exp(-x^6-x))
}
m = 100000
x = c(rep(0,m))
x[1] = rnorm(1)
n = 0
for (i in 2:m) {
  y = rnorm(1, mean = x[i-1], sd = 1.2)
  r = min(1, pi(y)/pi(x[i-1]))
  u = runif(1)
  if (u <= r) {
    x[i] = y
    n = n + 1
  } else {
    x[i] = x[i-1]
  }
}
# Acceptance Rate
n/m
```

```
## [1] 0.44289
```

```
# Autocorrelation Plot
acf(x)
```



```
# m is the total length
burnin_period = round(0.1*m)
x_burn = x[burnin_period:m]
# draw every kth sample from x
samples = x_burn[seq(1,length(x_burn),5)]
acf(samples)
```



```r
# mean of density function
mean(samples)
```

```
## [1] -0.3069715
```

```r
# standard error of the estimate
sqrt(var(samples)/length(samples))
```

```
## [1] 0.003893409
```

2

Implement the exact test using the Metropolis-Hastings algorithm to test the null hypothesis that birthday and deathday are independent. One way to define p-value for exact test is

$$p - value = \sum_{T\in\Omega} 1_{\{p(T)\leq p(T_0)\}} p(T)$$

where T_0 is the observed table, and Ω is the set of 12*12 contingency tables with the same row sums and column sums as T_0. Here p(T) is definded as

$$p(T) \propto \frac{1}{\prod_{i=1}^{12}\prod_{i=1}^{12} t_{ij}!}, T \in \Omega$$

```r
library(data.table)

pi = function(t){
  product = 1
  for (i in 1:2) {
    for (j in 1:2) {
      if (t[i,j] < 0) {
        return (0)
      } else {
        product = product * factorial(t[i,j])
      }
    }
  }
  return (1/product)
}

p_T = function(t){
  # input table is 12*12
  return (exp(-1*sum(log(factorial(t)))))
}

table = read.table("table.txt", header = TRUE)
n = 0
m = 10000
#indicator = c(rep(0, m))
#indicator[1] = 1
p = c(rep(0, m))
p[1] = 0.3
p0 = p_T(table[1:12,1:12])
for (i in 2:m){
  random = sample.int(2,1)
  table_new = table
  r = sort(sample.int(12,2))
  c = sort(sample.int(12,2))
  if (random == 1){
    table_new[r[1],c[1]] = table_new[r[1],c[1]] + 1
    table_new[r[1],c[2]] = table_new[r[1],c[2]] - 1
    table_new[r[2],c[1]] = table_new[r[2],c[1]] - 1
    table_new[r[2],c[2]] = table_new[r[2],c[2]] + 1
  } else {
    table_new[r[1],c[1]] = table_new[r[1],c[1]] - 1
    table_new[r[1],c[2]] = table_new[r[1],c[2]] + 1
    table_new[r[2],c[1]] = table_new[r[2],c[1]] + 1
    table_new[r[2],c[2]] = table_new[r[2],c[2]] - 1
  }
  u = runif(1)
  if (pi(table_new[r,c]) != 0 & u <= min(1, pi(table_new[r,c])/pi(table[r,c]))) {
    p[i] = p_T(table_new[1:12,1:12])
    table = table_new
    n = n + 1
  } else {
    p[i] = p[i-1]
  }
}
# Estimate of the p-value
indicator = p <= p0
mean(indicator[1001:m])
```

```
## [1] 0.8962222
```

```r
# Standard error of the estimate
z = indicator[1001:m]
rho = acf(as.numeric(z), plot=FALSE)$acf[-1]
std = sqrt(var(z)*(1+2*sum(rho))/(m-1000))
std
```

```
## [1] 0.02490214
```