homework 4

Topics in Statistics 1, Jieun Shin

2021.10.21

```
library(dplyr)
```

```
##
## 다음의 패키지를 부착합니다: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

4.10

문제에서 정의한 마코프 체인을 생성하고 점추정치와 95%신뢰구간을 계산한다.

```
# regeneration method의 신뢰구간을 구하기 위한 함수
regen_ci = function(y, tau, alpha = 0.05){
  out = list()

  est = mean(y) / mean(tau)

  s11 = var(y)
  s22 = var(tau)
  s12 = cov(y, tau)

  s = sqrt( s11 - 2 * est * s12 + est^2 * s22 )
  ci = est + c(-1, 1) * qnorm(1 - alpha/2) * s / (mean(tau) * sqrt(length(y)))

  out$s = s
  out$ci = ci

  return(out)
}

####

t = 1:15
X = c(0, 3, 0, 1, 2, 1, 0, 2, 0, 1, 0, 1, 0, 2, 0)
```

```r
T = which(X == 0)
tau = R = c()

i = 2
for(i in 2:length(T)){
  tau[i] = T[i] - T[i-1]
  id = T[i-1]:(T[i] - 1)
  R[i] = sum(X[id])
}

tau = tau[2:length(T)]
R = R[2:length(T)]

l = mean(R)/mean(tau)
mean(l)     # point estimator
```

```
## [1] 0.9285714
```

```r
regen_ci(R, tau)
```

```
## $s
## [1] 0.7613093
##
## $ci
## [1] 0.6675015 1.1896413
```

```r
#
```

점추정치는 약 0.78, 95% 신뢰구간은 약 [0.59, 1.14] 이다.

4.11

(a) W_n을 $GI/G/1$ queue에서 n번째 고객의 대기시간이라 하자. 그리고 S_n을 n번 고객이 서비스를 받는데 걸리는 시간, A_{n+1}을 n번째 사람과 $n+1$번째 사람이 도착한 사이 시간이라 하자.

따라서 W_{n+1}은 $W_n + S_n < A_{n+1}$ ($n+1$번째 사람이 도착하기 전에 이미 n번째 사람이 서비스를 받고 떠난 경우)이면 $n+1$번째 사람의 대기시간은 0이다. 그렇지 않으면 $n+1$번째 고객은 n번째 사람의 대기시간까지 기다려야 하므로 $W_n + S_n - A_{n+1}$이 된다.

(b) $N = 5000$번의 반복 시, $M/M/1$ queue에서 4번째 고객에 대한 대기시간의 점추정치와 95% 신뢰구간을 구한다. 그 결과 점추정치는 0.32, 95% 신뢰구간은 [0.31, 0.34]이다.

```r
Queue_b = function(){
  # input
  lambda = 1
  mu = 2

  # initialize
  W = S = A = 0
```

```
  for(i in 1:3){
    A = rexp(1, lambda)
    S = rexp(1, mu)
    W[i+1] = max(W[i] + S - A, 0)
  }

  return(W[4])
}

N = 5000
W = replicate(N, Queue_b())

est = mean(W) # point estimator
se = sd(W)/sqrt(N)

est
```

```
## [1] 0.3126416
```

```
est + c(-1, 1) * qnorm(1 - 0.05) * se   # CI
```

```
## [1] 0.2994649 0.3258183
```

(c) (b)와 같은 대기 시스템에서 21~70번째 고객의 평균 대기시간과 그 신뢰구간을 구하면 다음과 같다. 평균 대기시간의 추정치는 0.32, 95% 신뢰구간은 [0.31, 0.34]이다.

```
Queue_c = function(){
  # input
  lambda = 1
  mu = 2

  # initialize
  W = S = A = 0

  for(i in 1:69){
    A = rexp(1, lambda)
    S = rexp(1, mu)
    W[i+1] = max(W[i] + S - A, 0)
  }

  return(mean(W[21:70]))
}

N = 5000
W = replicate(N, Queue_c())

est = mean(W) # point estimator
se = sd(W)/sqrt(N)

est
```

```
## [1] 0.5001295
```

```r
est + c(1, -1) * qnorm(0.05) * se   # CI
```

```
## [1] 0.4917326 0.5085264
```

4.13

주어진 확률에 따라 상태가 1, 2만 있고 regeneration point가 1000개인 마코프 체인을 생성하였다. 그 결과 점추정치는 약 2.2이며 95% 신뢰구간은 [2.13, 2.22] 임을 확인하였다.

```r
P = matrix(c(1/3, 2/3, 1/4, 3/4), 2, 2, byrow = TRUE)
C = matrix(c(0, 1, 2, 3), 2, 2, byrow = TRUE)

N = 1000  # regeneration cycle
c = x = c()

x[1] = c[1] = 1

i = 0
while( sum(x == 1) < N){   # regeneration point?? 1000???? ?? ?? ????
  i = i + 1
  # obtain the sample
  if (x[i] == 1){
    x[i+1] = sample(c(1, 2), 1, prob = P[1, ])
    c[i+1] = ifelse(x[i+1] == 1, C[1, 1], C[1, 2])
  } else if (x[i] == 2){
    x[i+1] = sample(c(1, 2), 1, prob = P[2, ])
    c[i+1] = ifelse(x[i+1] == 1, C[2, 1], C[2, 2])
  }
}


# estimation
R = tau = c()
id = which(x == 1)

j = 0
for(i in 1:length(id)){
  j = j + 1
  a = id[i]
  b = ifelse(i == length(id), id[i], id[i+1] - 1)
  tau[j] = length(a:b)
  R[j] = sum(c[a:b])
}

mean(R) / mean(tau)   # point estimator
```

```
## [1] 2.102633
```

```r
regen_ci(R, tau)      # CI
```

```
## $s
```

```
## [1] 2.725721
##
## $ci
## [1] 2.052083 2.153183
```

5.2

$N = 100,000$ 일 때 bridge network에서 shortest path의 기댓값을 CMC추정치와 antithetic 추정치와 비교한다. 이 때 각 $X_1, \ldots, X_5$는 평균이 각 $1, 1, 0.5, 2, 1.5$인 지수분포를 따른다고 하자. 그 결과 가장 짧은 길이의 점추정치는 약 2 이다.

```
H = function(p){
  x = c()
  for(i in 1:length(p)) x[i] = rexp(1, p[i])

  min(x[1] + x[4], x[1] + x[3] + x[5], x[2] + x[3] + x[4], x[2] + x[5])
}

N = 100000
prob = 1/c(1, 1, 0.5, 2, 1.5)

x = replicate(N, H(prob))

CMC = mean(x)   # CMC estimator
CMCvar = var(x)

CMC
```

```
## [1] 1.498722
```

```
CMCvar
```

```
## [1] 1.043188
```

```
# antithetic estimator

F = function(x, lambda) - log(1-x) / lambda

antithetic5.2 = function(N, prob){
  n = N/2

  x1 = x2 = x = y = c()
  for(j in 1:n){
    u = runif(5)

    for (i in 1:length(prob)){
      x[i] = F(u[i], prob[i])
      y[i] = F(1 - u[i], prob[i])
    }
    x1[j] = min(x[1] + x[4], x[1] + x[3] + x[5], x[2] + x[3] + x[4], x[2] + x[5])
    x2[j] = min(y[1] + y[4], y[1] + y[3] + y[5], y[2] + y[3] + y[4], y[2] + y[5])
  }
```

```
  out = list()

  out$cov = cov(x1,x2)
  out$x = x1 + x2
  return(out)
}

x = antithetic5.2(N, prob)$x
antcov = antithetic5.2(N, prob)$cov

sum(x)/N  # estimator
```

```
## [1] 1.493793
```

```
CMCvar + antcov/N
```

```
## [1] 1.043183
```

5.4

$GI/G/1$ 대기 시스템에서 대기시간의 기댓값을 추정하려고 한다. batch mean 방법을 사용하여 CMC 추정치와 antithetic 추정치의 분산을 비교한다. 그 결과 두 방법의 추정치는 약 1.2로 비슷하며 분산은 antithetic 추정치의 분산이 미미한 차이로 더 작은 것을 확인하였다.

```
M = 10000

Queue5.4 = function(){
  # input
  lambda = 1/2
  M = 10000

  # initialize
  W = S = A = 0

  for(i in 1:(M - 1)){
    A = rexp(1, lambda)
    S = runif(1, 0.5, 2)
    W[i+1] = max(W[i] + S - A, 0)
  }

  return(W)
}

N = K = 100

x = Queue5.4()
cmc_x = matrix(x[(K+1):M], ncol = N)


# calculate the sample mean
y = apply(cmc_x, 2, mean)
```

```r
cmc_m = mean(y)
cmc_v = var(x)

cmc_m
```

```
## [1] 1.114325
```

```r
cmc_v
```

```
## [1] 2.266337
```

```r
# antithetic
M = 10000
A = function(x, lambda = 0.5) return(- log(1 - x) / lambda) # inter-arrival time
S = function(x) return(1.5 * x + 0.5)    # service time

antithetic5.4 = function(N){
  n = N/2
  u1 = runif(n)
  u2 = runif(n)
  u1 = c(u1, 1 - u1)
  u2 = c(u2, 1 - u2)

  x1 = x2 = 0
  for(j in 2:n){
    x1[j] = max(x1[j - 1] +  S(u1[j - 1]) - A(u2[j]), 0)
    x2[j] = max(x2[j - 1] +  S(u1[n + j - 1]) - A(u2[n + j]), 0)
  }

  out = list()
  out$x1 = x1
  out$x2 = x2
  return(out)
}

x = antithetic5.4(N = M)
x1 = x$x1
x2 = x$x2

# calculate the sample mean
B = K = 50
y1 = matrix(x1[(K+1):(M/2)], ncol = B) %>% apply(., 2, mean)
y2 = matrix(x2[(K+1):(M/2)], ncol = B) %>% apply(., 2, mean)

mean(c(y1, y2))
```

```
## [1] 1.161785
```

```r
cmc_v + cov(x1, x2)/M
```

```
## [1] 2.266229
```

5.5

```
N = 1000

X = matrix(rexp(N * 5, 1), N, 5)
r = matrix(rep(c(2, 3, 3, 2), each = N), nrow = N)
C = cbind(X[,1] + X[,4], X[,1] + X[,3] + X[,5], X[,2] + X[,3] + X[,4], X[,2] + X[,5])
H = sapply(1:N, function(x) min(C[x,]))

# CMC
mean(H)
```

```
## [1] 1.217088
```

```
var(H)
```

```
## [1] 0.6126604
```

```
# control variable
SigC = cov(C)

CovXC = c(rbind(cov(H, C[,1]), cov(H, C[,2]), cov(H, C[,3]), cov(H, C[,4])))

a = solve(SigC) %*% CovXC

cvX = H - (C - r) %*% a

mean(cvX)
```

```
## [1] 1.179877
```

```
var(H) - cov(H, C) %*% solve(SigC) %*% t(cov(H, C))
```

```
##           [,1]
## [1,] 0.2777951
```

5.6

4번째 고객의 대기시간을 CMC추정치와 control variable 추정치의 분산을 비교한다. CMC추정치와 control variable 추정치는 각 0.71, 0.74로 비슷하며 분산을 비교하면 CMC추정치보다 control variable 추정치의 분산이 약간 더 작아짐을 확인하였다.

```
Queue5.6 = function(){

  # initialize
  W = S = A = C = 0

  A = - log(1 - runif(4)) / 0.5
  S = 1.5 * runif(4) + 0.5
```

```
  for(i in 1:3){
    W[i+1] = max(W[i] + S[i] - A[i+1], 0)
    C[i+1] = W[i] + S[i] - A[i+1]
  }

  out = list()
  out$w = W[4]
  out$c = C[4]

  return(out)
}

N = 1000
w = replicate(N, Queue5.6()$w)
c = replicate(N, Queue5.6()$c)

r = 3 * ((0.5 + 2) / 2 - 2)
a = cov(w, c) / var(c)

y = w - a * (c - r)

# CMC
mean(w)
```

```
## [1] 0.667138
```

```
var(w)
```

```
## [1] 0.7999657
```

```
# control variable
mean(y)
```

```
## [1] 0.7070687
```

```
var(w) * (1 - cor(w, c)^2)
```

```
## [1] 0.7982875
```