2021년 1학기 통계계산 기말고사

2021-05-27

제출자 성명: 지성민

학번: 2021710322

1번

In [1]:

```
## 1)
game <- function(salary){
  i <- 1
  s <- salary
  dice <- c(1:6)
  while(s < 1000000){
    i <- i + 1
    diced <- sample(dice, 1)
    if(diced <= 4){
      s <- salary[i-1]*i
      salary <- c(salary,s)
    }else{
      s <- salary[i-1]*1
      salary <- c(salary,s)
    }
  }
  return(list(salary = salary, total = sum(salary), num = i, last = s))
}
```

In [2]:

```
set.seed(500)
salary <- game(1)$salary

salary
length(salary)
```

1 · 2 · 2 · 8 · 40 · 240 · 1680 · 1680 · 15120 · 151200 · 1663200

11

나통계씨는 i =(2, 4, 5, 6, 7, 9, 10, 11) 번째 달에 월급이 이전달보다 i배 만큼 상승하였으며, 나머지 달에는 이전달만큼의 월급을 받았다.

나통계씨는 총 11달을 근무하고 퇴사하였으며, 마지막 달에는 1,663,200 원의 급여를 받았다.

In [3]:

```
## 2)
set.seed(500)

number <- vector("numeric", 100)
totsal <- vector("numeric", 100)
for(i in 1:100){
  gamed <- game(1)
  totsal[i] <- gamed$total
  number[i] <- gamed$num
}
```

In [4]:

```
### mean of total salary
mean(totsal)
```

5421277.78

In [5]:

```
### mean of working period
mean(number)
```

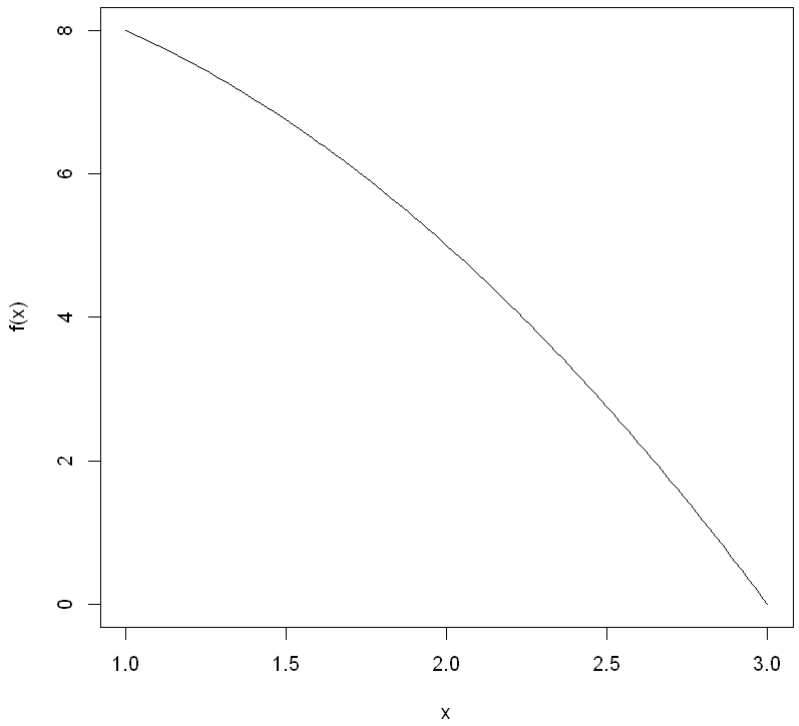
12.86

총 100번의 시뮬레이션 결과 나통계씨는 평균 5421277.78 원의 총 월급을 받았으며, 12.86달의 근무기간을 가졌다.

2번

In [6]:

```
## 1)
f <- function(x){
    9 - x^2
}
curve(f, from = 1, to  = 3)
```



이차원 평면상에서 f(x)의 최솟값은 0이며, 최댓값은 8이다.

In [7]:

```
## 2)
set.seed(600)
hom.mc <- function(n = 10000){
  x <- runif(n, 1, 3)
  y <- runif(n) * 8 ## maximum value

  # size of the squared region = 2 * 8
  p <- 2 * 8 * mean(y <= f(x))
  return(p)
}
hom.mc(10000)
```

9.4896

Hit or Miss 방법으로 적분값을 추정하였다.

이차원 평면상의 적분 영역에 해당하는 구간을 가로로 최댓값을 세로로 하는 면적이 16 이다.

따라서 이 면적을 단위로 균등분포를 따르는 난수 y가

균등분포를 따르는 난수 x에 대한 $f(x)$보다 작을 확률을 이용하여 적분을 한다.

그 결과 적분추정값은 9.4896 으로 계산되었다.

In [8]:

```
## 3)
set.seed(600)
sm.mc <- function(n = 10000){
  x <- runif(n, 1, 3)
  theta <- 2 * mean(f(x))
  return(theta)
}

round(sm.mc(10000), 4)
```

9.3856

Sample mean MC 방법으로 적분값을 추정하였다.

난수를 추출한 다음 난수 값에 대한 함숫값의 평균을 구한다.

난수를 추출한 균등분포의 pdf가 $1/2$ 이므로 마지막에 그의 역수인 2 를 곱해준다.

그 결과 적분추정값은 9.3856 으로 계산되었다.

In [9]:

```
## 4)
set.seed(600)
anti.mc <- function(n = 10000, anti = TRUE){
  u <- runif(n/2, 1, 3)

    ## check antithetic
  if(!anti) v <- runif(n/2, 1, 3) else{
   ## 상보되는 값
   v <- 4-u
  }

  u <- c(u, v)
  theta <- 2 * mean(f(u))
  return(theta)
}

round(anti.mc(10000), 4)
```

9.3372

위의 방법은 antithetic 방법을 적용할 수 있는 코드이다.

만약 anti = TRUE 이라면 난수 숫자 중 절반만 난수를 추출하고, 나머지는 그에 상보되는 값 여기서는 $4 -$ 추출된난수로 저장한다.

그 다음 몬테칼로 적분을 진행한다. 이때 난수를 추출한 균등분포의 pdf가 $1/2$ 이므로 마지막에 그의 역수인 2 를 곱해준다.

만약 anti = TRUE 이라면 4) 의 코드는 3) 의 코드와 동일한 방법이다.

In [10]:

```
### Simulating to compare antithetic MC and sample mean MC
n <- 10000
r <- 100

MC1 <- MC2 <- numeric(r)

set.seed(800)
for(i in 1:r){
  MC1[i] <- anti.mc(n, anti = FALSE)
  MC2[i] <- anti.mc(n, anti = TRUE)
}

round((var(MC2)) / var(MC1), 4)
```

0.0272

MC1 벡터에는 3) 의 방법을 적용하고, MC2 벡터에는 4) 의 방법을 적용하여 100번 시뮬레이션 한 결과를 저장했다.

그리고 시뮬레이션 한 결과의 분산을 비교하였다.

그 결과 MC2, 즉 antithetic 방법으로 추정한 적분값들의 분산이

MC1, 즉 simple MC 방법으로 추정한 적분값들의 분산의 2.72% 만큼의 크기를 가졌다.

이러한 antithetic 방법의 분산 감소는 서로 의존되는 난수 사이에 발생하는 공분산에 의한 효과이다.

3번

$$g(x) = \prod_{i=1}^{5} \frac{1}{\pi(1 + (x - a_i)^2)}, \quad a = (1, 2, 3, 7, 8)$$

문제에는 $g(y)$ 로 나타냈으나, 여기서는 편의상 표기를 y 대신 x 로 사용한다.

In [11]:

```
## 1)
g <- function(x){
  g.i <- function(x, a_i){
    1 / (pi*(1 + (x - a_i)^2))
  }
  a <- c(1,2,3,7,8)
  prod(g.i(x, a))
}
```

주어진 분포의 밀도함수를 plot 에 나타내보자.

먼저 이 분포는 Cauchy 분포의 support와 동일한 영역을 가질 수 있다. 따라서 support는 $(-\infty, \infty)$ 이다.

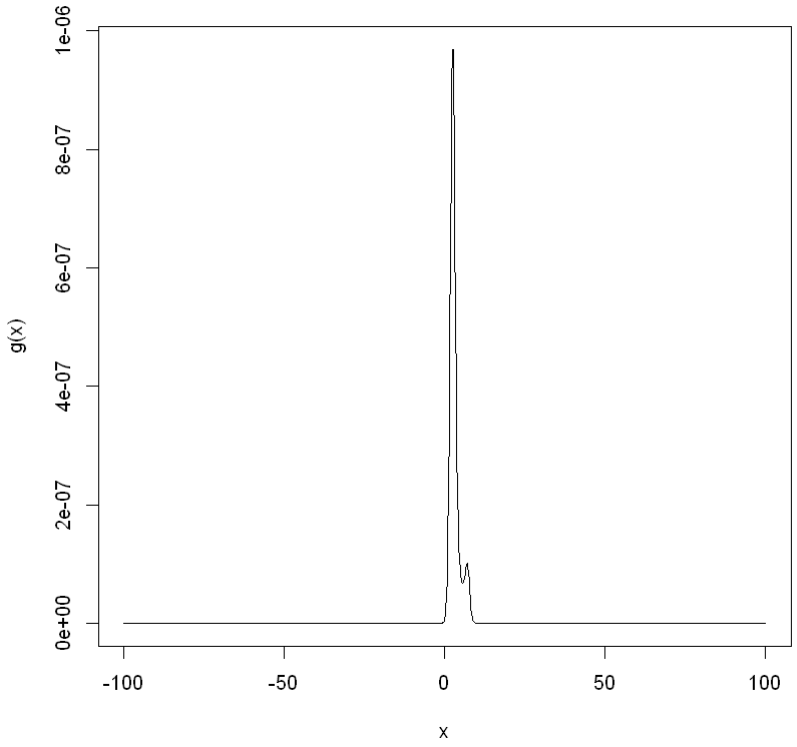
그러나 plot에는 무한한 support를 나타낼 수 없다. 그러므로 $[-100, 100]$ 의 값만 나타냈다.

따라서 -100~100 사이의 함숫값을 1000개 계산한 후 꺾은선 plot을 생성하였다.

In [12]:

```
x <- seq(-100, 100, length = 1000)
y <- vector("numeric", 1000)

for(i in 1:length(x)){
  y[i] <- g(x[i])
}
plot(x, y, type = "l", ylab = "g(x)", xlab = "x")
```



In [13]:

```
## 2)

MCMC.chain <- function(v, N, x0){
  x <- vector("numeric", N)
  x[1] <- x0
  k <- 0

  # MC chain
  for(i in 2:N){
    xt <- x[i-1]
    y <- rnorm(1, mean = xt, sd = sqrt(v))

    r <- g(y) / g(xt)

      # acceptance-rejection
    if(runif(1, 0, 1) <= r) x[i] <- y else{
      x[i] <- xt
      k <- k+1
    }
  }
  return(list(x = x, accept = N-k))
}
```

Random walk Metropolis 방법을 이용하여 MCMC 를 수행하는 코드를 작성하였다.

proposal distribution 은 $Normal(x_t, v)$ 이며, x_t 는 t 번째 MCMC 반복에서 accepted 된 값이다.

함수의 지역변수 중 r은 proposal distribution 에 의한 난수 추출 값 y 와 이전 accepted 값 x_t을 넣은 density function 의 비이다.

이 비를 이용하여 acceptance-reject을 진행한다. 만약, 균등분포$(0, 1)$로 추출된 난수값이 r 보다 작다면 새로운 y 값을 accept 한다.

In [14]:

```
## 3)

V <- c(0.05, 0.5, 2, 16)
N <- 10000
x0 <- 0
set.seed(1000)
rw1 <- MCMC.chain(v = V[1], N = N, x0 = x0)
rw2 <- MCMC.chain(v = V[2], N = N, x0 = x0)
rw3 <- MCMC.chain(v = V[3], N = N, x0 = x0)
rw4 <- MCMC.chain(v = V[4], N = N, x0 = x0)
```

proposal distribution 의 분산인 v값을 다르게 하여 추출되는 MCMC 를 실험해보자.

초기값 x_0 는 모두 0으로 동일하게, 시뮬레이션 수는 10000회로 하고

v 에는 순서대로 $0.05, 0.5, 2, 16$ 의 값을 넣어 실험하였다.

In [15]:

```
rw1$accept / N
rw2$accept / N
rw3$accept / N
rw4$accept / N
```

0.9236

0.7691

0.6028

0.334

위에서 아래 순서대로 $v = (0.05, 0.5, 2, 16)$ 에 따라 accept 된 비율이 감소하였다.

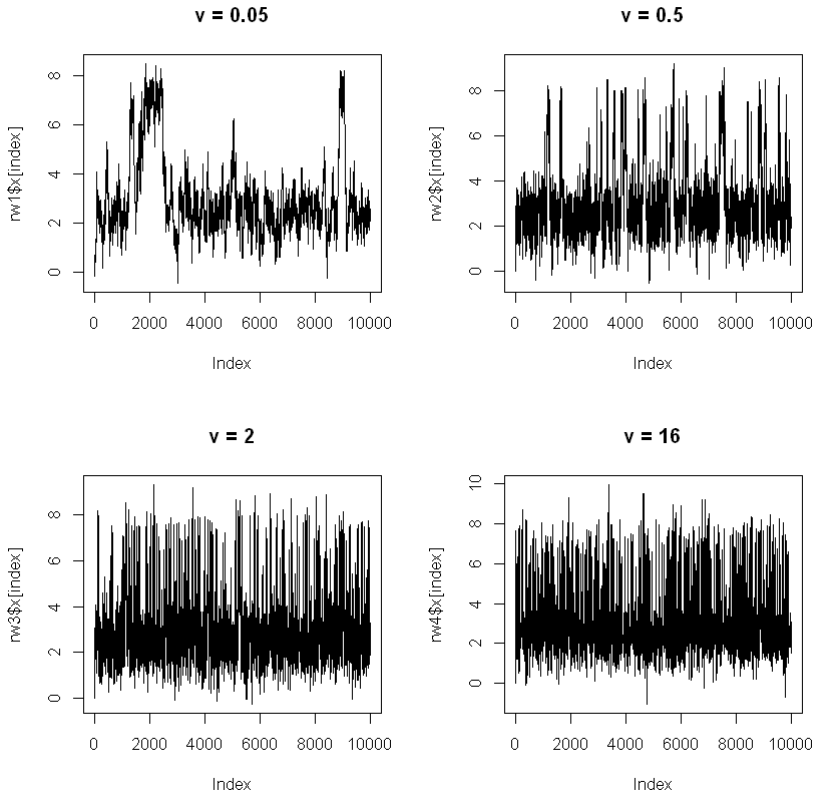
따라서 v가 커질 수록 accept 비율이 감소함을 추측할 수 있다.

In [16]:

```
index <- 1:10000

par(mfrow = c(2,2))

plot(rw1$x[index], type = "l", main = "v = 0.05")
plot(rw2$x[index], type = "l", main = "v = 0.5")
plot(rw3$x[index], type = "l", main = "v = 2")
plot(rw4$x[index], type = "l", main = "v = 16")
```



각 MCMC 결과에서 나온 Chain 들을 비교한 결과 $v = 0.05$ 에서 추출이 불안정했다.

즉, 10000 번의 MCMC 반복으로는 Markov chain의 수렴이 되지 않음을 나타냈다.

반면 $v = 2$ 나 $v = 16$ 에서는 초기의 1000회 반복 후에 비교적 안정화된 추출 결과가 나타났다.

In [17]:

```
a <- c(.05, seq(.1, .9, .1), .95)

rw <- cbind(rw1$x, rw2$x, rw3$x, rw4$x)
mc <- rw[2001:N, ]
Qrw <- apply(mc, 2, function(x) quantile(x, a))

result <- round(cbind(Qrw), 3)
colnames(result) <- c("v=0.05","v=0.5","v=2","v=16")

print(result)
```

```
    v=0.05 v=0.5   v=2  v=16
5%   1.152 1.193 1.282 1.155
10%  1.462 1.499 1.594 1.434
20%  1.858 1.875 1.957 1.831
30%  2.117 2.183 2.219 2.049
40%  2.348 2.428 2.467 2.363
50%  2.571 2.674 2.694 2.621
60%  2.796 2.958 2.968 2.880
70%  3.048 3.335 3.371 3.150
80%  3.438 4.047 4.186 3.613
90%  4.618 6.197 6.234 5.240
95%  7.126 7.098 7.049 6.648
```

먼저 각 MCMC 결과의 초기 2000개의 추출값을 burn 했다.

추출한 chain들의 quantile 을 비교했다.

그 결과 $v = 16$ 에서 다른 경우와 비교했을 때 90%, 95% 의 quantile 값이 0.3~0.8 차이났다.