통계계산 과제

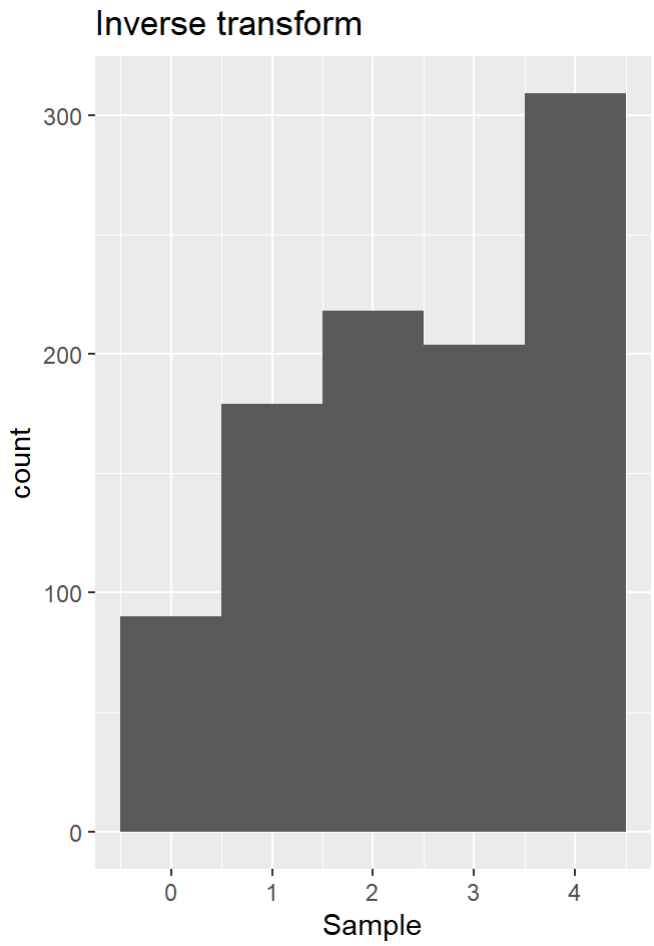
Seongmin Ji (student id:2021710322)

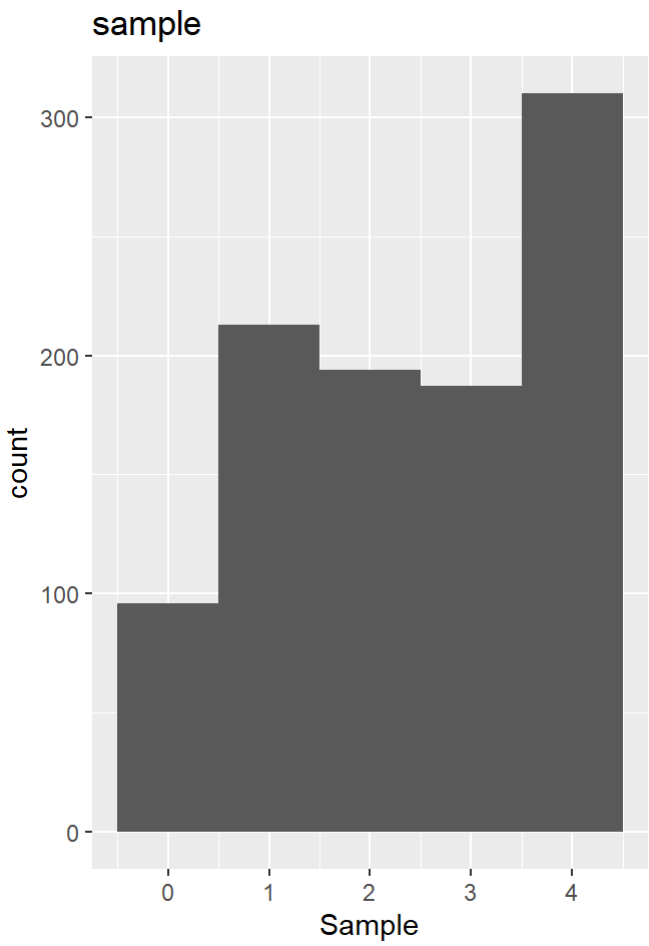
2021 4 16

각 답에 대한 설명이 필요할 경우, 뒤에 보충하였다.

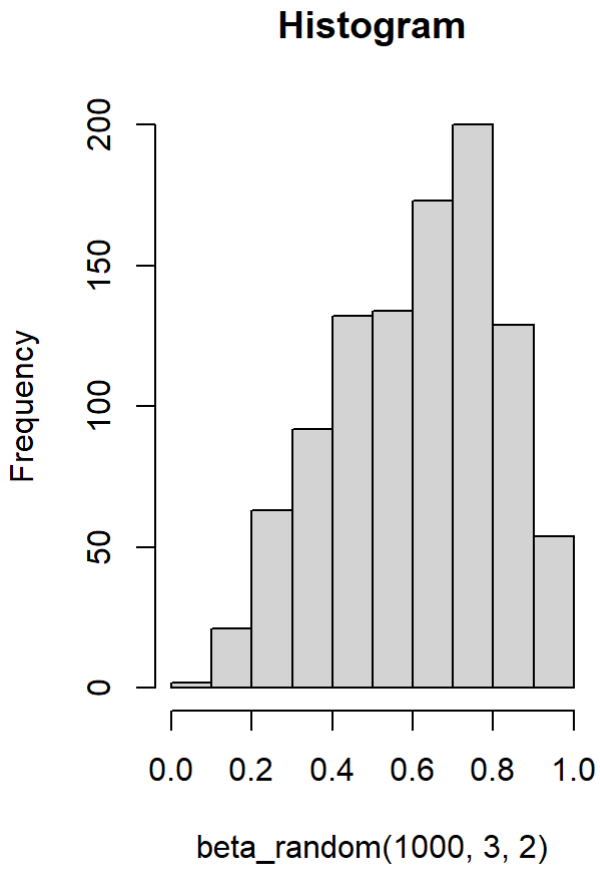
Chapter 3

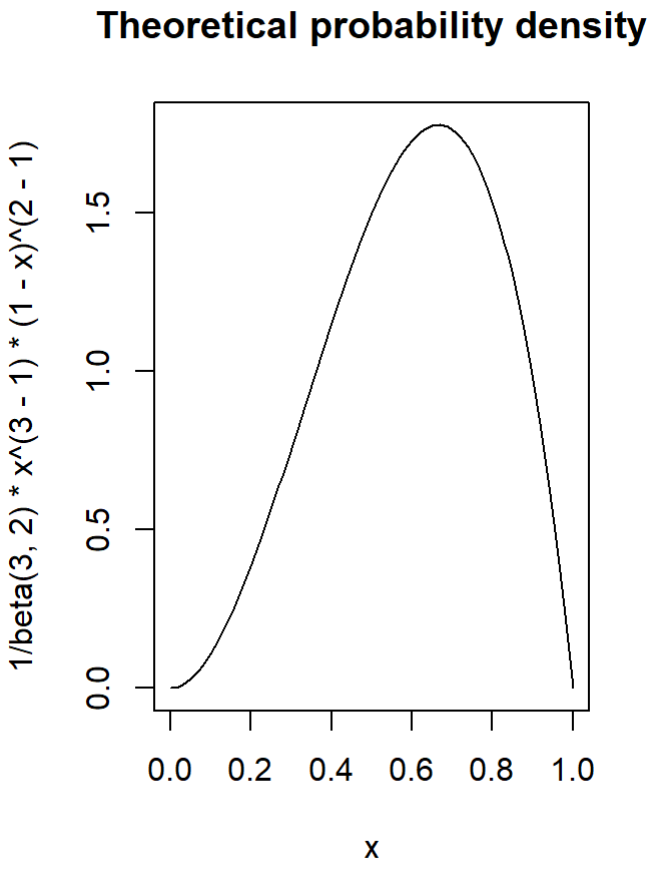
```
## 3.5
library(ggplot2)
library(cowplot)

sampler <- function(n){
  unif <- runif(n)
  d1 <- as.integer(unif >= 0.1)
  d2 <- as.integer(unif >= 0.3)
  d3 <- as.integer(unif >= 0.5)
  d4 <- as.integer(unif >= 0.7)
  x <- d1 + d2 + d3 + d4
  return(x)
}

aset <- c(0, 1, 1, 2, 2, 3, 3, 4, 4, 4)
df_3.5 <- data.frame(inverse = sampler(1000), sample = sample(aset, 1000, replace = T))

p1 <- ggplot(df_3.5, aes(x = inverse)) +
  ggtitle("Inverse transform") +
  xlab("Sample") + geom_histogram(bins = 5)

p2 <- ggplot(df_3.5, aes(x = sample)) +
  ggtitle("sample") +
  xlab("Sample") + geom_histogram(bins = 5)

plot_grid(p1, p2)
```


Inverse transform
300
200
100
0
count
0
1
2
3
4
Sample

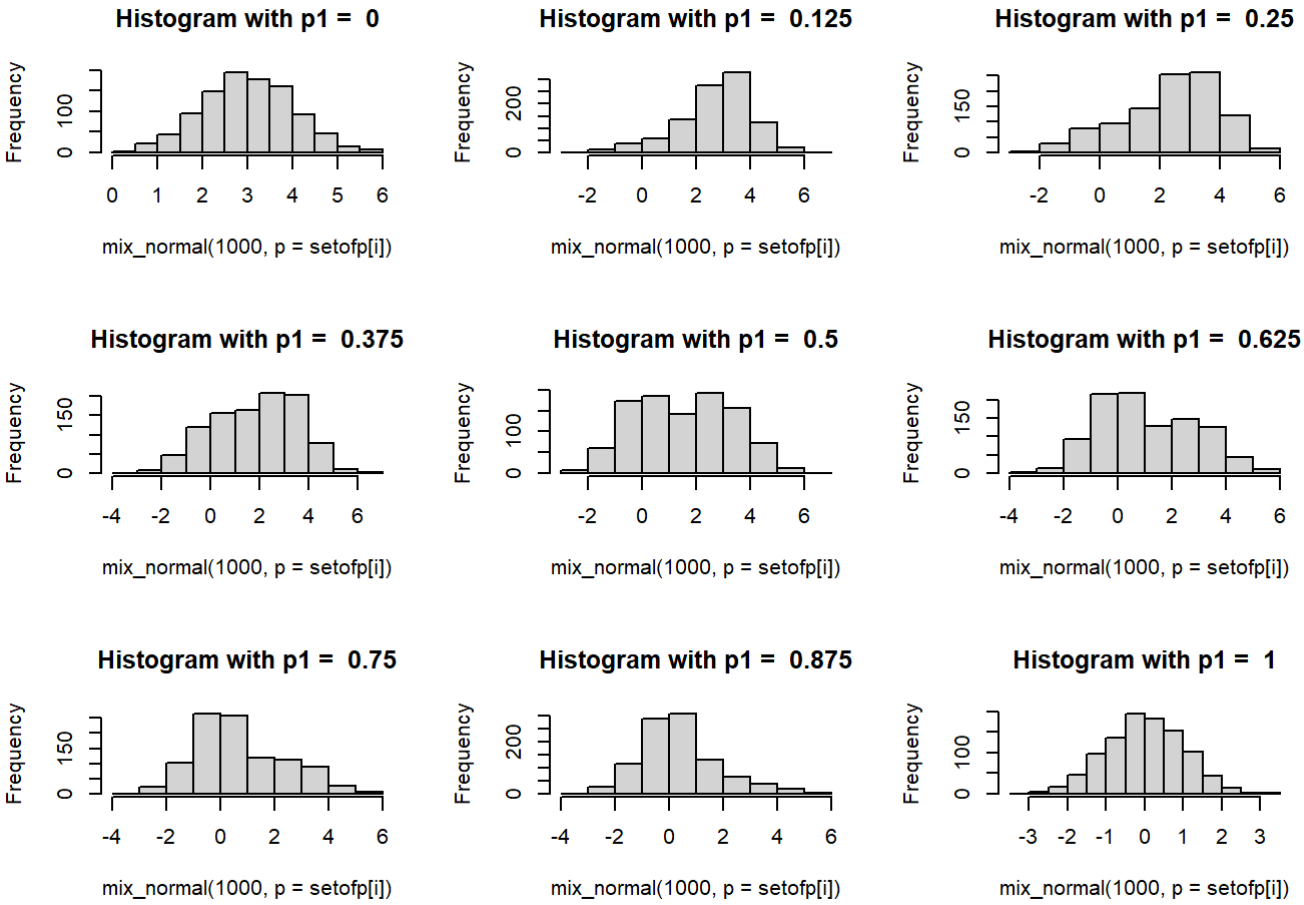

sample
300
200
100
0
count
0
1
2
3
4
Sample

```
## 3.7
beta_random <- function(n, a, b){
  if(a <= 0 | b <= 0) stop("a, b should have a positive value")
  else{
    if(a == 1 & b == 1){
      f <- function(x){1}
      C <- 1
    }else{
      f <- function(x) {1/beta(a, b)*x^(a-1)*(1-x)^(b-1)}
      argmax <- (1-a) / (2 - a - b)
      ## at x = (1 - a) / (2 - a - b), f(x) = C
      C <- 1/beta(a, b) * (argmax)^(a-1) * (1-argmax)^(b-1)
    }
    g <- function(x){1}
    j <- 1
    x <- vector("numeric", n)
  }
  ## acceptance-rejection method
  count <- 0
  while (j <= n + 1) {
    count <- count + 1
    if (j > n) return(x)
    else{
      y <- runif(1)
      u <- runif(1)
      if(u <= f(y) / (C*g(y))){
        x[j] <- y
        j <- j + 1
      }
    }
  }
}

par(mfrow = c(1,2))
hist(beta_random(1000, 3, 2), main = "Histogram")
curve(1/beta(3, 2)*x^(3-1)*(1-x)^(2-1), from = 0, to = 1, add =  FALSE, main = "Theoretical probability densit
y")
```

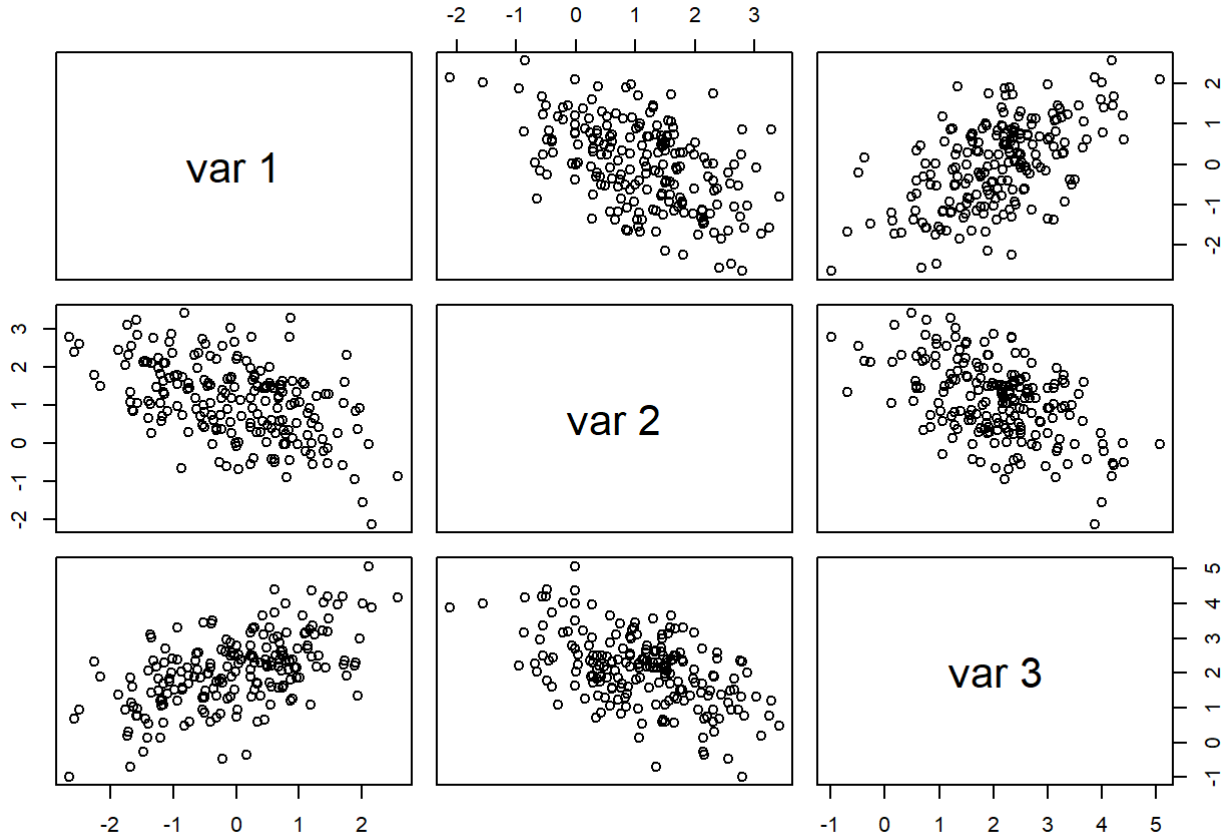





```
## 3.11
mix_normal <- function(n, p){
  if (p < 0 & p > 1) stop("p should be belong to [0, 1]")
  p1 <- as.integer(runif(n) < p)
  p2 <- 1 - p1
  return(p1 * rnorm(n, mean = 0, sd = 1) + p2 * rnorm(n, mean = 3, sd = 1))
}

par(mfrow = c(3,3))
setofp <- seq(from= 0, to = 1, length = 9)

for(i in 1:9){
  hist(mix_normal(1000, p = setofp[i]), main = paste("Histogram with p1 = ", toString(setofp[i])), breaks = 10)
}
```



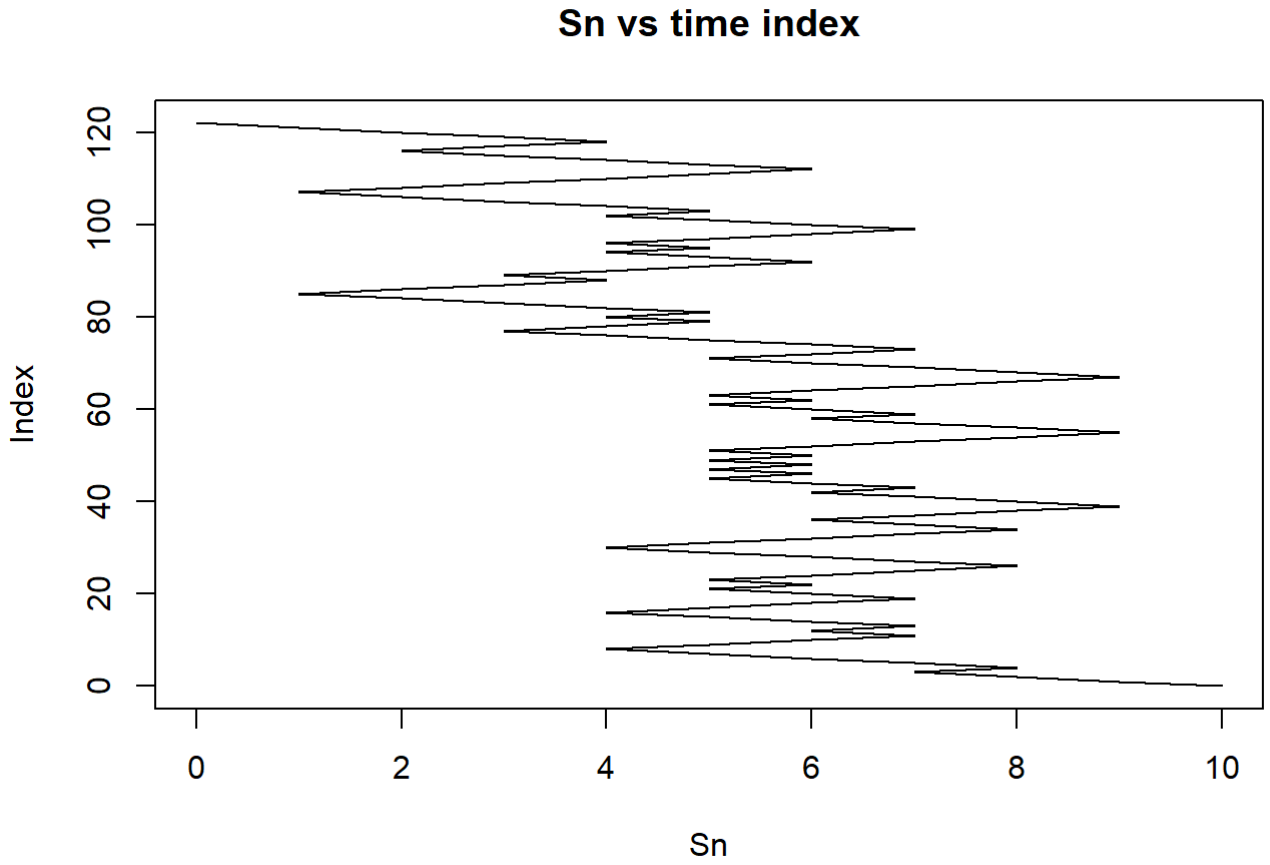

```
## 3.14
library(MASS)
library(mvtnorm)

rmvn_Choleski <- function(n, mu, Sigma){
  d <- length(mu)
  Q <- chol(Sigma)
  Z <- matrix(rnorm(n*d), nrow = n, ncol = d)
  X <- Z %*% Q + matrix(mu, n, d, byrow = TRUE)
}

choleski_sample <- rmvn_Choleski(200, mu = c(0,1,2), Sigma = matrix(c(1, -0.5, 0.5, -0.5, 1, -0.5, 0.5, -0.5, 1
), 3,3))

pairs(choleski_sample)
```




```
cov(choleski_sample)
```

```
##            [,1]       [,2]       [,3]
## [1,]  1.1227282 -0.5285476  0.6185893
## [2,] -0.5285476  0.9972267 -0.5299190
## [3,]  0.6185893 -0.5299190  1.0465015
```

```
## 3.19
Sn <- 10
x <- Sn
while(Sn < 20 & Sn > 0){
  incr <- sample(c(-1, 1), 1)
  Sn <- Sn + incr
  x <- c(x, Sn)
}

plot(x, 0:(length(x)-1), type = "l", main = "Sn vs time index", xlab = "Sn", ylab = "Index")
```



Chapter 5

추정하고자하는 θ 를 다음과 같이 정의한다.

$$C \sim Cauchy(0,\ 1)$$

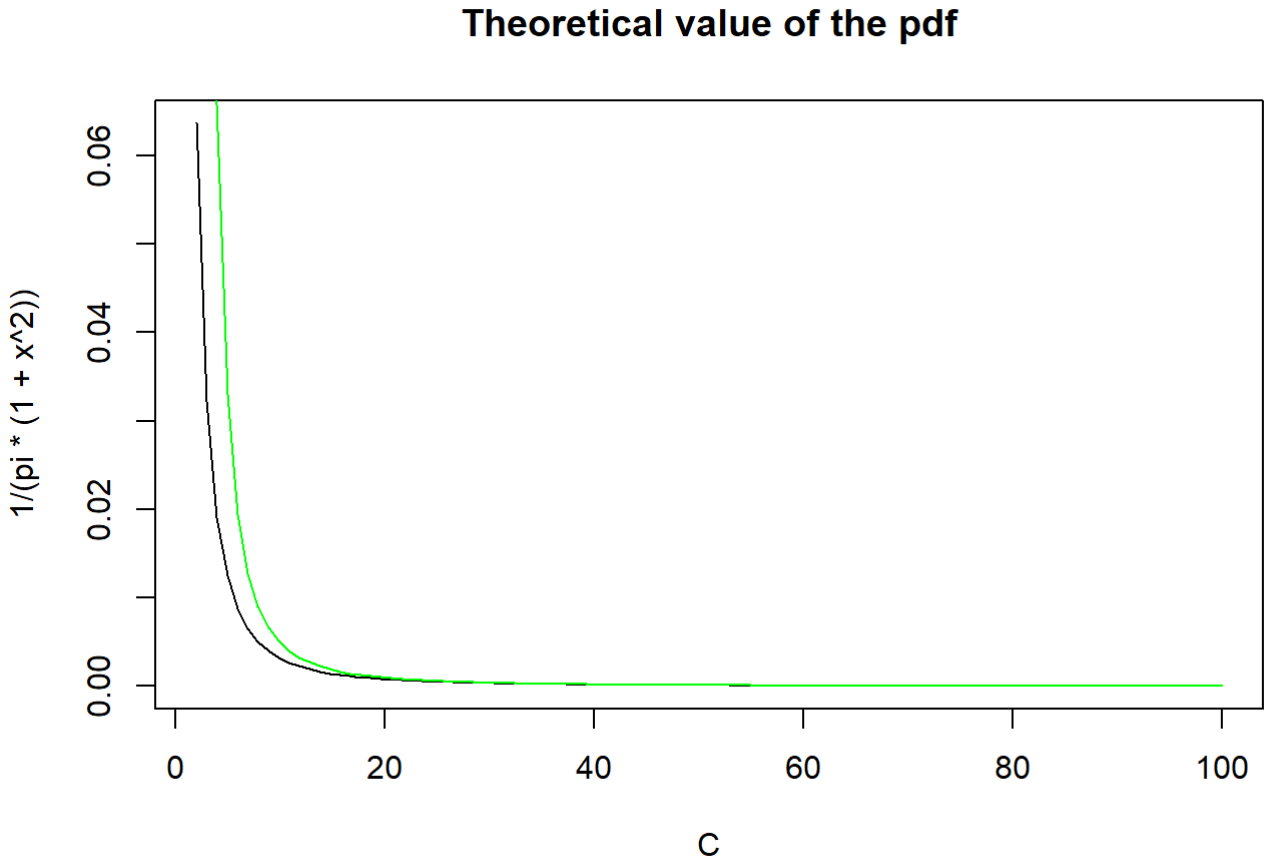
$$\theta = P(C > 2) = \int_2^{\infty} \frac{1}{\pi(1+x^2)} dx$$

$$= \frac{1}{2} - \pi^{-1} \tan(2) \approx 0.1476$$

```
## 5.0.1 : Simple Monte Calro
SMC_theta <- function(n = 10000){
  ind <- rcauchy(n) > 2
  return(mean(ind))
}
SMC_theta(10000)
```

```
## [1] 0.143
```

```
## 5.0.2: Importance sampling

### Theoretical value of the pdf
curve(1/(pi*(1+x^2)), 2, 100, n = 101, add = FALSE, xlab = "C", type = 'l', main = "Theoretical value of the pd
f")
curve(1/(pi*(1+(x-2)^2)), 2, 100, n = 101, add = TRUE, col = "green",type = 'l')
```



코시분포 (0, 1)의 확률밀도함수는 2보다 큰 정의역에서 매우 작은 치역을 갖기 때문에 (검정색 라인) 일반적인 몬테칼로 방법에서 필요한 표본을 얻기 어렵다.

한편, 위치모수를 변경한 코시분포(2, 1)의 확률밀도함수의 치역은 그보다 큰 값을 갖는다. (초록색 라인) 따라서 두번째 분포에서 표본을 추출하는 Importance sampling 방법을 사용하려고 한다.

```
imp_theta <- function(n = 10000){
  u <- rcauchy(n, location = 2)
  g <- function(x){(x > 2)}
  f <- function(x) {1/(pi*(1+x^2))}
  phi <- function(x) {1/(pi*(1+(x-2)^2))}
  weight <- f(u)/phi(u)
  return(mean(weight*g(u)))
}

imp_theta(10000)
```

```
## [1] 0.1473127
```

난수는 코시분포(2, 1)로 부터 추출된다. Local variable 중 'weight'는 앞서 언급한 두 확률밀도함수의 비를 의미하며, 모든 함수의 곱이 처음 적분식과 같도록 한다.

```
## 5.0.3: Hit or Miss
```

균등분포 난수를 사용하는 Hit or Miss 방법을 이용하기 위해 분포의 토대를 변경한다.

코시분포 (0,1)의 확률밀도함수는 0을 기준으로 좌우 대칭이므로

$$\theta = \int_2^{\infty} \frac{1}{\pi(1+x^2)} dx \iff$$

$$1-2\theta = 2(0.5-\theta) = 2\left(\int_0^{\infty} \frac{1}{\pi(1+x^2)} dx - \int_2^{\infty} \frac{1}{\pi(1+x^2)} dx \right)$$

$$= 2\int_0^{2} \frac{1}{\pi(1+x^2)} dx \quad (1)$$

따라서 위의 식 (1)을 이용하면, 균등분포 (0, 2) 난수를 이용할 수 있다.

```
hom_theta <- function(n = 10000){
  g <- function(x) {1/(pi*(1+x^2))}
  x <- runif(n) * 2
  y <- runif(n) / pi
  p <- 1 / pi * 2 * 2 * mean(y <= g(x))
  return((1-p)/2)
}

hom_theta(10000)
```

```
## [1] 0.1479493
```

Local variable 중 'g' 의 최댓값은 $1/\pi$ 이므로 'y'에 단위를 곱한다.

그리고 local variable 중 'p' 는 $1-2\theta$ 를 의미한다.

'mean(y <= g(x))' 은 비가 나오므로 단위 $2/\pi$ 를 곱해야 하며, 식 (1) 에 따라 총 적분값의 두배를 해야하므로 2 를 추가로 곱한다.

```
## 5.0.4: Control Variate
```

이번에도 균등분포를 이용하기위해 함수를 다음과 같이 변환한다.

$$\theta = \int_2^{\infty} \frac{1}{\pi(1+x^2)} dx \iff$$

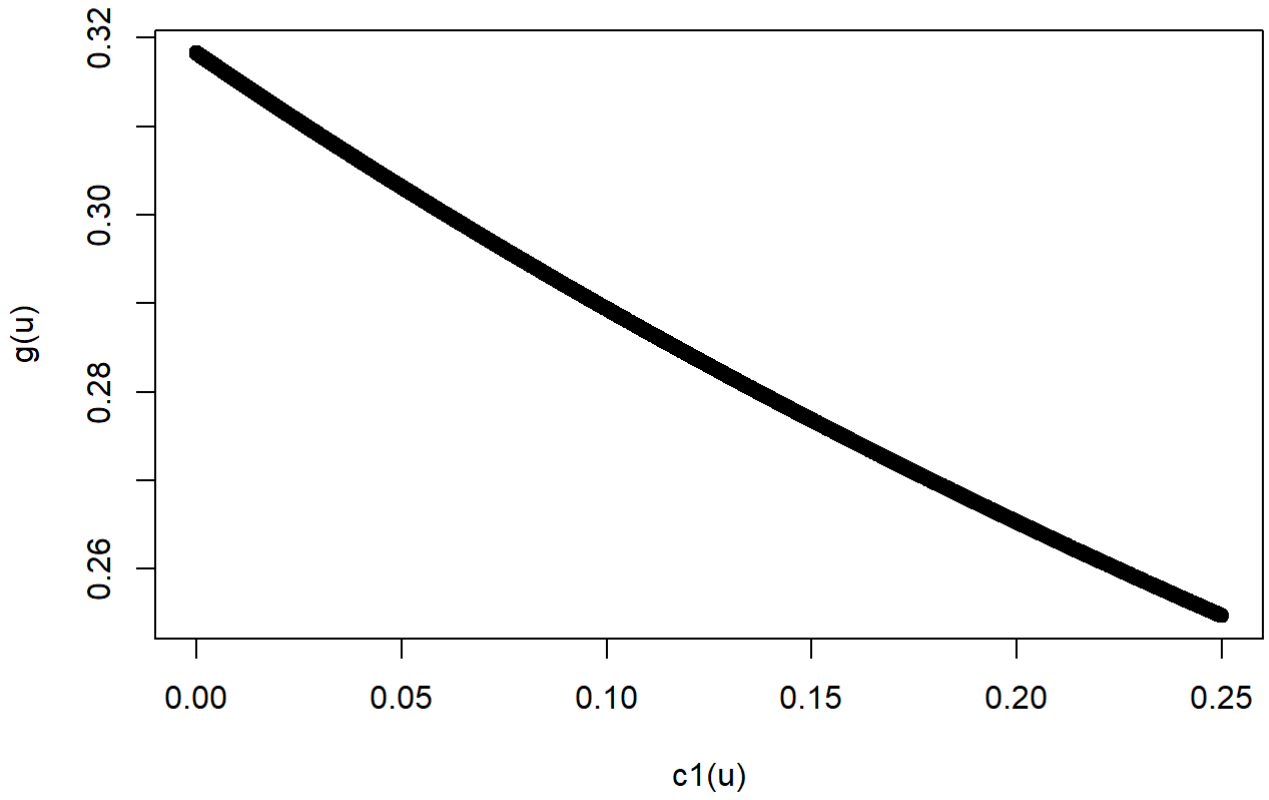
$$\theta = \int_0^{1/2} \frac{y^{-2}}{\pi(1+y^{-2})} dy, \quad \text{from } x = \frac{1}{y}$$

$$= \int_0^{1/2} \frac{1}{\pi(1+y^2)} dy \quad (2)$$

(2) 적분 내의 함수를 g라 하면 g 는 y^2 의 함수이다. 그러면 $g(y)$를 통과한 난수는 y^2이나 y^4 를 통과한 난수와 상관관계가 있다.

u 를 균등분포(0, 0.5)에서 추출한 난수라고 하자. (2)의 식을 통과한 $g(u)$ 를 구하면 u^2 이나 u^4 와 상관관계를 갖게 된다.

```
g <- function(x) {1/(pi*(1+x^2))}
c1 <- function(x) {x^2}

n = 10000
set.seed(1000)
u <- runif(n) / 2
plot(c1(u), g(u))
```



$c1(u) = u^2$ 라고 하면 $g(u)$ 와 위와 같은 관계를 갖는다.

이제 u^2 를 $g(u)$ 의 control variate 로 사용하자. U^2 의 Uniform(0, 0.5) 에 대한 기댓값(μ)은 $1/12$ 임을 알고 있으므로 사용한다.

```
### h(x) = g(x)/f(x), f(x) = pdf of U(0, 0.5)
h <- function(x) {g(x) / 2}

mu <- 1/12
b <- -cov(h(u), c1(u))/var(c1(u))

mean(h(u))
```

```
## [1] 0.1476194
```

```
mean(h(u) + b*(c1(u) - mu))
```

```
## [1] 0.1475829
```

```
var(h(u)) / var(h(u) + b*(c1(u) - mu))
```

```
## [1] 313.1503
```

따라서 Control variate 를 사용한 경우가 그렇지 않은 경우보다 작은 분산을 나타냈다. 이를 이용하여 함수를 작성하자.

```
### X^2 as a control variate
cv_theta_x2 <- function(n = 10000){

  # f = pdf_u(0, 0.5) = 2
  u <- runif(n) / 2

  # h = g / f
  h <- function(x) {1/(pi*(1+x^2)) / 2}

  # control variate
  c1 <- function(x) {x^2}
  # since int^0.5_0 c1 * f = 1/12 = mu
  mu <- 1/12
  b <- -cov(h(u), c1(u))/var(c1(u))
  return(mean(h(u) + b*(c1(u) - mu)))
}

### X^4 as a control variate
cv_theta_x4 <- function(n = 10000){

  # f = pdf_u(0, 0.5) = 2
  u <- runif(n) / 2

  # h = g / f
  h <- function(x) {1/(pi*(1+x^2)) / 2}

  # control variate
  c2 <- function(x) {x^4}
  # since int^0.5_0 c2 * f = 1/80 = mu
  mu <- 1/80
  b <- -cov(h(u), c2(u))/var(c2(u))
  return(mean(h(u) + b*(c2(u) - mu)))
}

cv_theta_x2(10000)
```

```
## [1] 0.1475795
```

```
cv_theta_x4(10000)
```

```
## [1] 0.1475679
```

다음 5.0.5 에서는 이전 5.0.1~4 에서 나타낸 모든 방법을 100번 시뮬레이션하여 평균값과 표준오차를 계산하려고 한다.

```
## 5.0.5
set.seed(1200)
simul_matrix <- function(n = 100){
  # write a matrix
  sm <- matrix(NA, nrow = 100, ncol = 5)

  # naive MC
  for(i in 1:100){
    sm[i, 1] <- SMC_theta(10000)
  }

  # Importance sampling
  for(i in 1:100){
    sm[i, 2] <- imp_theta(10000)
  }

  # Hit or Miss
  for(i in 1:100){
    sm[i, 3] <- hom_theta(10000)
  }

  # Control variate
  for(i in 1:100){
    sm[i, 4] <- cv_theta_x2(1000)
  }

  for(i in 1:100){
    sm[i, 5] <- cv_theta_x4(1000)
  }

  s.mean <- apply(sm, 2, mean)
  s.e. <- apply(sm, 2, sd)

  result <- rbind(c(0.1476, s.mean),
                  c(NA, s.e.))
  colnames(result) <- c("True value", "Simple MC", "Importance", "Hit or Miss", "Control X^2", "Control X^4")
  rownames(result) <- c("mean", "s.e.")
  return(result)
}

simul_matrix(n=100)
```

```
##      True value   Simple MC  Importance Hit or Miss  Control X^2  Control X^4
## mean     0.1476 0.147676000 0.147454814 0.147385221 1.475859e-01 1.475955e-01
## s.e.         NA 0.003321102 0.001981265 0.002852208 1.624671e-05 9.995967e-05
```

표준오차 값이 작을수록, θ의 더 좋은 추정량이라고 할 수 있다. 따라서, Control variate의 방법이 가장 좋고(그 중 X^2를 사용), 그 다음 Importance sampling, 그 다음 Hit or Miss, 그 다음 Simple MC 의 방법 순으로 좋다는 결과를 얻었다.

```
## 5.7

MC_theta <- function(n = 10000, antithetic = TRUE) {
  u <- runif(n/2)
  ### Check the choice whether simple Monte Calro or Antithetic
  if (!antithetic) v <- runif(n/2) else
    v <- 1 - u
  u <- c(u, v)
  g <- (1-0) * mean(exp(u))
  return(g)
}
```

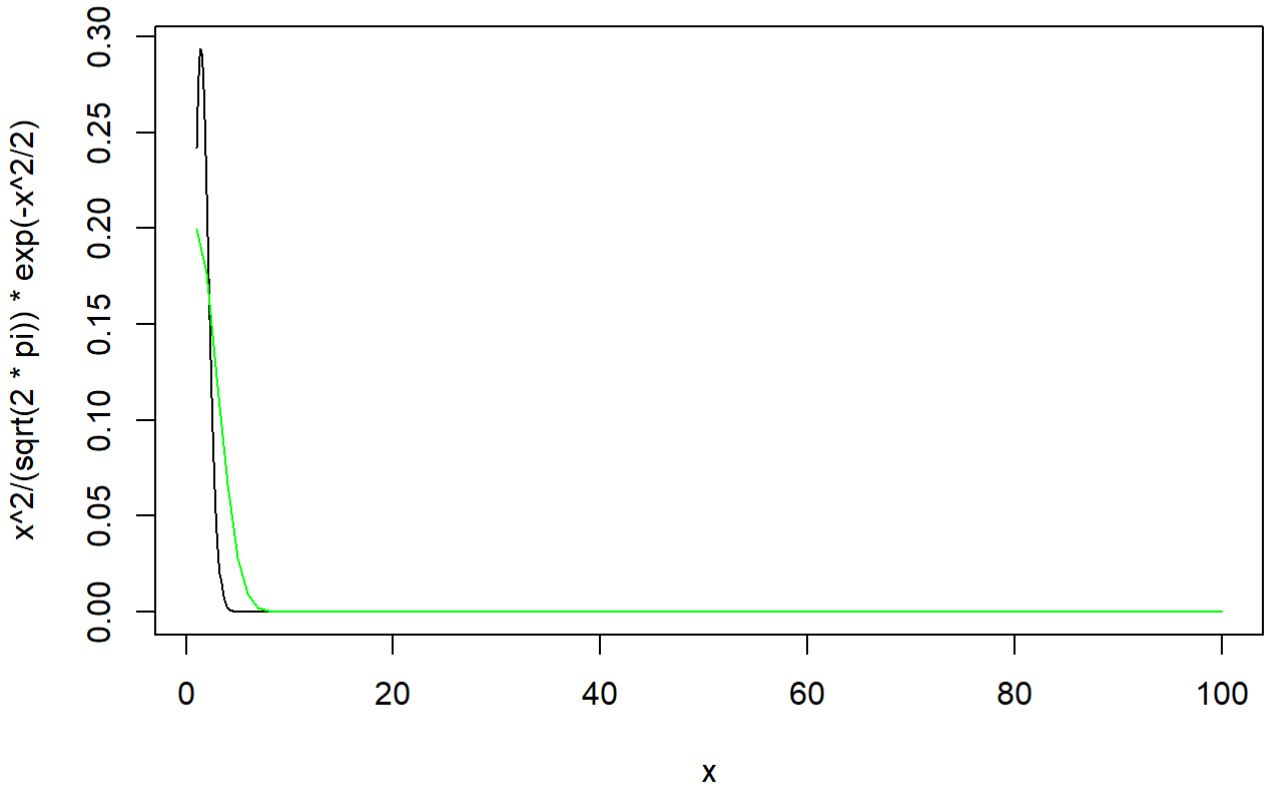
위 함수는 'antithetic' 를 통해 Antithetic 방법과 Simple MC 방법을 모두 구현한다. 이제 이것을 100번 시뮬레이션해서 표본분산을 비교하려고 한다.

```
set.seed(1300)
n <- 10000
r <- 100
MC1 <- MC2 <- numeric(r)

for(i in 1:r){
  MC1[i] <- MC_theta(n, antithetic = FALSE)
  MC2[i] <- MC_theta(n, antithetic = TRUE)
}


(var(MC1) - var(MC2)) / var(MC1)
```

```
## [1] 0.9662505
```

따라서 Antithetic 방법을 사용했을 때, Simple MC 방법을 사용했을 때보다 분산이 95% 이상 감축된 결과를 얻었다.

```
## 5.14

curve(x^2/(sqrt(2*pi))*exp(-x^2/2), 1, 100, n = 1000, add = FALSE, type = 'l')
curve(1/(sqrt(2*pi*4))*exp(-(x-1)^2/8), 1, 100, n = 101, add = TRUE, col = "green",type = 'l')
```




```
imp_5.14 <- function(n = 10000){
  u <- rnorm(n, mean = 1, sd = 2)
  g <- function(x){x^2*(x > 1)}
  f <- function(x) {1/(sqrt(2*pi))*exp(-x^2/2)}
  phi <- function(x) {1/(sqrt(2*pi*4))*exp(-(x-1)^2/8)}
  weight <- f(u)/phi(u)
  return(mean(weight*g(u)))
}

imp_5.14(10000)
```

```
## [1] 0.3975435
```

이 값은 WolframAlpha (R) 를 이용해서 얻은 값인 0.400626 과 유사하다.

```
knitr::include_graphics("5.14.jpg")
```

Definite integral:

$$\int_1^{\infty} \frac{x^2 \exp\left(-\frac{x^2}{2}\right)}{\sqrt{2\pi}}\, dx = \frac{1}{2}\,\text{erfc}\left(\frac{1}{\sqrt{2}}\right) + \frac{1}{\sqrt{2\,e\,\pi}} \approx 0.400626$$

Chapter 6

```
## 6.A
n <- 20
alpha <- .05
```

유의수준을 0.05로 설정하고, 카이제곱(1), 균등분포(0, 2), 지수분포(1) 에서 얻은 표본평균에 대한 검정을 하려고 한다. 모두 기댓값이 1이므로, $\mu = 1$ 이라는 귀무가설을 세운다. t검정을 이용하여 검정통계량의 p-value 값을 구한 다음, 유의수준을 기준으로 귀무가설을 기각시킨다. 그리고 실제 귀무가설이 참임에도 불구하고 기각된 비율, 즉 1종오류율을 구하여 유의수준과 차이가 있는지를 알아본다.

```
### (i) chi-squared(1)
set.seed(1500)
#mu0 of chisq(1) = 1
mu0 <- 1

m <- 10000          #number of replicates
p <- numeric(m)     #storage for p-values
for (j in 1:m) {
  x <- rchisq(n, 1)
  ttest <- t.test(x, alternative = "two.sided", mu = mu0)
  p[j] <- ttest$p.value
}

p.hat.chi <- mean(p < alpha)

### (ii) Uniform(0,2)

#mu0 of Uniform(0,2) = 1
mu0 <- 1

m <- 10000          #number of replicates
p <- numeric(m)     #storage for p-values
for (j in 1:m) {
  x <- runif(n, min = 0, max = 2)
  ttest <- t.test(x, alternative = "two.sided", mu = mu0)
  p[j] <- ttest$p.value
}

p.hat.unif <- mean(p < alpha)

### (iii) Exponential(1)

#mu0 of Exponential(1) = 1
mu0 <- 1

m <- 10000       #number of replicates
p <- numeric(m)     #storage for p-values
for (j in 1:m) {
  x <- rexp(n, rate = 1)
  ttest <- t.test(x, alternative = "two.sided", mu = mu0)
  p[j] <- ttest$p.value
}

p.hat.exp <- mean(p < alpha)



result <- c(alpha, p.hat.chi, p.hat.unif, p.hat.exp)
names(result) <- c("Alpha", "Chisq(1)", "Unif(0,2)", "Exp(1)")

result
```

```
##     Alpha  Chisq(1) Unif(0,2)    Exp(1)
##    0.0500    0.1070    0.0496    0.0816
```

10000번의 시뮬레이션 결과, 카이제곱(1)과 지수분포(1)의 경우에는 실제 유의수준 값보다 2배에 달하는 1종오류율이 나타났다. 반면 균등분포(0, 2) 의 경우 실제 유의수준과 유사한 1종오류율이 나타났다. 이는 정규분포와 균등분포가 대칭적인 분포 형태를 갖는 것과 반대로, 카이제곱과 지수분포는 기운분포 형태를 갖기 때문이라고 추론할 수 있다.