HW2

Congyuan Duan

November 11, 2021

1 2.10

1.1 (1)

Proof that the sum of r independent geometric random variables with parameters p is a negative binomial(r,p) random variable: Suppose $X_1,\cdots,X_r \overset{iid}{\sim} geometric(p)$, i.e. $P(X_i = x_i) = p(1-p)^{x_i-1}$. Then

$$\begin{aligned} P(\sum_{i=1}^{r} X_i = x) &= \sum_{x_1+\cdots+x_r=x} P(X_1 = x_1,\cdots,X_r = x_r) \\ &= \sum_{x_1+\cdots+x_r=x} P(X_1 = x_1)\cdots P(X_r = x_r) \\ &= \sum_{x_1+\cdots+x_r=x} p^r(1-p)^{x-r} \\ &= C_{x-1}^{r-1} p^r (1-p)^{x-r} \end{aligned}$$

which is the probability distribution of negative binomial(r,p). So we can generate $r \times n$ geometric random number to obtain n negative binomial random number.

1.2 (2)

Suppose $U \sim U(0,1)$, $X \sim$ negative binomial(r,p)

$$F(x) = P(X \le x) = \begin{cases} \sum_{k=r}^{x} C_{k-1}^{r-1} p^r (1-p)^{k-r} & x = r, r+1, \cdots \\ 0 & x = 0, \cdots, r-1 \end{cases}$$

$Y = x$ when $F(x-1) < U \le F(x)$, then Y is the negative binomial random number.

[1]:
```r
F <- function(x) {
    if(x<r) {return(0)}
    else{
        k <- c(r:x)
        return(sum(choose(k-1, r-1)*p^r*(1-p)^(k-r)))
    }
} #cdf
N <- 10000 # sample size
X <- c(1:10000000) # value of X, when X has infinite value, set the upper
 ↪bound=10000000
```

```
inver_sampling <- function(N, F, X, seed) {
    set.seed(seed)
    U <- runif(N, 0, 1)
    Y <- rep(0, N)
    for (i in 1:N){
        k <- 1
        while(U[i] > F(k-1)) k <- k + 1
        Y[i] <- k - 1
    }
    return(Y)
}
```

2 2.11

2.1 (1)

$U \sim U(0,1)$. The cdf of $Beta(\frac{1}{n},1)$ is $F(x)=x^{1/n}$. Then $X=U^n$ is the $Beta(\frac{1}{n},1)$ random variable.

2.2 (2)

The cdf of $Beta(n,1)$ is $F(x)=x^n$. Then $X=U^{1/n}$ is the $Beta(n,1)$ random variable.

2.3 (3)

The corresponding cdf is $F(x)=\int_0^x \frac{2}{\pi\sqrt{1-t^2}}dt=\frac{2}{\pi}\arcsin x$. Then $X=\sin(\frac{\pi}{2}U)$ is the target random variable.

2.4 (4)

The corresponding cdf is $F(x)=\int_{-\infty}^x \frac{1}{\pi(1+t^2)}dt=\frac{1}{\pi}\arctan x+\frac{1}{2}$. Then $X=\tan(\pi(U-\frac{1}{2}))$ is the target random variable.

2.5 (5)

The corresponding cdf is $F(x)=\int_0^x \cos t dt=\sin x$. Then $X=\arcsin U$ is the target random variable.

2.6 (6)

The corresponding cdf is $F(x)=\int_0^x \frac{\alpha}{\eta}t^{\alpha-1}e^{-x^\alpha/\eta}dt=1-e^{-x^\alpha/\eta}$. Then $X=(-\eta\log(1-U))^{1/\alpha}$ is the target random variable.

3 2.13

The inverse transformation is

$$\begin{pmatrix} \alpha & =\arctan\frac{Y}{X} \\ R & =X^2+Y^2 \end{pmatrix} \tag{1}$$

The Jacobi determinant is

$$J = \begin{vmatrix} -\sqrt{R}\sin\alpha & \frac{1}{2\sqrt{R}}\cos\alpha \\ \sqrt{R}\cos\alpha & \frac{1}{2\sqrt{R}}\sin\alpha \end{vmatrix} = -\frac{1}{2} \tag{2}$$

Therefore, the joint density of (X, Y) is

$$f(x, y) = 2\frac{1}{2\pi}\frac{1}{2}e^{-\frac{(x^2+y^2)}{2}} \tag{3}$$
$$= \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}} \tag{4}$$

So X and Y are independent and both $\sim N(0, 1)$.

4 3.2

4.1 (1)

[2]:
```
# random point sampling
randompoint_samp <- function(N, h, low, high, m = 0, M, B = 1, seed = NA) { #N␣
↪is the sample size, h(x) is the target function
    Iall <- rep(0, B)                                    #B is repeated␣
↪times(when calculate variance)
    for(i in 1:B) {                                      #M and m are the upper␣
↪bound and lower bound of h(x)
        if (!is.na(seed)) {set.seed(seed[i])}            #low and high are the␣
↪upper bound and lower bound of x
        else {set.seed(i)}
        x <- runif(N, low, high)
        y <- runif(N, m, M)
        p  <- sum(y < h(x))
        Iall[i] <- p / N * (M-m) * (high-low)
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
N <- 10000
I1 <- randompoint_samp(N, exp, -1, 1, 0, exp(1))
I1$mean
```

2.37577831807321

[3]:
```
# average sampling
N <- 10000
average_samp <- function(N, h, low, high, B = 1, seed = NA) {
    Iall <- rep(0, B)
    for (i in 1:B) {
        if (!is.na(seed)) {set.seed(seed[i])}
```

```r
        else {set.seed(i)}
        u <- runif(N, low, high)
        Iall[i] <- (high - low) * sum(h(u)) / N
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
I2 <- average_samp(N, exp, -1, 1)
I2$mean
```

2.35766992054428

[4]:

```r
# importance sampling using U(-1,1) as g(x)
importance_samp <- function(N, g, h, Ginver, B = 1, seed = NA) { #g is the␣
↪selected function, Ginver is the inverse of its cdf
    Iall <- rep(0, B)
    for (i in 1:B) {
        if (!is.na(seed)) {set.seed(seed[i])}
        else {set.seed(i)}
        u <- runif(N, 0, 1)
        x <- Ginver(u)
        Iall[i] <- sum(h(x)/g(x)) / N
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
g <- function(x) {
    return(1/2)
}
Ginver <- function(y) {
    return(2*y-1)
}
I3 <- importance_samp(N, g, exp, Ginver)
I3$mean
```

2.35766992054428

[5]:

```r
# importance sampling using 1/2*(x+1) as g(x)
importance_samp <- function(N, g, h, Ginver, B = 1, seed = NA) {
    Iall <- rep(0, B)
    for (i in 1:B) {
        if (!is.na(seed)) {set.seed(seed[i])}
        else {set.seed(i)}
        u <- runif(N, 0, 1)
        x <- Ginver(u)
        Iall[i] <- sum(h(x)/g(x)) / N
```

```
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
g1 <- function(x) {
    return(1/2*(x+1))
}
Ginver1 <- function(y) {
    return(2*sqrt(y)-1)
}
I3_1 <- importance_samp(N, g1, exp, Ginver1)
I3_1$mean
```

2.35460530650625

(I do not try this $g(x)$ in the following analysis, but it may be better)

[6]:
```
# hierachical sampling, mixed with average sampling
hierachical_samp <- function(n, h, seg, B = 1, seed = NA) {
    m <- length(seg) - 1 #number of segments
    Iall <- rep(0, B)
    for (j in 1:B) {
        for (i in 1:m) {
            if (!is.na(seed)) {Iall[j] <- Iall[j] + average_samp(n[i], h,␣
↪seg[i], seg[i+1], seed = seed[j])$mean}
            else {Iall[j] <- Iall[j] + average_samp(n[i], h, seg[i], seg[i+1],␣
↪seed = c(j))$mean}
        }
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
I4 <- hierachical_samp(c(5000, 5000), exp, c(-1, 0, 1))
I4$mean
```

2.34722253862014

4.2 (2)

random point sampling
$Var(\hat{I}_1) = [M(b-a)]^2 p(1-p)/N$, where $M = e$, $b = 1$, $a = -1$ and $p = (e - e^{-1})/2e$.

$$
\begin{aligned}
Var(\hat{I}_1) &= \frac{(2e)^2(e-e^{-1})(2e-e+e^{-1})}{N \times 2e \times 2e} \\
&= \frac{7.254}{N}
\end{aligned}
$$

In order to store only 3 digits after the decimal point, we need to keep $z_{0.975}\sqrt{Var(\hat{I}_1)} < 0.0005$, i.e. $N > 111467866$.

average sampling

$Var(\hat{I}_2) = (b-a)^2 Var(h(U))/N$, where $b = 1$, $a = -1$ and $Var(h(U)) = \int_a^b [h(u)-E(h(U))]^2 du/(b-a) = 1/2 - 1/2 \times e^{-2}$.

$$\begin{aligned} Var(\hat{I}_2) &= \frac{2^2(1/2 - 1/2 \times e^{-2})}{N} \\ &= \frac{1.729}{N} \end{aligned}$$

In order to store only 3 digits after the decimal point, we need to keep $z_{0.975}\sqrt{Var(\hat{I}_2)} < 0.0005$, i.e. $N > 26568506$.

importance sampling

$Var(\hat{I}_3) = Var(h(X)/g(X))/N$, where $X \sim g(x) = U(-1, 1)$, then $Var(h(X)/g(X)) = Var(2e^x) = 2 - 2e^{-2}$.

$$\begin{aligned} Var(\hat{I}_3) &= \frac{2 - 2e^{-2}}{N} \\ &= \frac{1.729}{N} \end{aligned}$$

In order to store only 3 digits after the decimal point, we need to keep $z_{0.975}\sqrt{Var(\hat{I}_3)} < 0.0005$, i.e. $N > 26568506$. **hierachical sampling**

$Var(\hat{I}_4) = Var(\hat{I}_{41}) + Var(\hat{I}_{42}) = (b_1 - a_1)^2 Var(h(U_1))/(N/2) + (b_2 - a_2)^2 Var(h(U_2))/(N/2)$, where $b_1 = a_2 = 0$, $a_1 = -1$, $b_2 = 1$, $U_1 \sim U(-1, 0)$, and $U_2 \sim U(0, 1)$.

$$\begin{aligned} Var(\hat{I}_4) &= \frac{(1 - e^{-2}) - 2(1 - e^{-1})^2}{N} + \frac{(e^2 - 1) - 2(e - 1)^2}{N} \\ &= \frac{0.55}{N} \end{aligned}$$

In order to store only 3 digits after the decimal point, we need to keep $z_{0.975}\sqrt{Var(\hat{I}_4)} < 0.0005$, i.e. $N > 8451520$.

4.3 (3)

[7]:
```
B <- 100
I1 <- randompoint_samp(N, exp, -1, 1, 0, exp(1), B)
I2 <- average_samp(N, exp, -1, 1, B)
I3 <- importance_samp(N, g, exp, Ginver, B)
I3_1 <- importance_samp(N, g1, exp, Ginver1, B)
I4 <- hierachical_samp(c(5000, 5000), exp, c(-1, 0, 1), B)
Var <- data.frame(name = c("I1", "I2", "I3", "I4"), Var = c(I1$var, I2$var,
 ↪I3$var, I4$var),
                  Var_hat=c(7.254e-4, 1.729e-4, 1.729e-4, 0.55e-4))
Var
#column Var is calculated with sample variance, Var_hat is calculated in (2),
 ↪they are very similar
```

A data.frame: 4 × 3

name <chr>	Var <dbl>	Var_hat <dbl>
I1	6.017275e-04	0.0007254
I2	1.522177e-04	0.0001729
I3	1.522177e-04	0.0001729
I4	9.956849e-05	0.0000550

4.4 (4)

[8]:
```
I <- exp(1) - exp(-1)
seed <- matrix(c(1:100), nrow = 100)
Iall1 <- unlist(mapply(randompoint_samp, seed = seed, MoreArgs = list(N = N, h␣
↪= exp, low = -1, high = 1, m = 0,
                                                                    M = exp(1),␣
↪B = 1))[1,])
Iall2 <- unlist(mapply(average_samp, seed = seed, MoreArgs = list(N = N, h =␣
↪exp, low = -1, high = 1, B = 1))[1,])
Iall3 <- unlist(mapply(importance_samp, seed = seed, MoreArgs = list(N = N, g =␣
↪g, h = exp, Ginver = Ginver, B = 1))[1,])
Iall4 <- unlist(mapply(hierachical_samp, seed = seed, MoreArgs = list(n =␣
↪c(5000, 5000), h = exp, seg = c(-1, 0, 1), B = 1))[1,])
MAE <- data.frame(name = c("I1", "I2", "I3", "I4"), MAE = c(sum(abs(Iall1 -␣
↪I)), sum(abs(Iall2 - I)),
                                                              sum(abs(Iall3 -␣
↪I)), sum(abs(Iall4 - I))) / B)
MAE
```

A data.frame: 4 × 2

name <chr>	MAE <dbl>
I1	0.019757754
I2	0.009963982
I3	0.009963982
I4	0.007786707

$MAE(I_1) > MAE(I_2) = MAE(I_3) > MAE(I_4)$. Hierachical sampling is the most accurate method.

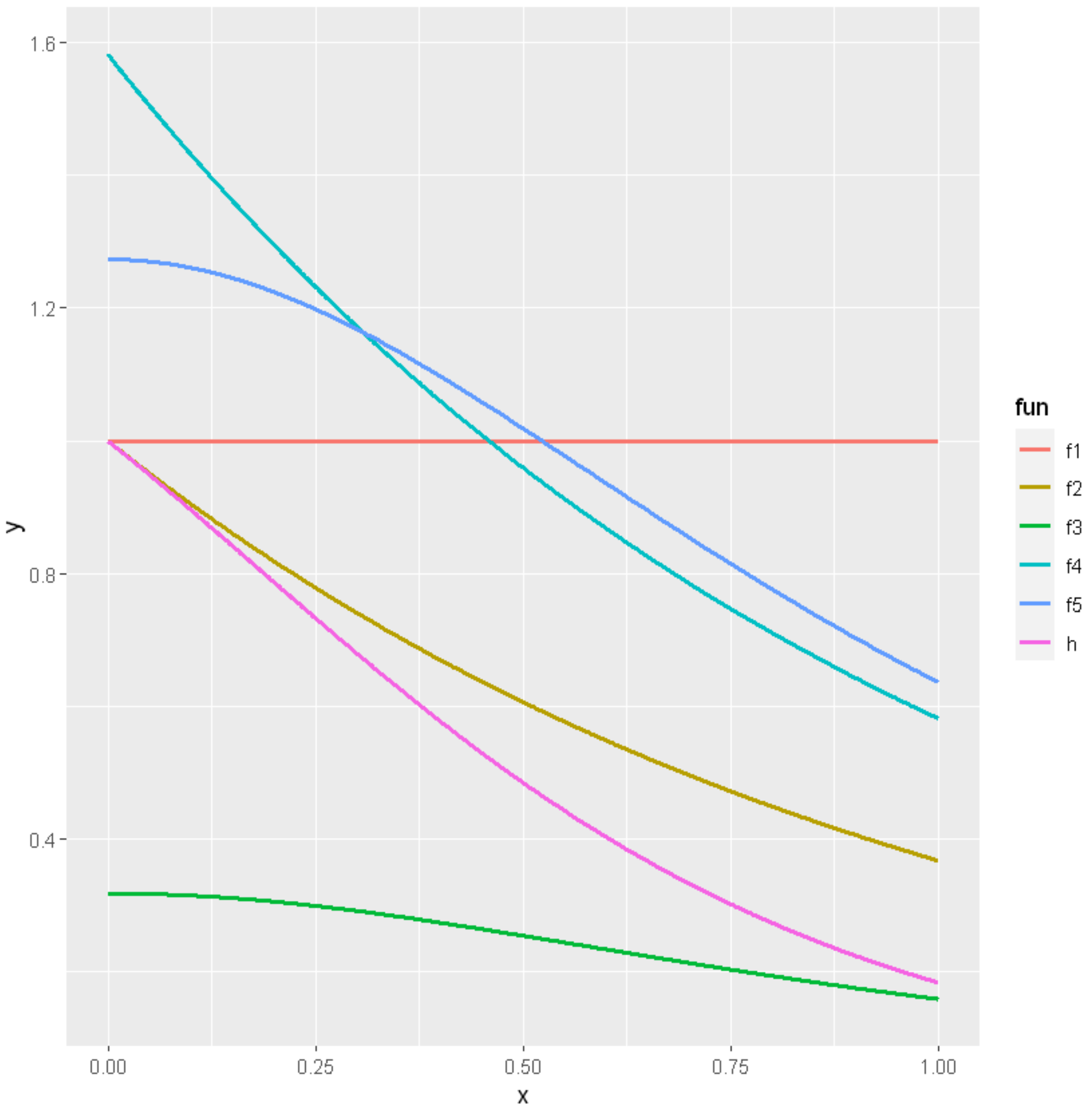
5 3.3

5.1 (1)

plot all the functions in $x \in [0, 1]$

[9]:
```
library(ggplot2)
x1 <- seq(length = 1000, from = 0, to = 1)
h <- function(x) {
    return(exp(-x)/(1+x^2))
}
```

```r
f1 <- function(x) {
    return(rep(1, length(x)))
}
f2 <- function(x) {
    return(exp(-x))
}
f3 <- function(x) {
    return(1/(pi*(1+x^2)))
}
f4 <- function(x) {
    1/(1-exp(-1))*exp(-x)
}
f5 <- function(x) {
    4/(pi*(1+x^2))
}
dist <- data.frame(x = rep(x1, 6),
                   y = c(h(x1), f1(x1), f2(x1), f3(x1), f4(x1), f5(x1)),
                   fun = rep(c("h", "f1", "f2", "f3", "f4", "f5"), each = 1000))
ggplot(dist) + geom_line(aes(x = x, y = y, color = fun), size = 1)
```



5.2 (2)

The domain of f_2 and f_3 is larger than $(0, 1)$. When sampling a number outside the domain $(0, 1)$, we need to drop it.

[10]:
```
importance_samp2 <- function(N, g, h, Ginver, a, b, B = 1, seed = NA) {
    Iall <- rep(0, B)              # (low,high)is the domain of h(x)
    for (i in 1:B) {
        if (!is.na(seed)) {set.seed(seed[i])}
        else {set.seed(i)}
        u <- runif(N, 0, 1)
        x <- Ginver(u)
        x1 <- x[(x>a)&(x<b)]
```

```r
        Iall[i] <- sum(h(x1)/g(x1)) / N #attention!here should be N not␣
↪length(x1)
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
```

[11]:
```r
F1inver <- function(y) {
    return(y)
}
F2inver <- function(y) {
    return(-log(1-y))
}
F3inver <- function(y) {
    return(tan(pi*(y-1/2)))
}
F4inver <- function(y) {
    return(-log(1-y*(1-exp(-1))))
}
F5inver <- function(y) {
    return(tan(pi/4*y))
}
set.seed(1)
I1 <- importance_samp(N, f1, h, F1inver, B = 100)
I2 <- importance_samp2(N, f2, h, F2inver, 0, 1, B = 100)
I3 <- importance_samp2(N, f3, h, F3inver, 0, 1, B = 100)
I4 <- importance_samp(N, f4, h, F4inver, B = 100)
I5 <- importance_samp(N, f5, h, F5inver, B = 100)
result <- data.frame(name = c("I1", "I2", "I3", "I4", "I5"),
                     mean = c(I1$mean, I2$mean, I3$mean, I4$mean, I5$mean),
                     var = c(I1$var, I2$var, I3$var, I4$var, I5$var))
result
```

A data.frame: 5 × 3

name	mean	var
<chr>	<dbl>	<dbl>
I1	0.5244436	4.849249e-06
I2	0.5242430	1.510319e-05
I3	0.5246926	6.861769e-05
I4	0.5246850	8.236157e-07
I5	0.5246004	1.625485e-06

5.3 (3)

From the plot in (1), the shape of $f_1(x)$ is very different from $h(x)$, so it may have larger variance. Compared with $f_4(x)$ and $f_5(x)$, the domain of f_2 and f_3 is larger than $(0, 1)$ and we drop many sample points, which can explain their larger variace. Actually, if we transform f_2 and f_3 to $(0, 1)$, they are the same as f_4 and f_5.

5.4 (4)

[12]:
```
n <- rep(1000, 10)
seg <- seq(from = 0, to = 1, length = 11)
I6 <- hierachical_samp(n, h, seg, B = 100)
I6$mean
I6$var
```

0.524789484149965

4.88366826222391e-07

$Var(I_6)$ is smaller than the above.

5.5 (5)

[13]:
```
#hierachical sampling another version
hierachical_samp2 <- function(m, h, B = 1) {
    Iall <- rep(0, B)
    for (i in 1:B) {
        set.seed(i)
        u <- runif(m, 0, 1)
        n <- c(1:m)
        Iall[i] <- sum(h((n - 1 + u) / m)) / m
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
I7 <- hierachical_samp2(N, h, B = 100)
I7$mean
I7$var
```

0.524797107715719

5.48934860538895e-14

$Var(I_7)$ is smaller than the above. Hierachical sampling further reduces the variance.

6 3.9

[14]:
```
# multiple integration
multiple_average_samp <- function(N, m, h, low, high, B = 1) { # m is dimension
    Iall <- rep(0, B)
    for (j in 1:B) {
        set.seed(j)
        u <- matrix(0, nrow = N, ncol = m)
        for (i in 1:m) {
            u[,i] <- runif(N, low[i], high[i])
```

```
        }
        Iall[j] <- prod(high - low) * sum(apply(u, 1, h)) / N
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
low <- c(0, 0)
high <- c(1, 1)
h <- function(x) {
    return(exp((sum(x))^2))
}
I <- multiple_average_samp(N, 2, h, low, high, B = 100)
I$mean
I$var
```

4.90653697166515

0.00342042897686122

[15]:

```
# antithetic sampling
set.seed(2)
antithetic_sampling <- function(N, m, h, low, high, B = 1) {
    Iall <- rep(0, B)
    for (j in 1:B) {
        set.seed(j)
        u <- matrix(0, nrow = N, ncol = m)
        for (i in 1:m) {
            u[,i] <- runif(N, low[i], high[i])
        }
        Iall[j] <- prod(high - low) * (sum(apply(u, 1, h)) +
 ↪sum(apply(low+high-u, 1, h))) / (2 * N)
    }
    I <- mean(Iall)
    Var <- mean((Iall - I)^2)
    return(list(mean = I, var = Var))
}
low <- c(0, 0)
high <- c(1, 1)
h <- function(x) {
    return(exp((sum(x))^2))
}
I <- antithetic_sampling(N, 2, h, low, high, B = 100)
I$mean
I$var
```

4.89894833859011

0.00107924962051474

Antithetic sampling can reduce variance.