Homework 6.1

Michael Pena

2024-04-05

1.

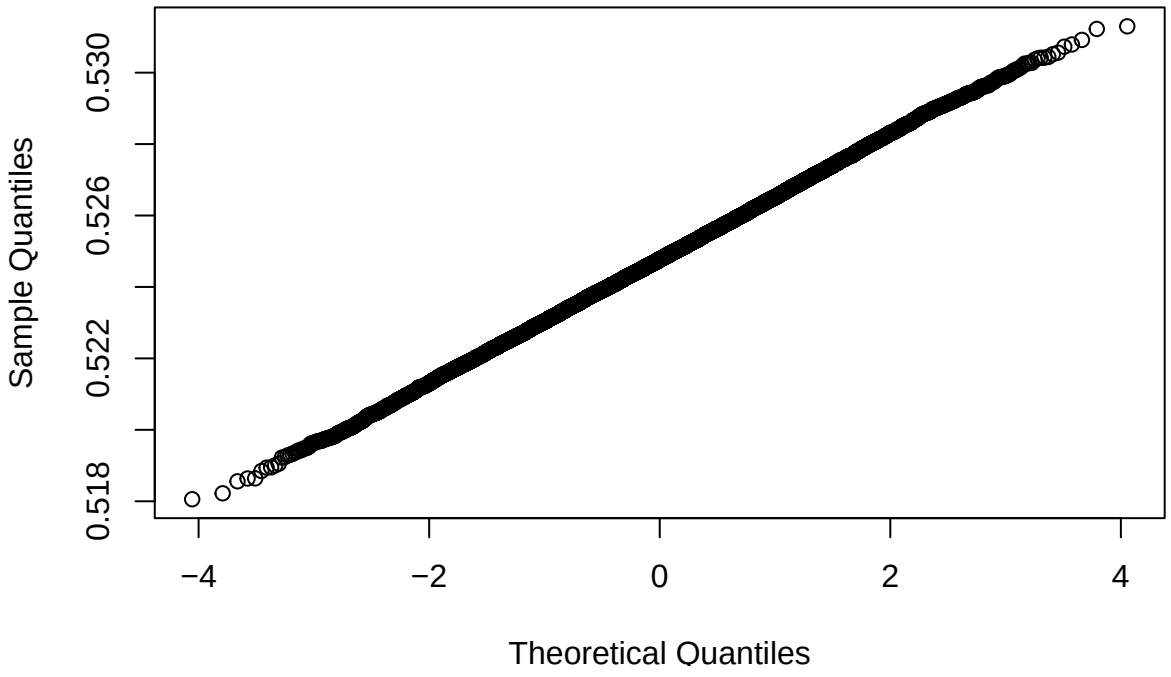
part (a).

```
#using Monte Carlo Approximation to find the
mcnorm <- function (n, a, b) {
  x = runif(n,a,b)
  g = exp(-x)/(1+x^2)
  tetah=(b-a)*mean(g) # this is the Monte Carlo approximation to the integral
  tetah_se=(b-a)*sd(g)/sqrt(n) #This is the standard error of the approximation
  exact=0.52479714326 #This is the exact value
  list(exact=exact, tetah=tetah,tetah_se=tetah_se)
}

n = 20000
a=0; b=1
I = mcnorm(n,a,b)
output <- data.frame(Exact = I$exact, M_C_approx = I$tetah, S.E. = I$tetah_se)
print(output)
```

```
##       Exact M_C_approx        S.E.
## 1 0.5247971  0.5229477 0.001728992
```

```
# Check standard error and normality
rep=20000
Ivec = numeric(rep)
for (i in 1:rep){
  I = mcnorm(n,a,b)
  Ivec[i]= I$tetah
}
qqnorm(Ivec)
```




```
sd(Ivec)
```

```
## [1] 0.001741434
```

```
# true variance /standard error of MC estimator
pab = pnorm(b)-pnorm(a)
v = (((b-a)/(2*sqrt(pi)))*(pnorm(sqrt(2)*b)-pnorm(sqrt(2)*a))-pab^2)/n
se_mc = sqrt(v)
print(se_mc)
```

```
## [1] 0.0003423826
```

part (b).

```
# antithetical method of montecarlo
# render points
x = runif(n/2)
# first part of function
gu1 = exp(-x)/(1+x^2)
# second part
gu2 = exp(-(1-x))/(1+(1-x)^2)
# make the theta_A and theta_SE
thetaA = (1/n)*sum(gu1+gu2)
thetaSE = sqrt((1/(2*n)))*(sd(gu1+gu2))
thetaA;thetaSE
```

```
## [1] 0.5250648
```

```
## [1] 0.0003315516
```

###part (c).

```r
# variance reduction from the original MC variance
var1 = I$tetah_se^2
var2 = thetaSE^2
(var1 - var2)/var1
```

```
## [1] 0.9635618
```

this yields a 96.32335% reduction from the original M.C. variance to the "antithetic" version.

part (d).

```r
cor(gu1,gu2)
```

```
## [1] -0.9636302
```

the antithetical method has two functions that are negatively correlated; this method should render a lower S.E.

2

part (a).

```r
# basic calc tells us that that we need to find F_x = F(x) - F(0) to get Finv_x
# invF
invF <- function(n){
  U = runif(n,0,1)
  X = -log(1-U*(1-exp(-1)))
  return(X)
}
```

part (b).

```r
# generating new data
invF(20000) -> X.rvs
#using Monte Carlo Approximation to find the
mc2<- function (n, a, b) {
  x = X.rvs
  g = exp(-x)/(1+x^2)
  tetah=(b-a)*mean(g) # this is the Monte Carlo approximation to the integral
  tetah_se=(b-a)*sd(g)/sqrt(n) #This is the standard error of the approximation
  exact=0.5247971 #This is the exact value
  list(exact=exact, tetah=tetah,tetah_se=tetah_se)
}

n = 20000
a=0; b=1
I = mc2(n,a,b)
output <- data.frame(Exact = I$exact, M_C_approx = I$tetah, S.E. = I$tetah_se)
print(output)
```

```
##       Exact M_C_approx        S.E.
## 1 0.5247971  0.5965194 0.001748835
```

```r
# Check standard error and normality
rep=20000
```

```
Ivec = numeric(rep)
for (i in 1:rep){
  I = mcnorm(n,a,b)
  Ivec[i]= I$tetah
}
#qqnorm(Ivec)
sd(Ivec)
```

```
## [1] 0.001736705
```

```
# true variance /standard error of MC estimator
pab = pnorm(b)-pnorm(a)
v = (((b-a)/(2*sqrt(pi)))*(pnorm(sqrt(2)*b)-pnorm(sqrt(2)*a))-pab^2)/n
se_mc = sqrt(v)
print(se_mc)
```

```
## [1] 0.0003423826
```

part (c).

```
# antithetical method of montecarlo
# render points
x <- invF(n/2)
# first part of function
gu1 = exp(-x)/(1+x^2)
# second part
gu2 = exp(-(1-x))/(1+(1-x)^2)
# make the theta_A and theta_SE
thetaA = (1/n)*sum(gu1+gu2)
thetaSE = sqrt((1/(2*n)))*(sd(gu1+gu2))
thetaA;thetaSE
```

```
## [1] 0.5262354
```

```
## [1] 0.000333946
```

3

part (a).

```
# make sigma and mu
Sig <- matrix(c(1,3/5,1/3,
                3/5,1,11/15,
                1/3,11/15,1),nrow = 3, byrow = T)
mu <- c(0,0,0)

# generate points from 3-variate norm
mvrnorm(20000,mu,Sig) -> X_i

# count hits
hit = 0
for(i in 1:n){
  if (X_i[i,1] <= 1 && X_i[i,2] <= 4 && X_i[i,3] <= 2){hit = hit + 1}
}
# probability
```

```r
p0 = hit/n
p0
```

```
## [1] 0.8282
```

part (b).

```r
# estimating the standard error of approximation
stan.err = sqrt((p0*(1-p0))/n)
stan.err
```

```
## [1] 0.002667253
```

part (c).

```r
z <- qnorm(.95)
CI <- c(p0 - z*stan.err, p0 + z*stan.err)
CI
```

```
## [1] 0.8238128 0.8325872
```

this confidence interval contains the true value of 0.8279...

4

part (a).

```r
# create uniform points
u <- runif(20000,0,1)
# find E[h(u_i)]
hu <- exp(-.5)/(1+u^2)
hu.bar <- mean(hu)
# find E[f(u_i)]
fu <- (exp(-u))/(1+u^2)
fu.bar <- mean(fu)
# get c-star
c = -cov(fu,hu)/var(hu)
# final estimate
thetaCV <- fu.bar + c*(hu.bar - 0.476368066183) # 0.476368066183 is exact E[h(x)] from Desmos
thetaCV
```

```
## [1] 0.5236584
```

part (b).

```r
0.001737464 -> thetaMC.se
#getting correlation
correl <- cov(fu,hu)/(sd(fu)*sd(hu))
correl
```

```
## [1] 0.9738543
```

if we reference https://www.value-at-risk.net/variance-reduction-with-control-variates-monte-carlo-simulation/ [5.68] $\sigma^* = \sigma\sqrt{1-\rho^2}$ thus

```
thetaCV.SEhat = thetaMC.se*sqrt(1 - correl^2)
thetaCV.SEhat
```

```
## [1] 0.0003947055
```

part (c).

```
# estimating the S.E.
sd(fu + c*(hu - 0.476368066183))/sqrt(n) -> thetaCV.se
thetaCV.se
```

```
## [1] 0.0003900191
```

numbers are pretty close together in the hundred-thousandths place.

part (d).

```
# render points from u
x <- runif(n)
u <- 2*(1-x)
# redo part a
# find E[h(u_i)]
hu = exp(-.5)/(1+u^2)
hu.bar = mean(hu)
# find E[f(u_i)]
fu = exp(-u)/(1+u^2)
fu.bar = mean(fu)
# get c-star
c = -cov(fu,hu)/var(hu)
# final estimate
thetaCV = fu.bar + c*(hu.bar - 0.476368066183) # 0.476368066183 is exact E[h(x)] from Desmos
thetaCV
```

```
## [1] 0.5418688
```

part (e).

```
# estimate standard error
sd(fu + c*(hu - 0.476368066183))/sqrt(n) -> thetaCV.se
thetaCV.se
```

```
## [1] 0.0005251042
```

standards errors are not as close in value as before.