STA3127 Statistical Computing

Jeong Geonwoo

DUE Monday, Dec 1

```
rm(list=ls())
set.seed(2018122062)
```

Note: You can only use runif $(\cdot, 0, 1)$ for random number generation.

Q1.

Suppose we want to estimate $\Pr\left\{(X-10)^2+(Y-10)^2 \leq 10\right\}$, where

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathrm{N}_2\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\right)$$

Let

$$\begin{pmatrix} X^* \\ Y^* \end{pmatrix} \sim \mathrm{N}_2\left(\begin{pmatrix} 10 \\ 10 \end{pmatrix}, v\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}\right)$$

for $v > 0$. Writing the densities of $(X, Y)'$ and $(X^*, Y^*)'$ as f and g, respectively, we can estimate the probability of interest using importance sampling as

$$\Pr\left\{(X-10)^2+(Y-10)^2 \leq 10\right\} = E_g\left[\mathbb{1}\left\{(X^*-10)^2+(Y^*-10)^2 \leq 10\right\}\frac{f(X^*,Y^*)}{g(X^*,Y^*)}\right].$$

i) (C) For each of $v \in \{0.1, 1, 10, 100\}$, generate $n = 10^6$ independent pairs of $(X^*, Y^*)'$, and estimate the probability and its standard error.

I will define the functions used for solving this problem.

```
# Fuction 1. Generating the Multivariate normal random variables
my_rmvnorm <- function(n, mean, sigma) {
  # Cholesky decomposition
  L <- chol(sigma)
  # Box-Muller Transformation
  my_norm <- function(n) {
    u1 <- runif(n/2); u2 <- runif(n/2)
    z1 <- sqrt(-2 * log(u1)) * cos(2 * pi * u2)
    z2 <- sqrt(-2 * log(u1)) * sin(2 * pi * u2)
    return(c(z1, z2))
  }
  # Standard Normal distribution
  z <- matrix(my_norm(n * length(mean)), ncol = length(mean), byrow = TRUE)
  # cov matrix
  mean + z %*%L
}
```

```r
# Function 2. The density of Multivariate normal distribution
mvnorm_density <- function(x_vec, mean_vec, cov_matrix) {
  k <- length(mean_vec)
  sqrt_det_cov <- sqrt(det(cov_matrix))
  inv_cov_matrix <- solve(cov_matrix)
  diff_vec <- x_vec - mean_vec
  exponent <- -0.5 * t(diff_vec) %*% inv_cov_matrix %*% diff_vec
  return( (1 / ((2 * pi)^(k / 2) * sqrt_det_cov)) * exp(exponent)[1,1] )
}

# Function 3,4. The density function of f and g
f.density <- function(vec){
  return(mvnorm_density(vec, c(0,0), matrix(c(1,0.5,0.5,1),ncol=2)))}

g.density <- function(vec, v){
  return(mvnorm_density(vec, c(10,10), v*matrix(c(1,0.5,0.5,1),ncol=2)))}
```

```r
# The number of pairs of (X*, Y*)
n = 1e6

# The R.V from g with v = (0.1, 1, 10, 100)
g.v.0.1 <- my_rmvnorm(n, c(10,10), 0.1 * matrix(c(1, 0.5, 0.5, 1), nrow=2))
g.v.1 <- my_rmvnorm(n, c(10,10), 1 * matrix(c(1, 0.5, 0.5, 1), nrow=2))
g.v.10 <- my_rmvnorm(n, c(10,10), 10 * matrix(c(1, 0.5, 0.5, 1), nrow=2))
g.v.100 <- my_rmvnorm(n, c(10,10), 100 * matrix(c(1, 0.5, 0.5, 1), nrow=2))

# The indicator function : (X-10)^2+(Y-10)^2<=10
indicator <- function(mat){
  # input : n*2 matrix
  # output : n indicator
  return(rowSums(matrix(c((mat[,1] - 10)^2, (mat[,2] - 10)^2), ncol=2))<10)
}

# weight = f/g
weight.v.0.1 <- apply(g.v.0.1, 1, f.density) / apply(g.v.0.1, 1, function(row) g.density(row, 0.1))
weight.v.1 <- apply(g.v.1, 1, f.density) / apply(g.v.1, 1, function(row) g.density(row, 1))
weight.v.10 <- apply(g.v.10, 1, f.density) / apply(g.v.10, 1, function(row) g.density(row, 10))
weight.v.100 <- apply(g.v.100, 1, f.density) / apply(g.v.100, 1, function(row) g.density(row, 100))

cat("v = 0.1 |mean:", mean(indicator(g.v.0.1) * weight.v.0.1), "\n",
    "v = 1 |mean:", mean(indicator(g.v.1) * weight.v.1), "\n",
    "v = 10 |mean:", mean(indicator(g.v.10) * weight.v.10), "\n",
    "v = 100 |mean:", mean(indicator(g.v.100) * weight.v.100))
```

```
## v = 0.1 |mean: 4.242025e-24
##  v = 1 |mean: 1.049956e-19
##  v = 10 |mean: 1.047975e-19
##  v = 100 |mean: 1.117123e-19
```

```r
cat("v = 0.1 |standard error:", sd(indicator(g.v.0.1) * weight.v.0.1) / sqrt(n), "\n",
    "v = 1 |standard error:", sd(indicator(g.v.1) * weight.v.1) / sqrt(n), "\n",
    "v = 10 |standard error:", sd(indicator(g.v.10) * weight.v.10) / sqrt(n), "\n",
    "v = 100 |standard error:", sd(indicator(g.v.100) * weight.v.100) / sqrt(n))
```

```
## v = 0.1 |standard error: 1.611155e-24
```

```
##  v = 1 |standard error: 1.890065e-21
##  v = 10 |standard error: 1.346979e-21
##  v = 100 |standard error: 3.86335e-21
```

ii) Find the most different estimate among the four, which means that the corresponding v produces the worst estimator. Observe that this worst estimator has the smallest (or nearly smallest depending on your seed) standard error. Explain, based on the densities f and g, why this particular v results in the poorest estimator, despite having the smallest standard error.

In terms of mean and standard error, the estimates are different for v=0.1 than for v=1, v=10, and v=100. Therefore, we can say that v=0.1 is the worst estimate. The reason is that at v=0.1, (X^*, Y^*) appears only at "too" close to (10,10), so even if weight = f/g is multiplied, we can not say datapoint is same to data from f.The small standard error means that the data was generated "too" close to (10,10), not that the estimator is accurate.

iii) What would happen if you were to increase v beyond 100? Explain the rationale behind this phenomenon based on the densities f and g.

If v is greater than 100, the data will be generated from a sufficiently large area centered at (10,10), so we can properly estimate the expectation value by multiplying by weight=f/g. However, if v is so large that too little data is generated near (10,10), then the intent of using g to generate data near (10,10) becomes meaningless and has no reducing variance effect.

Q2.

Suppose that the price $P(u)$ of a stock at time u follows a geometric Brownian motion with drift μ and volatility σ. Consider a European barrier call option with expiration t such that the option is alive only if $a \leq P(s) \leq b$ at $s < t$, where $a < b$. Let K be the strike price at t and $v \equiv P(0)$ be the initial stock price.

i) By applying the variance reduction techniques as discussed in class, derive an efficient estimator for the expected payoff using the expression of the expected payoff $C(K, t, v)$ of a vanilla call option. You must provide a rationale for introducing $C(K, t, v)$ into your estimator.

The expected value of payoff (Black-Scholes Model) :

$$\begin{aligned} C(K,t,v) &= E\left((P(t)-K)^+\right) = E\left((ve^w - K)^t\right) \\ &\text{where } w = \log\frac{P(t)}{V} \\ &= ve^{\mu t+\sigma^2 t/2}\Phi(\sigma\sqrt{t}+m) - K\Phi(m) \\ &\text{where } m = m\,(K,t,v) = \frac{t\mu - \log\frac{k}{V}}{\sigma\sqrt{t}} \end{aligned}$$

The payoff of barrier call option

$$R = I\left(ve^X > b\right)\left(ve^{X+Y} - K\right)^+$$

The expected payoff of barrier call option

$$\begin{aligned} E[R] =& E\left[R \mid ve^X > b\right] P\left(ve^X > b\right) + E\left[R \mid ve^X \leqslant b\right] P\left(ve^X < b\right) \\ =& E\left[R \mid X > \log\frac{b}{v}\right] P\left(X > \log\frac{b}{v}\right) + 0 \\ =& E\left[R \mid X > \log\frac{b}{v}\right] \bar{\Phi}\left(\frac{\log\frac{b}{v} - s\mu}{\sigma\sqrt{s}}\right) \end{aligned}$$

We assume that the option is alive only if $a \leq P(s) \leq b$. So from above result, we can say that :

$$\begin{aligned}
E[R] &= E[R \mid a \leq P(s) \leq b]P(a \leq P(s) \leq b) \\
&= E[R \mid log(\frac{a}{v}) \leq log(\frac{P(s)}{v}) \leq log(\frac{b}{v})]P(log(\frac{a}{v}) \leq log(\frac{P(s)}{v}) \leq log(\frac{b}{v})) \\
&= E[R \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})]P(log(\frac{a}{v}) \leq X \leq log(\frac{b}{v}))
\end{aligned}$$

From the fact that $E_X[X \mid Y] = E_Z[E_X[X \mid Y, Z] \mid Y]$,

$$\begin{aligned}
&E[R \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})] \\
&= E\left[E\left[R \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v}), X\right] \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})\right]
\end{aligned}$$

If a barrier call option with a current price of v and expiration at time t has a price P(s)>b at time s<t, then it is equivalent to a regular European call option with a current price of P(s) and expiration at time t-s.

Therefore,

$$\begin{aligned}
&E[R \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})] \\
&= E[C(K, t-s, ve^X) \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})]
\end{aligned}$$

That means X is from Truncated Normal distribution.

$$\begin{aligned}
&P(log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})) \\
&= P(\frac{log(\frac{a}{v}) - t\mu}{\sigma\sqrt{t}} \leq \frac{X - t\mu}{\sigma\sqrt{t}} \leq \frac{log(\frac{b}{v}) - t\mu}{\sigma\sqrt{t}}) \\
&= P(\frac{log(\frac{a}{v}) - t\mu}{\sigma\sqrt{t}} \leq Z \leq \frac{log(\frac{b}{v}) - t\mu}{\sigma\sqrt{t}}) \\
&= \Phi(\frac{log(\frac{b}{v}) - t\mu}{\sigma\sqrt{t}}) - \Phi(\frac{log(\frac{a}{v}) - t\mu}{\sigma\sqrt{t}})
\end{aligned}$$

In conclusion, we can express the expected payoff of barrier call option as follow :

$$\begin{aligned}
E[R] &= E[R \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})]P(log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})) \\
&= E[C(K, t-s, ve^X) \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})]\left[\Phi(\frac{log(\frac{b}{v}) - t\mu}{\sigma\sqrt{t}}) - \Phi(\frac{log(\frac{a}{v}) - t\mu}{\sigma\sqrt{t}})\right]
\end{aligned}$$

ii) (C) For $\mu = 1, \sigma = 1, a = 1, b = 2, K = 10, v = 1, s = 3$, and $t = 10$, estimate the expected payoff of the option using your estimator in i) with $n = 10^5$, and estimate the standard error of the estimator. [Use the expression of $C(K, t, v)$ as provided in the lecture slides.]

First, I will set the given parameters

```
mu=1; sigma=1; a=1; b=2; K=10; v=1; s=3; t=10; n=1e5
```

Second, I will define $C(K, t, s)$ as function C

```
C <- function(mu, sigma, K, t, v){
  m <- (t * mu - log(K/v)) / sigma / sqrt(t)
  return (v * exp( mu * t + sigma^2 * t / 2)
          + pnorm(sigma * sqrt(t) + m) - K * pnorm(m))}
```

Third, I will define $\frac{log(\frac{a}{v})-s\mu}{\sigma\sqrt{s}}$ and that for b.

```
log_normalized_a.v <- (log(a/v)-s*mu) / sigma / sqrt(s)
log_normalized_b.v <- (log(b/v)-s*mu) / sigma / sqrt(s)
```

Fourth, I will generate X from truncated normal distribution using Inverse CDF technique. The CDF and Inverse CDF of truncated normal distribution are following :

$$F(x) = \frac{\Phi(x) - \Phi(a)}{\Phi(b) - \Phi(a)}, \quad F^{-1}(x) = \Phi^{-1}([\Phi(b) - \Phi(a)]U + \Phi(a))$$

```
q <- (pnorm(log_normalized_b.v) - pnorm(log_normalized_a.v)
      ) * runif(n) + pnorm(log_normalized_a.v)
X_trunc <-  qnorm(q) * sigma * sqrt(s) + s * mu
```

Finally, I will get $E[C(K, t-s, ve^X) \mid log(\frac{a}{v}) \leq X \leq log(\frac{b}{v})] \left[\Phi(\frac{log(\frac{b}{v})-t\mu}{\sigma\sqrt{t}}) - \Phi(\frac{log(\frac{a}{v})-t\mu}{\sigma\sqrt{t}})\right]$.

```
E.R.Expectation_part = C(mu=mu, sigma=sigma, K=K, t=t-s, v=v*exp(X_trunc))
E.R.CDF_part = pnorm( (log(b/v) - s*mu) / sigma / sqrt(s)
                      )- pnorm( (log(a/v) - s*mu) / sigma / sqrt(s) )
E.R <- E.R.Expectation_part * E.R.CDF_part
```

I will check the expected payoff of the option, and check the standard error

```
cat("Expected payoff of the barrier call option:", mean(E.R), "\n",
    "Standard error of payoff of the barrier call option:", sd(E.R)/sqrt(n))
```

```
## Expected payoff of the barrier call option: 2700.619
##  Standard error of payoff of the barrier call option: 1.649999
```

iii) (C) Use the usual average to estimate the expected payoff with the parameters in ii). Compare the standard error with that in ii).

From ii), we consider the case that the condition $a \leq P(s) \leq b$ is met. But, if that condition is not met, payoff of option is computed as $R = (P(t) - K)^+$. To get the usual average, we can express $P(t) = ve^{X+Y}$ where $X = log(\frac{P(s)}{v}) \sim N(s\mu, s\sigma^2)$, and $Y = log(\frac{P(t)}{P(s)}) \sim N((t-s)\mu, (t-s)\sigma^2)$

That means if the condition $a \leq P(s) \leq b$ is not met, the payoff is 0, and if the condition is met, the payoff is $(P(t) - K)^+ = (ve^{X+Y} - K)^+$

```
X <- qnorm(runif(n)) * sqrt(s * sigma^2) + s * mu
Y <- qnorm(runif(n)) * sqrt((t-s) * sigma^2) + (t-s) * mu

# if the condition is not met, payoff becomes zero.
# if the condition is met, payoff becomes v*exp(X+Y)-K
E.R.usual <- ifelse(X <= log(a/v) | X >= log(b/v), 0, v*exp(X+Y)-K)
```

I will check the expected payoff of the option, and check the standard error

```
# Check the estimated value and standard error
cat("Expected payoff of the barrier call option:", mean(E.R.usual), "\n",
    "Standard error of payoff of the barrier call option:", sd(E.R.usual)/sqrt(n))
```

```
## Expected payoff of the barrier call option: 2133.356
##  Standard error of payoff of the barrier call option: 200.2138
```

From the result of Q2. ii) and iii), we can say that the standard error of ii) is less than standard error of iii). Therefore, It can be concluded that the prediction in ii) is more accurate than the prediction in ii).