FE 515 Assignment 4

Akshay Pradeep Patade

2023-04-22

```
#Question 1.1
Sigma <- matrix(c(0.01, 0.002, 0.001,
                  0.002, 0.011, 0.003,
                  0.001, 0.003, 0.02), nrow=3, ncol=3)

# Define the objective function
f <- function(x) {
  return(0.5 * t(x) %*% Sigma %*% x)
}

# Define the constraints
ui <- matrix(c(-0.0427, -0.0015, -0.0285,
               1, 1, 1), nrow=2, ncol=3)

ci <- c(-0.05, 1.1)


set.seed(123)
n <- ncol(ui)
theta <- runif(n, min = 0, max = 1)
# Find optimal portfolio weights and value
result <- constrOptim(theta=theta, f, ui=ui, ci=ci, grad = NULL)
print(result$par)
```

```
## [1] -0.1869506  0.7552098  0.3445098
```

```
print(result$value)
```

```
##             [,1]
## [1,] 0.004932254
```

```
#Question 1.2

if (!requireNamespace("quadprog", quietly = TRUE)) {
  install.packages("quadprog")
}
library(quadprog)

# Define D and d
D <- 2 * Sigma
```

```
d <- rep(0, 3)

# Define A and b
A <- matrix(c(-0.0427, 1,
              -0.0015, 1,
              -0.0285, 1), nrow=3, ncol=2)

b <- c(-0.05, 1.1)

# Find optimal portfolio weights and value
result <- solve.QP(D, d, A, b)
result$solution
```

```
## [1]  0.74164557 -0.01779496  0.35784728
```

```
result$value
```

```
## [1] 0.008504752
```

```
#Question 2.1
MC_sim = function(Type, S, K, T , r , sigma, n , m ) {

  S0 = S
  dt = T/ n
  sum = 0
  for(i in 1:m)
  {
    S=S0
    for(j in 1:n) {
      E = rnorm(1, 0,1);
      S = S + r * S * dt + sigma * S * sqrt(dt) * E
    }

    if(Type == "c") {

      payoff = max(S-K,0)
    }
    else if(Type == "p") {

        payoff = max(K-S,0)
    }
    else {

      payoff = max(S-K,0)
    }
    sum = sum + payoff
  }
  Option_Price = (sum/m) * exp(-r * T);
  return(Option_Price)
}
Put_MC <- MC_sim("p", 100, 100, 1, 0.05, 0.2, 252, 10000)
Put_MC
```

```
## [1] 5.545524
```

```
PutPrice_MC <- MC_sim("c", 100, 100, 1, 0.05, 0.2, 252, 10000)
PutPrice_MC
```

```
## [1] 10.38247
```

```
#Question 2.2
BSM <- function(S, K, sigma, r, q, t, type) {

  a <- r - q
  d1 <- (log(S / K) + (a + sigma^2 / 2) * t) / (sigma * sqrt(t))
  d2 <- d1 - sigma * sqrt(t)
  if(type == "call") {
    price <- S * exp((a - r) * t) * pnorm(d1) - K * exp(-r * t) * pnorm(d2)
  }
  else if (type == "put") {
    price <-(K * exp(-r * t) * pnorm(-d2) - S * exp((a - r) * t) * pnorm(-d1))
  }
  return(price)
}
BSM(100, 100, 0.2, 0.05, 0, 1, type = "put")
```

```
## [1] 5.573526
```

```
BSM(100, 100, 0.2, 0.05, 0, 1, type = "call")
```

```
## [1] 10.45058
```