IE522_Project

2024-11-26

```r
EuropeanPut <- function(n, alpha){
  S0 = 100 # stock price
  K = 100 # strike price
  t = 0.5 # time to maturity
  r = 0.04 # risk free rate
  q = 0.02 # dividend yield
  sigma = 0.2 # implied volatility

  start_time <- Sys.time()

  Z = rnorm(n)
  S_T = S0*exp((r - q - 1/2*sigma^2)*t + sigma*sqrt(t)*Z[1])
  p = exp(-r*t) * max(0, K - S_T)

  x_bar = p
  y_bar = p^2

  for (k in 2:n){
    S_T = S0*exp((r - q - 1/2*sigma^2)*t + sigma*sqrt(t)*Z[k])
    p = exp(-r*t) * max(0, K - S_T)

    x_bar = (1 - 1/k)*x_bar + 1/k*p
    y_bar = (1 - 1/k)*y_bar + 1/k*p^2
  }

  se <- sqrt((y_bar - x_bar^2) / (n - 1))

  z_alpha <- qnorm(1 - alpha/2)
  lower_bound <- x_bar - z_alpha * se
  upper_bound <- x_bar + z_alpha * se

  d1 <- (log(S0/K) + (r - q + 1/2*sigma^2)*t) / (sigma * sqrt(t))
  d2 <- (log(S0/K) + (r - q - 1/2*sigma^2)*t) / (sigma * sqrt(t))
  exact_put_price <- K * exp(-r*t) * pnorm(-d2) - S0 * exp(-q*t) * pnorm(-d1)

  absolute_price_error = abs(x_bar - exact_put_price)

  end_time <- Sys.time()
  comp_time = as.numeric(difftime(end_time, start_time, units = "secs"))

  result_row <- data.frame(
    "Sample Size" = n,
    "Option Price" = x_bar,
    "Standard Error" = se,
    "Lower CI" = lower_bound,
    "Upper CI" = upper_bound,
    "Exact Put Price" = exact_put_price,
    "Absolute Error" = absolute_price_error,
    "Time (seconds)" = comp_time
  )
  return(result_row)
}
```

```
n = 10000
alpha = 0.05
EuropeanPut(n, alpha)
```

```
##   Sample.Size Option.Price Standard.Error Lower.CI Upper.CI Exact.Put.Price
## 1       10000     5.125489      0.0726787 4.983042 5.267937        5.074637
##   Absolute.Error Time..seconds.
## 1     0.05085277    0.009341002
```

```
convergence_table <- function(sample_sizes, alpha){
  results <- data.frame(matrix(ncol = 8, nrow = 0))
  colnames(results) <- c("Sample Size", "Option Price", "Standard Error", "Lower CI", "Upper CI", "Exac

  for (N in sample_sizes){
    result <- EuropeanPut(N, alpha)
    results <- rbind(results, result)
  }
  return(results)
}

# Define N values and alpha
sample_sizes <- c(10000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 100000000)
alpha <- 0.05

# Generate the table
table <- convergence_table(sample_sizes, alpha)
print(table)
```

```
##   Sample.Size Option.Price Standard.Error Lower.CI Upper.CI Exact.Put.Price
## 1       1e+04     5.155129   0.0723272565 5.013371 5.296888        5.074637
## 2       1e+05     5.033395   0.0225963553 4.989107 5.077683        5.074637
## 3       5e+05     5.069516   0.0101555755 5.049612 5.089421        5.074637
## 4       1e+06     5.071519   0.0071873011 5.057433 5.085606        5.074637
## 5       5e+06     5.077679   0.0032125577 5.071382 5.083975        5.074637
## 6       1e+07     5.078916   0.0022738507 5.074459 5.083373        5.074637
## 7       5e+07     5.073734   0.0010158841 5.071743 5.075725        5.074637
## 8       1e+08     5.074488   0.0007183803 5.073080 5.075896        5.074637
##   Absolute.Error Time..seconds.
## 1   0.0804927927     0.00918889
## 2   0.0412414313     0.11053514
## 3   0.0051200918     0.35793900
## 4   0.0031171601     0.72282696
## 5   0.0030422499     3.51090097
## 6   0.0042794329    6.95701718
## 7   0.0009025188   40.25287509
## 8   0.0001488399   81.43364501
```

```
EuropeanPut <- function(n, alpha) {
  S0 = 100   # stock price
  K = 100    # strike price
  t = 0.5    # time to maturity
  r = 0.04   # risk-free rate
  q = 0.02   # dividend yield
  sigma = 0.2   # implied volatility

  start_time <- Sys.time()

  x_bar = 0
  y_bar = 0

  for (k in 1:(n/2)) {
    Z1 = rnorm(1)
    Z2 = -Z1

    S_T1 = S0*exp((r - q - 1/2*sigma^2)*t + sigma*sqrt(t)*Z1)
    S_T2 = S0*exp((r - q - 1/2*sigma^2)*t + sigma*sqrt(t)*Z2)

    p1 = exp(-r*t) * max(0, K-S_T1)
    p2 = exp(-r*t) * max(0, K-S_T2)

    x = (p1 + p2) / 2

    x_bar = (1 - 1/k)*x_bar + 1/k*x
    y_bar = (1 - 1/k)*y_bar + 1/k*x^2
  }

  se <- sqrt((y_bar - x_bar^2) / (n/2 - 1))

  z_alpha <- qnorm(1 - alpha/2)
  lower_bound <- x_bar - z_alpha * se
  upper_bound <- x_bar + z_alpha * se

  d1 <- (log(S0/K) + (r - q + 1/2*sigma^2)*t) / (sigma * sqrt(t))
  d2 <- (log(S0/K) + (r - q - 1/2*sigma^2)*t) / (sigma * sqrt(t))
  exact_put_price <- K * exp(-r*t) * pnorm(-d2) - S0 * exp(-q*t) * pnorm(-d1)

  absolute_price_error = abs(x_bar - exact_put_price)

  end_time <- Sys.time()
  comp_time = as.numeric(difftime(end_time, start_time, units = "secs"))

  result_row <- data.frame(
    "Sample Size" = n,
    "Option Price" = x_bar,
    "Standard Error" = se,
    "Lower CI" = lower_bound,
    "Upper CI" = upper_bound,
    "Exact Put Price" = exact_put_price,
    "Absolute Error" = absolute_price_error,
    "Time (seconds)" = comp_time
  )
  return(result_row)
}
```

```
convergence_table <- function(sample_sizes, alpha){
  results <- data.frame(matrix(ncol = 8, nrow = 0))
  colnames(results) <- c("Sample Size", "Option Price", "Standard Error", "Lower CI", "Upper CI", "Exac

  for (N in sample_sizes){
    result <- EuropeanPut(N, alpha)
    results <- rbind(results, result)
  }
  return(results)
}

# Define N values and alpha
sample_sizes <- c(10000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 100000000)
alpha <- 0.05

# Generate the table
table <- convergence_table(sample_sizes, alpha)
print(table)
```

```
##   Sample.Size Option.Price Standard.Error Lower.CI Upper.CI Exact.Put.Price
## 1       1e+04     5.093351   0.0507348062 4.993912 5.192789        5.074637
## 2       1e+05     5.090981   0.0160958033 5.059433 5.122528        5.074637
## 3       5e+05     5.071349   0.0072009894 5.057235 5.085463        5.074637
## 4       1e+06     5.075103   0.0050889796 5.065128 5.085077        5.074637
## 5       5e+06     5.073628   0.0022750831 5.069169 5.078087        5.074637
## 6       1e+07     5.073470   0.0016075288 5.070319 5.076620        5.074637
## 7       5e+07     5.075666   0.0007194381 5.074256 5.077077        5.074637
## 8       1e+08     5.074682   0.0005085184 5.073685 5.075679        5.074637
##   Absolute.Error Time..seconds.
## 1   1.871409e-02     0.01142907
## 2   1.634401e-02     0.12812901
## 3   3.287686e-03     0.66861105
## 4   4.660979e-04     1.38294387
## 5   1.008458e-03     6.95437312
## 6   1.166782e-03    13.97020602
## 7   1.029918e-03    69.18065691
## 8   4.532518e-05   137.83861208
```