Project 2

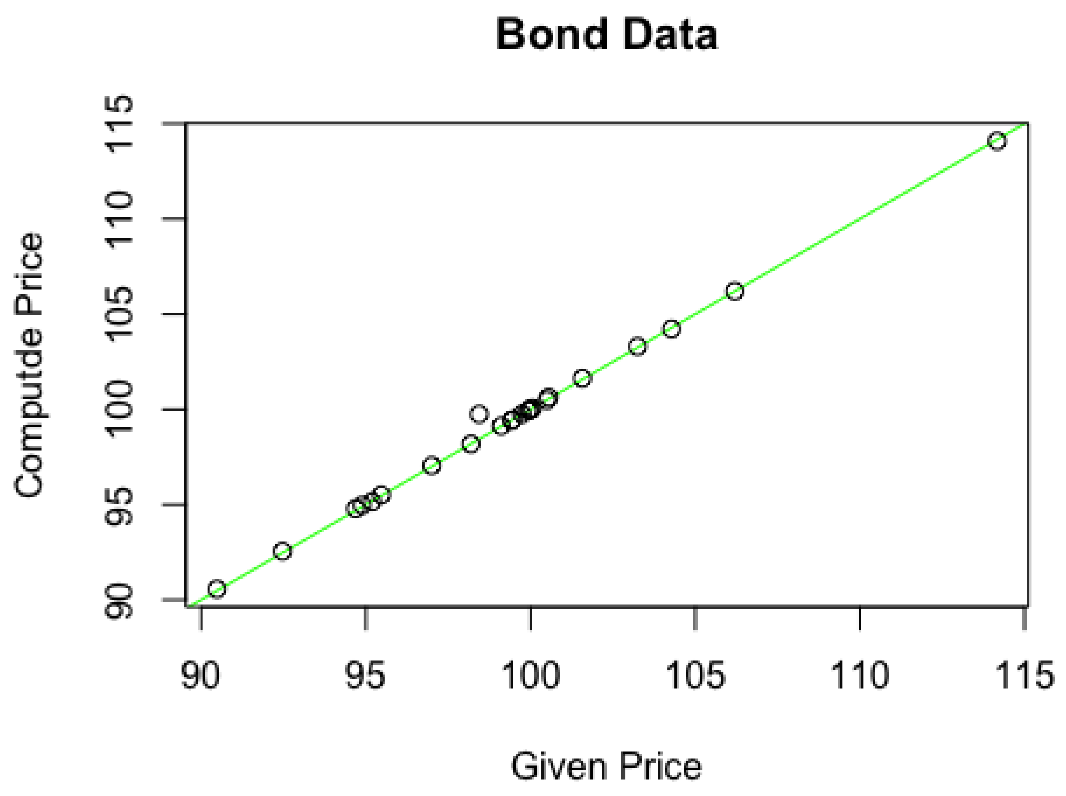
Atharva Rane, Mrinal Gupta

2024-04-26

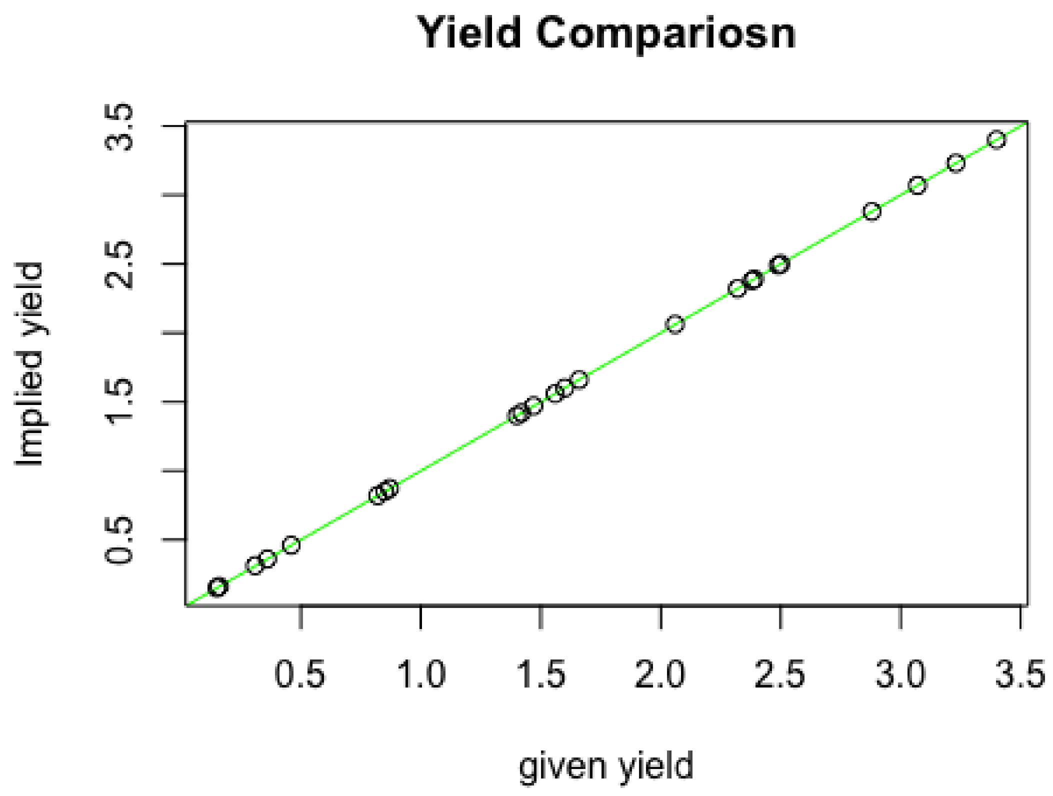
```r
library(quantmod)

library(dplyr)

symbols <- c("DGS2", "DGS5", "DGS10", "DGS30")
start_date <- as.Date("2018-01-02")
end_date <- as.Date("2022-03-31")

getSymbols(symbols, src = 'FRED', from = start_date, to = end_date)

## [1] "DGS2"  "DGS5"  "DGS10" "DGS30"

data_list <- lapply(symbols, function(sym) {
  data <- get(sym)
  data <- fortify.zoo(data)
  colnames(data) <- c("DATE", sym)
  return(data)
})

data <- Reduce(function(x, y) full_join(x, y, by = "DATE"), data_list)

data <- na.omit(data)
bond <- read.csv("/Users/atharvarane/Downloads/bonds.csv")


# Set the tolerance levels
tol1 <- 0.02
tol2 <- 0.025
tol3 <- 0.05
tol4 <- 0.015

# Find the dates where yields are within the specified tolerance levels
period1 <- data[abs(data$DGS2 - 2.88) <= tol1 & abs(data$DGS5 - 3.07) <= tol1
&
                abs(data$DGS10 - 3.23) <= tol1 & abs(data$DGS30 - 3.40) <=
tol1, ]

period2 <- data[abs(data$DGS2 - 1.47) <= tol2 & abs(data$DGS5 - 1.40) <= tol2
&
                abs(data$DGS10 - 1.56) <= tol2 & abs(data$DGS30 - 2.06) <=
tol2, ]
```

```
period3 <- data[abs(data$DGS2 - 0.31) <= tol3 & abs(data$DGS5 - 0.46) <= tol3
&
                abs(data$DGS10 - 0.85) <= tol3 & abs(data$DGS30 - 1.42) <=
tol3, ]

period4 <- data[abs(data$DGS2 - 0.15) <= tol4 & abs(data$DGS5 - 0.36) <= tol4
&
                abs(data$DGS10 - 0.82) <= tol4 & abs(data$DGS30 - 1.60) <=
tol4, ]
period4 <- period4[3, ]

period5 <- data[abs(data$DGS2 - 0.16) <= tol4 & abs(data$DGS5 - 0.87) <= tol4
&
                abs(data$DGS10 - 1.66) <= tol4 & abs(data$DGS30 - 2.32) <=
tol4, ]
period5 <- period5[1, ]

period6 <- data[abs(data$DGS2 - 2.38) <= tol3 & abs(data$DGS5 - 2.50) <= tol3
&
                abs(data$DGS10 - 2.39) <= tol3 & abs(data$DGS30 - 2.49) <=
tol3, ]

# Combine all periods into one data frame
periods_combined <- rbind(period1, period2, period3, period4, period5,
period6)

# Print the result
print(periods_combined)

##            DATE DGS2 DGS5 DGS10 DGS30
## 199  2018-10-05 2.88 3.07  3.23  3.40
## 460  2019-10-07 1.46 1.38  1.56  2.05
## 583  2020-03-26 0.30 0.51  0.83  1.42
## 744  2020-11-06 0.16 0.36  0.83  1.60
## 851  2021-04-06 0.16 0.88  1.67  2.32
## 1106 2022-03-29 2.35 2.49  2.41  2.53

# Part 2 -

coupon <- c(2.75, 2.88, 2.88, 3.00, 1.50, 1.50, 1.63, 2.25, 1.13, 1.13, 1.50,
2.00, 0.13, 0.25, 0.63, 1.38, 0.13, 0.75, 1.13, 1.88, 2.25, 2.50, 1.88, 2.25)
coupon <- coupon / 100
price <- c(99.74, 99.11, 97.00, 92.47, 100.06, 100.50, 100.55, 104.28,
101.57, 103.25, 106.20, 114.17, 99.95, 99.45, 98.19, 94.69, 99.93, 99.42,
95.19, 90.48, 98.44, 100.00, 95.47, 94.89)
yield <- c(2.88, 3.07, 3.23, 3.40, 1.47, 1.40, 1.56, 2.06, 0.31, 0.46, 0.85,
1.42, 0.15, 0.36, 0.82, 1.60, 0.16, 0.87, 1.66, 2.32, 2.38, 2.50, 2.39, 2.49)
yield <- yield / 100
maturity <- c(2, 5, 10, 30, 2, 5, 10, 30, 2, 5, 10, 30, 2, 5, 10, 30, 2, 5,
```

```r
10, 30, 2, 5, 10, 30)
length(maturity)

## [1] 24

data_0 <- data.frame(coupon,price,yield,maturity)
names(data_0) <- c("Coupon", "Price", " Yield", "Maturity")
data_0 <- data.matrix(data_0)


computed_price <- function(yield_0,coupon_0,FV_0,maturity_0){
  cash_flow <- c(rep(FV_0 * coupon_0, maturity_0 - 1), FV_0 * (1 + coupon_0))
  cash_flow <- data.frame(cash_flow)
  cash_flow$time <- as.numeric(rownames(cash_flow))
  cash_flow$present_value_fac <- 1 / (1 + yield_0)^cash_flow$time
  cash_flow$pv <- cash_flow$cash_flow * cash_flow$present_value_fac
  sum(cash_flow$pv)
}

price_0 <- c()

for(i in 1:24){

  price_0[i] <- computed_price(yield_0 = yield[i], coupon_0 = coupon[i], FV_0
= 100, maturity_0 = maturity[i])
}




plot(x = price, y = price_0, xlab = "Given Price", ylab = "Computde Price",
main ="Bond Data",
     abline(0,1, col = "green"))
```

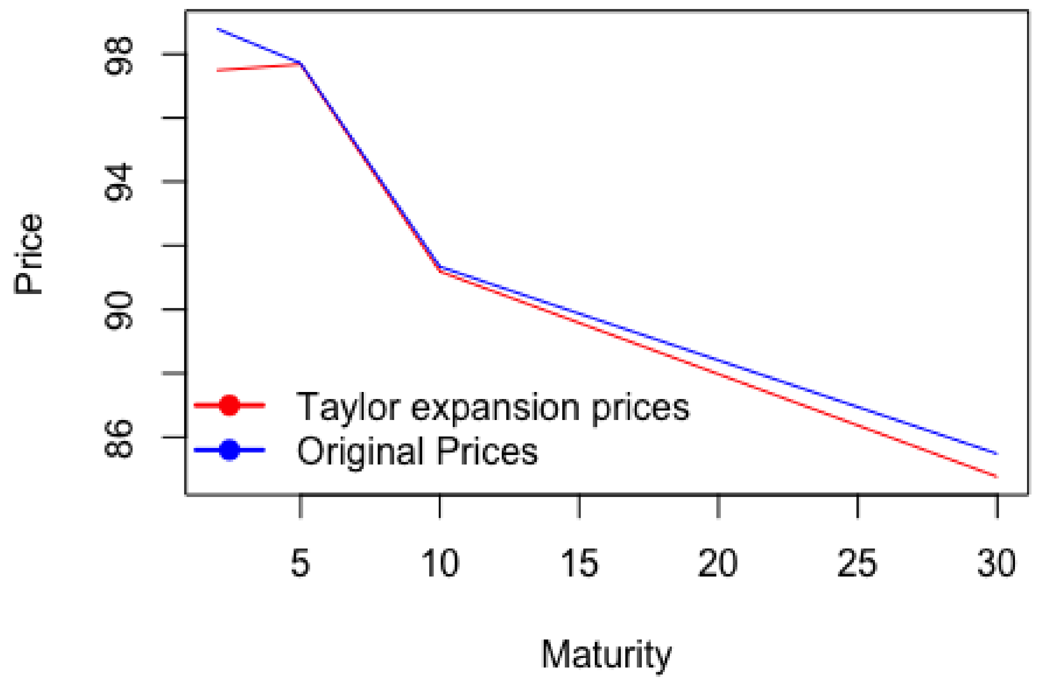


```r
# part 3

imp_yield <- c()

yield_f <- function(yield_1) {
  b_p <- sum(coupon_1 * (1 / (1 + yield_1))^(1:maturity_1)) + (FV_1 / (1 +
yield_1)^maturity_1) - computed_price_1

}


for(i in 1:24){
  coupon_1 <- coupon[i] * 100
  computed_price_1 <- price_0[i]
  maturity_1 <- maturity[i]
  FV_1 <- 100
  solve_yield <- uniroot(yield_f, c(0, 1))
  imp_yield <- c(imp_yield, solve_yield$root)
}

plot(x = yield *100 ,y = imp_yield*100, xlab = "given yield", ylab = "Implied
yield",main = "Yield Compariosn",
     abline(0,1,col = "green"))
```



```
# Part 4

# Creating the Macaulay Duration Function
calculate_md <- function(coupon_2, yield_2, maturity_2, face_value_2){
  cf <- c(rep(face_value_2 * coupon_2, maturity_2 - 1), face_value_2 * (1 +
coupon_2))
  cf <- data.frame(cf)
  cf$t <- as.numeric(rownames(cf))
  cf$pv_factor <- 1 / (1 + yield_2)^cf$t
  cf$pv <- cf$cf * cf$pv_factor
  cf$pv_tw_cf <- cf$t * cf$pv
  sum_pv <- sum(cf$pv)
  sum_pv_tw_cf <- sum(cf$pv_tw_cf)
  mac_d <- sum_pv_tw_cf / sum_pv
  return(mac_d)
}

md <- c()

for(i in 1:24){
  md[i] <- calculate_md(coupon_2 = coupon[i], yield_2 = yield[i], maturity_2
= maturity[i], face_value_2 = 100)
}



# Applying Macaulay Duration function to each bond
B_18 <- matrix(c('1', '2', '3', '4', md[1:4]), nrow = 2, ncol = 4, byrow =
TRUE)
B_19 <- matrix(c('5', '6', '7', '8', md[5:8]), nrow = 2, ncol = 4, byrow =
TRUE)
B_20 <- matrix(c('9', '10', '11', '12', md[9:12]), nrow = 2, ncol = 4, byrow
= TRUE)

B_21 <- matrix(c('13', '14', '15', '16', md[13:16]), nrow = 2, ncol = 4,
byrow = TRUE)
B_22 <- matrix(c('17', '18', '19', '20', md[17:20]), nrow = 2, ncol = 4,
byrow = TRUE)
B_23 <- matrix(c('21', '22', '23', '24', md[21:24]), nrow = 2, ncol = 4,
byrow = TRUE)


# Creating a dataframe that contains duration values for each period
Bond_duration <- data.frame(B_18[2,], B_19[2,], B_20[2,], B_21[2,], B_22[2,],
B_23[2,])
colnames(Bond_duration) <- c("First Period", "Second Period", "Third Period",
"Fourth Period", "Fifth Period", "Sixth Period")



# Function to calculate min and max for each row (Maturity)
calculate_min_max <- function(d) {
  result <- data.frame(
    Min = apply(d, 1, min),
    Max = apply(d, 1, max)
  )
  return(result)
}

result_df <- calculate_min_max(Bond_duration)
rownames(result_df) <- c("1 Year", "5 Year", "10 Year", "30 Year")
print(result_df)

##                       Min              Max
## 1 Year  1.97320305514723 1.99870142881729
## 5 Year  4.72640182873713 4.97504307951172
## 10 Year 8.80862193726395 9.71972588499073
## 30 Year 19.7587125451573 24.5930815523133
```

```r
# Part 5 -

duration <- c(1.973203,  4.726402,  8.808622, 19.758713,  1.985226, 4.854796,
9.311003, 22.318520,  1.988916,  4.891613,  9.383758, 23.377183, 1.998701,
4.975043,  9.719726, 24.593082,  1.998701,  4.925852, 9.496574, 22.787362,
1.977968,  4.761974,  9.186622, 21.893530)

bond <- cbind(bond,duration)

Price <- bond$Price
macaulay_du <- bond$duration
yield_3 <- bond$Yield


# Calculating Taylor Expansion
Taylor_21 <- Price[21] + ((-macaulay_du[21])/(1 + yield_3[21]/100)) *
Price[21] * (0.5/100)
Taylor_22 <- Price[22] + ((-macaulay_du[22])/(1 + yield_3[22]/100)) *
Price[22] * (0.5/100)
Taylor_23 <- Price[23] + ((-macaulay_du[23])/(1 + yield_3[23]/100)) *
Price[23] * (0.5/100)
Taylor_24 <- Price[24] + ((-macaulay_du[24])/(1 + yield_3[24]/100)) *
Price[24] * (0.5/100)
Taylor_ex_Prices <- c(Taylor_21, Taylor_22, Taylor_23, Taylor_24)
Taylor_ex_Prices

## [1] 97.48908 97.67709 91.18713 84.75498

# We will now calculate the original price after 50 bps increase

og_price <- c()


for(i in 1:24){
  og_price[i]<- computed_price(yield_0 = (yield[i]+0.005), coupon_0 =
coupon[i], FV_0 = 100, maturity_0 = maturity[i])
}

og_price[21:24]

## [1] 98.79241 97.71015 91.33593 85.47691

plot(x = maturity[21:24], y = Taylor_ex_Prices, xlab = "Maturity",
ylab="Price",
     type = "l", col = 'red', pch=19, ylim = range(c(Taylor_ex_Prices,
og_price[21:24])))
lines(x = maturity[21:24], y = og_price[21:24], type="l", col = "blue",
pch=19)
legend("bottomleft", legend = c("Taylor expansion prices", "Original
Prices"),
       col = c("red", "blue"), lty = 1, pch=19, lwd = 2, bty = "n")
```



```
# part 6 -

fund <- 100000
dur_4 <- 4
dur_8 <- 8

# Duration of 4 years
duration_data<- matrix(NA, 2, 2)
duration_data[1, 1:2] <- as.numeric(B_23[2, 1:2]) - dur_4
duration_data[2, 1:2] <- Price[21:22]
rownames(duration_data) <- c("Macaulay Duration", "Price")
Units_matrix <- matrix(NA, 2, 1)
Units_matrix[1:2] <- c(0, fund)

# Calculating weights of each bond
Units_Allocation <- solve(duration_data) %*% Units_matrix
colnames(Units_Allocation) <- c("Units of Bonds 21/22 for 4 years duration")
Units_Allocation
```

```
##      Units of Bonds 21/22 for 4 years duration
## [1,]                                  274.8706
## [2,]                                  729.4174

# Duration of 8 years

duration_data_0<- matrix(NA, 2, 2)
duration_data_0[1, 1:2] <- as.numeric(B_23[2, 1:2]) - dur_8
duration_data_0[2, 1:2] <- Price[21:22]
rownames(duration_data_0) <- c("Macaulay Duration", "Price")
Units_matrix_0 <- matrix(NA, 2, 1)
Units_matrix_0[1:2] <- c(0, fund)

# Calculating weights of each bond
Units_Allocation_0 <- solve(duration_data_0) %*% (Units_matrix_0)
colnames(Units_Allocation_0) <- c("Units of Bonds 21/22 for 8 years
duration")
Units_Allocation_0

##      Units of Bonds 21/22 for 8 years duration
## [1,]                                 -1142.354
## [2,]                                  2124.534

# Part 7 bonus qus -

library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

setup_matrix <- matrix(NA, 2, 4)
setup_matrix[1, 1:4] <- as.numeric(B_23[2, 1:4]) - dur_8
setup_matrix[2, 1:4] <- Price[21:24]
rownames(setup_matrix) <- c("Macaulay Duration", "Price")
colnames(setup_matrix) <- c("Bond 21", "Bond 22", "Bond 23", "Bond 24")
bond_allocation <- ginv(setup_matrix) %*% Units_matrix
colnames(bond_allocation) <- "Allocation of Bonds 21/22/23/24"
rownames(bond_allocation) <- c("Bond 21", "Bond 22", "Bond 23", "Bond 24")
bond_allocation

##         Allocation of Bonds 21/22/23/24
## Bond 21                        303.1917
## Bond 22                        291.2466
## Bond 23                        253.1679
## Bond 24                        177.6711
```

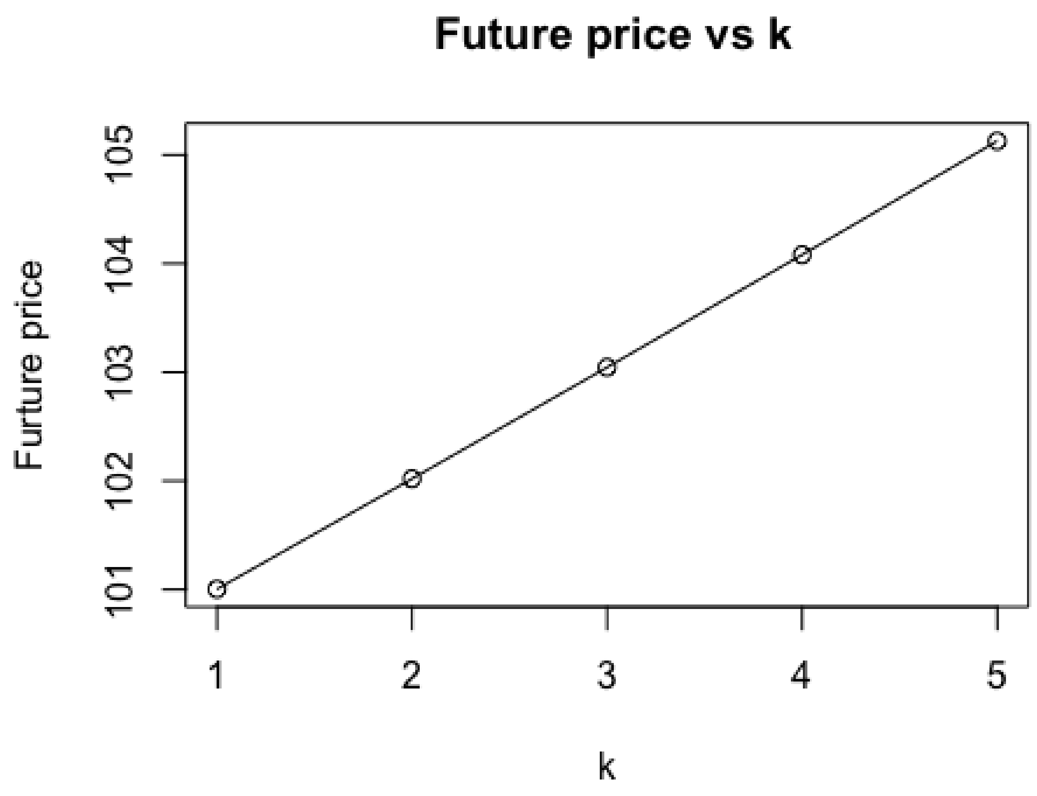
Qus 1.4 - Bonds with longer maturities have clearly longer durations, as seen in the columns; this is because the final cash flows (face value) are paid later and the average length is more affected by the bond's present value. We record the minimum and maximum range in the rows, which are determined by the various bond attributes (e.g., coupons)."The range widens with maturity: this is because a small change in the coupon rate implies a change in many more cash flows at higher maturities, which has a significant impact on the weights and consequently the durations.

Qus 1.5 - We notice that as the interest rates are up the prices go down. Although this not always true because duration is the measure of exposure to the interest rates which is risk and the volatility. We see that higher the duration, higher the bon price canvary when interest rates vary. The same way if maturity is longer the duration is higher.

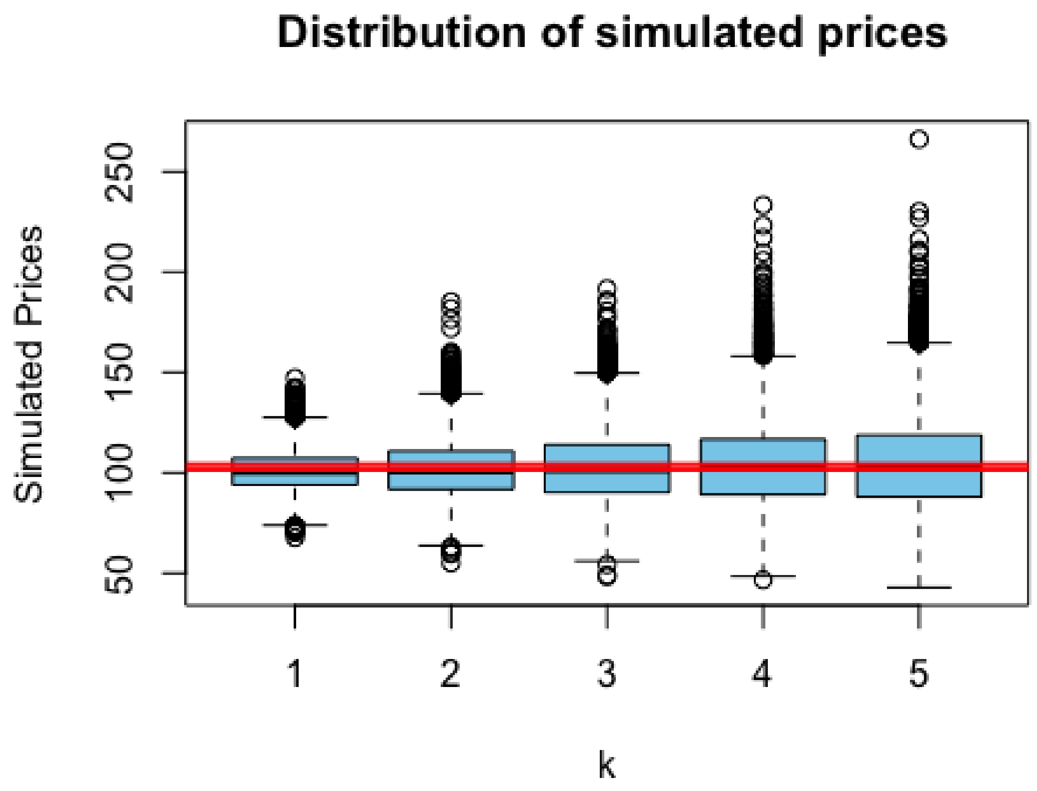
Qus 1.6 - When we limit the portfolio to 4 years using only 2 bonds which are bond 21 and 22, bond21 should be allocated with 274.8706 units where as for bond22 it is 729.4174. We see that bond 21 has duration of around 1.9 and bond 22 has duration of 4.7, to achieve the target of 4 years we allocate most of the fund to bond 22 and around 27 percent to bond 21.

Here we are trying to allocate in duration of 8 using both the bonds. We observe that both the bonds doesnt satisfy the desired duration. We short the bond with lowest duration which is bond 21 with -1142.3 and buying 2124.5 using the leverage effect to allocate the fund.

```r
# Q2

# Part 1

r <- 0.01
s_0 <- 100
k <- seq(1,5)


fwd_price <- c()

for (i in k ){
  fwd_price_1 <- s_0 * exp(r * i)
  fwd_price <- c(fwd_price,fwd_price_1)
}
fwd_price

## [1] 101.0050 102.0201 103.0455 104.0811 105.1271

plot(k,fwd_price, type = "o", xlab = " k ", ylab = "Furture price", main = "
Future price vs k")
```



```
# part 2

n <- 10^4
sigma <- 0.1
gbm <- function(k){
  f_price <- s_0 * exp((r - 0.5 * sigma^2)*k + sigma * sqrt(k) * rnorm(n))
  return(f_price)
}

# Initialize the matrix to store simulated prices
sim_price <- matrix(NA, nrow = n, ncol = length(k))

# Simulate the prices for each time step
for (j in 1:length(k)){
  sim_price[,j] <- gbm(k[j])
}

future_p <- apply(sim_price,2,FUN = mean)
data.frame(future_p)

##   future_p
## 1 100.9616
## 2 102.0103
## 3 102.9072
## 4 104.1602
## 5 104.8055

boxplot(sim_price, col = "skyblue", xlab = "k", ylab = "Simulated Prices",
main = " Distribution of simulated prices" )
abline(h = future_p, col = "red")
```



```
# part 3 -

### a) For forward contract, we long the forward contract as we believe that
the underlying will be priced higher when we settle the contract and profit.

set.seed(10)
K <- 101.005
S_0 <- 100
N <- 10^3
periods <- 252
dt <- 1 / periods

drift <- (0.01 - (0.1^2) / 2)
# Monte Carlo simulation function
sim <- function(n) {
```

```r
  R_seq <- rnorm(periods, drift * dt, 0.1 * sqrt(dt))
  S <- S_0 * exp(c(0, cumsum(R_seq)))
  return(S)
}
S_mat <- sapply(1:N, sim)

S_1 <- data.frame(PL = S_mat[periods + 1,] - K)

meanPL <- mean(S_1$PL)
# Calculate Value at Risk (VaR) at 5% confidence level
Qtc <- quantile(S_1$PL, 0.05)
# Display the result
sprintf("The VaR(0.05) is %s", meanPL - Qtc)

## [1] "The VaR(0.05) is 14.970318245616"

sprintf("The Expected pnl is %s", meanPL)

## [1] "The Expected pnl is -0.300114550666257"

#b) For underlying asset we borrow s_0 at risk free rate r today and buy the
underlying. After 1 year we sell the stock ( hoping that underlying topped)
and give back whatever we owe to the lender

set.seed(10)
K <- 105.1271
S_0 <- 100
N <- 10^3
periods <- 1260
dt <- 1 / periods
drift <- (0.01 - (0.1^2) / 2)

sim <- function(n) {
  R_seq <- rnorm(periods, drift * dt, 0.1 * sqrt(dt))
  S <- S_0 * exp(c(0, cumsum(R_seq)))
  return(S)
}

S_mat <- sapply(1:N, sim)

S_1 <- data.frame(PL = S_mat[periods + 1, ] - K)


meanPL2 <- mean(S_1$PL)

Qtc2 <- quantile(S_1$PL, 0.01)
sprintf("The VaR(0.05) is %s", meanPL2 - Qtc2)

## [1] "The VaR(0.05) is 21.6923809373826"
```

```
sprintf("The Expected Pnl is %s", meanPL2)

## [1] "The Expected Pnl is -4.19645679578824"
```

Q 2.1 - Longer the maturity, higher will be the price of forward, observing the upaward trend in the underlying.

Q 2.2 - The no-arbitrage price is consitent with risk-neutral esitmation under market efficiency assumptions. We observe that monte carlo simulations had near similar results to the forward prices while we understand that more the number of simulations more accurate the results will be.

```
# Q3

# We will get the quotes for GBP/USD fwd contracts.

So <- 1.2273

fwd_price <-
read.csv('/Users/atharvarane/Downloads/FE535_Forward_Prices.csv')

fwd_price$mid <- (fwd_price$Ask + fwd_price$Bid) / 2

fwd_rates <- So + fwd_price$mid / 10^4

names(fwd_rates) <- fwd_price$Name

fwd_rates

##  GBPUSD 1M FWD  GBPUSD 2M FWD  GBPUSD 3M FWD  GBPUSD 4M FWD  GBPUSD 5M FWD
##       1.227505       1.227888       1.228134       1.228386       1.228718
##  GBPUSD 6M FWD  GBPUSD 7M FWD  GBPUSD 8M FWD  GBPUSD 9M FWD GBPUSD 10M FWD
##       1.228967       1.229260       1.229514       1.229729       1.229976
## GBPUSD 11M FWD  GBPUSD 1Y FWD
##       1.230222       1.230434

# 3.1
# Part a -

# Calculating theta using forward-looking approch-

theta <- log(fwd_rates/So)*12/1:12
theta

##  GBPUSD 1M FWD  GBPUSD 2M FWD  GBPUSD 3M FWD  GBPUSD 4M FWD  GBPUSD 5M FWD
##    0.002004233    0.002873914    0.002717239    0.002653434    0.002771316
##  GBPUSD 6M FWD  GBPUSD 7M FWD  GBPUSD 8M FWD  GBPUSD 9M FWD GBPUSD 10M FWD
##    0.002714689    0.002735533    0.002702892    0.002636247    0.002613627
## GBPUSD 11M FWD  GBPUSD 1Y FWD
##    0.002594632    0.002550318

# Part b -

# Reading the LIBOR rates CSV file
libor_rates <-
read.csv("/Users/atharvarane/Downloads/FE535_Libor_USD_GBP.csv")

# Filtering the data for the 13 Nov 2023
libor_rates <- libor_rates[libor_rates$Dates == '11/13/2023',]

# Calculating theta for different maturities based on LIBOR rates
theta_1M <- libor_rates['US0001M.Index'] / 100 - libor_rates['BP0001M.Index']
/ 100
theta_3M <- libor_rates['US0003M.Index'] / 100 - libor_rates['BP0003M.Index']
/ 100
theta_6M <- libor_rates['US0006M.Index'] / 100 - libor_rates['BP0006M.Index']
/ 100

# Creating a matrix for comparison of LIBOR and forward-looking theta values
theta_comparison <- matrix(c(theta_1M, theta_3M, theta_6M, theta[1],
theta[3], theta[6]), nrow=3, ncol=2, byrow=TRUE)
rownames(theta_comparison) <- c('Theta 1M', 'Theta 3M', 'Theta 6M')
colnames(theta_comparison) <- c('Libor', 'Forward-Looking')

# comparison matrix
theta_comparison

##          Libor       Forward-Looking
## Theta 1M 0.0123548   0.0028025
## Theta 3M 0.0109803   0.002004233
## Theta 6M 0.002717239 0.002714689

# Part c -
library(quantmod)

currency_pair <- "GBPUSD=X"

# Set the period for historical data retrieval
start <- "2018-01-01"
end <- "2022-04-03"

# Downloading the historical data for the currency pair from Yahoo Finance
historical_data <- getSymbols(currency_pair, from = start, to = end, src =
"yahoo", auto.assign = F)
```

```
# Cleaning the NA values from the downloaded data
data <- na.omit(historical_data)

# Calculate the log returns of the adjusted closing prices
log_returns <- na.omit(log(data$`GBPUSD=X.Adjusted` /
lag(data$`GBPUSD=X.Adjusted`)))

sig <- sd(log_returns) * sqrt(252)
sig

## [1] 0.08552515

# 3.2

# VaR for unhedged

gbm <- function(n) {
  st <- So * exp((theta[11] - 0.5 * sig^2) * 11/12 + sig * sqrt(11/12) *
rnorm(n))
  return(st)
}

st <- gbm(10^6)

Vt <- 1.25*10^6 * (st - So)

Expected_Vt <- mean(Vt)

VaR_Vt <- Expected_Vt - quantile(Vt, 0.01)

VaR_Vt

##       1%
## 270974.3

# 3.3
#Unitary Hedge
units <- 62500
contracts<- 20
quantity <- units * contracts

# a)

FO <- as.numeric(getSymbols("6BZ24.CME", src="yahoo", auto.assign =
FALSE)["2023-11-13"][,"6BZ24.CME.Adjusted"])

## Warning: 6BZ24.CME contains missing values. Some functions will not work
if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.
```

```
s1 <- st
F1 <- st*exp(theta[2]*2/12)
PnL <- quantity * (s1 - F1) - quantity * (So - FO)
Expected_PnL <- mean(PnL)

VaR_PnL <- Expected_PnL - quantile(PnL, 0.01)
data.frame(PnL=c(Expected_PnL, VaR_PnL), row.names=c("Expected PnL", "VaR 99%
PnL"))

##                    PnL
## Expected PnL 4013.1788
## VaR 99% PnL   151.5812

# b)

GBM <- function(n) {
  st <- So * exp((theta[10] - 0.5 * sig^2) * 10/12 + sig * sqrt(10/12) *
rnorm(n))
  return(st)
}

st <- GBM(10^6)
FO <- as.numeric(getSymbols("6BU24.CME", src = "yahoo", auto.assign =
F)["2023-11-13", "6BU24.CME.Adjusted"])

## Warning: 6BU24.CME contains missing values. Some functions will not work
if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

S1 <- st

GBM <- function(n) {
  st <- S1 * exp((theta[1] - 0.5 * sig^2) * 1/12 + sig * sqrt(1/12) *
rnorm(n))
  return(st)
}

S2 <- GBM(10^6)
PnL <- quantity * (S2 - S1) - quantity * (So - FO)
Expected_PnL <- mean(PnL)
VaR_PnL <- Expected_PnL - quantile(PnL, 0.01)
data.frame(PnL=c(Expected_PnL, VaR_PnL), row.names=c("Expected PnL", "VaR 99%
PnL"))

##                    PnL
## Expected PnL  4399.897
## VaR 99% PnL  87338.671
```

c -

We observe that the VaR is highest when we do not use futures to hedge, this suggests we are fully exposed to the exchnage rate fluctuations. During the unitary hedging we see varying results while hedging with sep 2024 and dec 2024 futures. We observe that VaR is much higher when we hedged with sep 2024 futures as compared to dec futures this is because when we hedge with dec futures we have stability from exhcange rate flactuations till the end of contract and in other case we are only safe till sep and as it is shorter time period.

```
# Part 4 -

# We will choose Inverse
# 1.FXB - Invesco CurrencyShares British Pound Sterling Trust
# 2.USDU - WisdomTree Bloomberg U.S. Dollar Bullish Fund
# 3.EUFX - ProShares Short Euro
# 4.EMB - iShares J.P. Morgan USD Emerging Markets Bond ETF
# 5.UDN - Invesco DB U.S. Dollar Index Bullish fund

start_date <- "2018-01-01"
end_date <- "2022-04-03"

FXB <- getSymbols('FXB', from = start_date, to = end_date, auto.assign = F)
USDU <- getSymbols('USDU', from = start_date, to = end_date, auto.assign = F)
EUFX <- getSymbols('EUFX', from = start_date, to = end_date, auto.assign = F)
EMB <- getSymbols('EMB', from = start_date, to = end_date, auto.assign = F)
UDN <- getSymbols('UDN', from = start_date, to = end_date, auto.assign = F)

# Calculate log returns for the ETFs
returns.FXB <- na.omit(log(FXB$FXB.Adjusted/lag(FXB$FXB.Adjusted)))
returns.USDU <- na.omit(log(USDU$USDU.Adjusted/lag(USDU$USDU.Adjusted)))
returns.EUFX <- na.omit(log(EUFX$EUFX.Adjusted/lag(EUFX$EUFX.Adjusted)))
returns.EMB <- na.omit(log(EMB$EMB.Adjusted/lag(EMB$EMB.Adjusted)))
returns.UDN <- na.omit(log(UDN$UDN.Adjusted/lag(UDN$UDN.Adjusted)))

returns <-
merge(log_returns,returns.FXB,returns.USDU,returns.EUFX,returns.EMB,returns.U
DN, join = 'inner')

## Warning in merge.xts(log_returns, returns.FXB, returns.USDU, returns.EUFX,
:
## 'join' only applicable to two object merges

# Fetch GBP/USD exchange rate data
GBPUSD <- getSymbols('GBPUSD=X', from = start_date, to = end_date,
auto.assign = F)

# Linear models to relate GBP/USD to each ETF's returns
model1 <- lm(GBPUSD.X.Adjusted  ~ FXB.Adjusted, data = returns)
```

```
model2 <- lm(GBPUSD.X.Adjusted  ~ USDU.Adjusted, data = returns)
model3 <- lm(GBPUSD.X.Adjusted  ~ EUFX.Adjusted, data = returns)
model4 <- lm(GBPUSD.X.Adjusted  ~ EMB.Adjusted, data = returns)
model5 <- lm(GBPUSD.X.Adjusted  ~ UDN.Adjusted, data = returns)

# Hedge Effectivness with different etfs-
# FXB
summary(model1)$r.squared

## [1] 0.2295911

# USDU
summary(model2)$r.squared

## [1] 0.05721054

# EUFX
summary(model3)$r.squared

## [1] 0.03790647

# EMB
summary(model4)$r.squared

## [1] 0.006023108

# UDN
summary(model5)$r.squared

## [1] 0.1431672
```

FXB - This fund is designed to track the price of the British Pound sterling. It's a direct play on the currency, meaning it moves in line with the pound's performance against the dollar. Having provided direct exposure to GBP/USD fluctuations and highest R -squared values proves to be better hedge with stronger correlation.

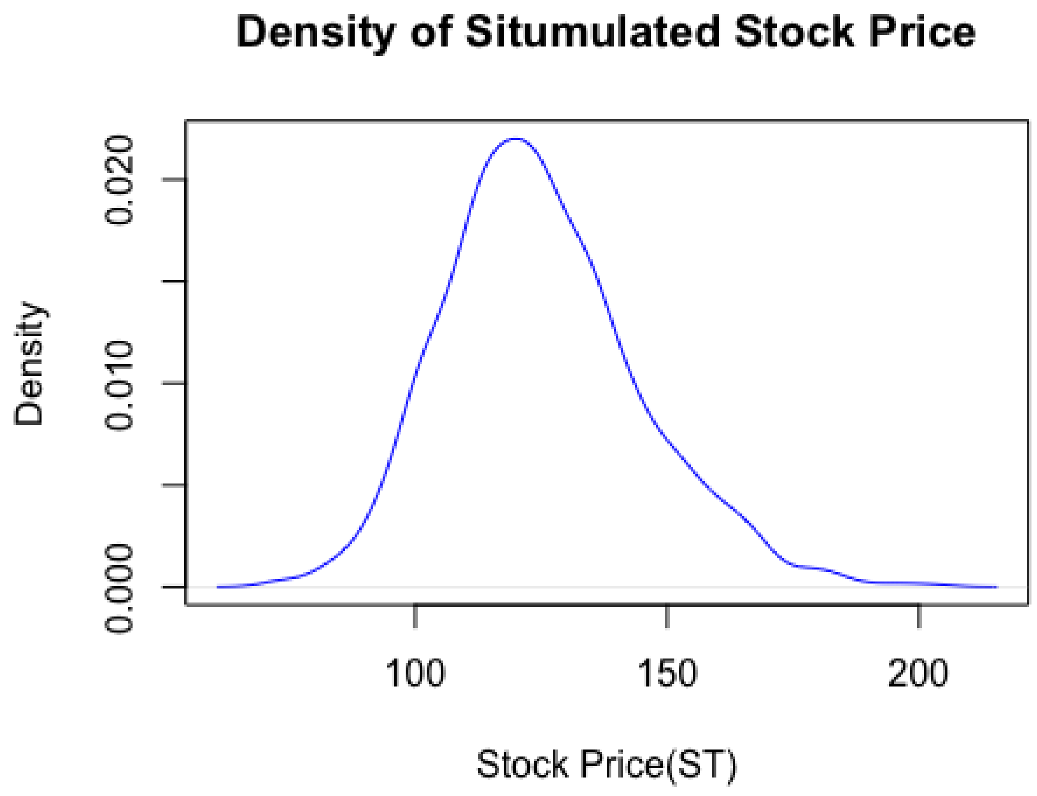
USDU - While this isn't a direct GBP/USD hedge, this fund tracks the U.S. dollar against a basket of currencies, including the GBP. When the dollar strengthens against this basket, the fund should increase in value, indirectly hedging against a weaker pound.

EUFX - This Etf is short euro and can provide a proxy hedge for GBP exposure if the movements in GBP/USD are partly due to broader USD strenght or weakness.

EMB - While not a direct currency play, this ETF is affected by USD strength. Since it holds dollar-denominated bonds from emerging markets, when the USD is strong (and thus GBP/USD is likely falling), this ETF tends to perform better as the underlying bond payments become more valuable.

UDN - This ETF invests in USD-denominated securities and aims to benifit from a rising USD.

```r
#Section 4
#Part 4.1.1

u <- 0.10
sig <- 0.15
S0 <- 120
r <- 0.03
k <- 125
T <- 1
D <- 0
N <- 1000

Z<- rnorm(N, mean=0, sd=1 )
ST <- S0* exp((r-(sig^2)*0.5)*T+ sig* sqrt(T) *Z)
plot(density(ST), main = "Density of Situmulated Stock Price", xlab = "Stock
Price(ST)", ylab="Density", col="blue")
```



```r
#Part 4.1.2


expected_value<- mean(ST)
variance<- var(ST)
cat("Expected value of S1 given S0 under Q", expected_value)
```

```
## Expected value of S1 given S0 under Q 124.9812

cat("Variance of S1 given S0 under Q", variance)

## Variance of S1 given S0 under Q 375.5688

# simulations vs true value

true_exp <- S0*exp(r)
true_var <- (exp(sig^2)-1)*S0^2*exp(2*r)

sim_true <- data.frame(Mean=c(expected_value,true_exp),
Var=c(variance,true_var))
rownames(sim_true)=c('Simulation', 'True Vlaues')
sim_true

##                 Mean      Var
## Simulation  124.9812 375.5688
## True Vlaues 123.6545 347.9346

#Part 4.2
#4.2.1

c<- exp(-r*T) *mean(pmax(ST-k,0))
c

## [1] 7.361394

#4.2.2
#using Black Scholes Model

d2<- (log(S0/k)+(r-0.5*sig^2)*T)/ sig*sqrt(T)
d1<- d2 + sig*sqrt(T)

c_bsm<- S0*pnorm(d1)-k*exp(-r*T)*pnorm(d2)
c_bsm

## [1] 6.579149
```

#4.2.3 We know that the BSM model provides a closed form solution for the option pricing while Monte Carlo model simulation uses numerical model. Though the results from both of these methods differ we obeserve that BSM model is accurate and we can gainapproximately same accuracy from MC simulations as well by higher number of simulations but at the cost of computational power and time.

```
#Part 4.3
#4.3.1
u <- 0.10
sig <- 0.15
S0 <- 120
r <- 0.03
```

```r
k <- 125
T <- 1
N <- 1000
M<-100


estimated_prices <- numeric(M)
# Loop for M experiments
for (m in 1:M) {

  Z <- rnorm(n=N, mean = 0, sd = 1)
  ST <- S0 * exp((r - (sig^2) / 2) * T + sig * sqrt(T) * Z)
  c_hat <- exp(-r * T) * mean(pmax((ST - k), 0))
  estimated_prices[m] <- c_hat
}

average_option_price <- mean(estimated_prices)
average_option_price

## [1] 6.593724

#section 4.3, part 2
# Calculate the Mean Squared Error (MSE)
MSE <- (estimated_prices - c_bsm)^2
MSE <- mean(MSE)
MSE

## [1] 0.1274784

# decomposition MSE

bias <- mean(estimated_prices - c_bsm)^2
var <- var(estimated_prices)
bias

## [1] 0.0002124259

var

## [1] 0.1285515

# section 4.4

# part 1 -
Z <- c(Z[1:500], -Z[1:500])
dRt_seq <- (r - sig^2/2)*T + sig*sqrt(T)*Z
S1_seq <- S0*exp(dRt_seq)
S1_exp <- mean(S1_seq)
S1_var <- var(S1_seq)
cbind(S1_exp, S1_var)
```

```
##         S1_exp   S1_var
## [1,] 123.7237 365.2304

# part 2  -

seq <- pmax(0,S1_seq - k)
call <- mean(seq)*exp(-r*T)
call

## [1] 6.788332

# part 3 -
# MSE
set.seed(123)
M <- 100
GBM_c <- function(n){
  Z <- rnorm(10^3)
  Z <- c(Z[1:500], -Z[1:500])
  dRt_seq <- (r - sig^2/2)*T + sig*sqrt(T)*Z
  S1 <- S0*exp(dRt_seq)
  c_seq <- pmax(0, S1 - k)
  c <- mean(c_seq)*exp(-r*T)
  c
  return(c)
}

c_vec <- sapply(1:M, GBM_c)
c_mean <- mean(c_vec)
c_mean

## [1] 6.535805

sq_error <- (c_vec - c_bsm)^2
MSE <- mean(sq_error)
MSE

## [1] 0.0758366

# decomposition MSE

bias <- mean(c_vec - c_bsm)^2
variance <- var(c_vec)
bias

## [1] 0.001878713

variance

## [1] 0.07470494
```

part 4 –

a) YES it reduced the MSE of the option price.

b) Antithetic variates reduce Mean Squared Error (MSE) by leveraging negatively correlated path pairs in simulations. This technique minimizes the variance of the estimated option prices from simulations, improving the estimate precision. By mitigating variance more than bias, it effectively lowers the overall MSE, making Monte Carlo simulations in option pricing more accurate.