Module 12.4: DRM ABM-Based Option Valuation Models
R Commentary

See module *Ch 12.4 DRM ABM-Based Option Models*.

R code comments

We now review the R code here. The critical inputs are contained in the ABM Option MCVaR Setup.R file whereas the results are stored in spreadsheets (VaR and Valuations CorStockVol.xlsx and MC VaR and RVaR CorStockVol.xlsx).

ABM OVM MCVaR Test.R

The Monte Carlo simulation is applied to ABM-based option valuation models through this test program. Given the computational demand are not as demanding as binomial based models.

```
# ABM OVM MCVaR Test.R
# NOTE: This code follows closely Module 12.1 GBM BOVM MCVaR Test.R
# The goal is to facilitate comparison.
rm(list = ls())  # Take out the Environment "trash"
cat("\014") # Clear Console, making error checking easier.
while (!is.null(dev.list()))  dev.off() # Clear old plots
par(family = 'Times New Roman') # Globally set fonts for graphs
# Libraries
#  MASS - Multivariate normal random number generator
Packages <- c("MASS","openxlsx")
if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(Packages, rownames(installed.packages())))
} # Make sure libraries are installed on this computer
lapply(Packages, library, character.only=TRUE) # Load and attach libraries
rm(Packages)
# Build data frames of inputs
# Note:
#      w = weight of strike price
#      X = (1 - w)S, S, (1 + w)S of par (evaluate only three)
# 19 Strategies compared:
#   1) Long N stock
#   2) Long N calls (XL, X, XH)
#   3) Long N puts (XL, X, XH)
#   4) Covered call writing (long N stock, short N calls) (XL, X, XH)
#   5) Protective put buying (long N stock, long N puts) (XL, X, XH)
#   6) Leveraged call buying (long N stock, long N calls) (XL, X, XH)
#   7) Leveraged put selling (long N stock, short N puts) (XL, X, XH)
#
#
# Study #1: Role of correlation between Stock and Volatility
#
# Variable <- "CorStockVol"
Variable <- "CorRateVol"
LowerBound <- -0.9
UpperBound <- 0.9
NumberOfStrategies <- 19 # Long only for now
```

Once the particular variable to analyze is selected (the correlation between stock and volatility), the remaining inputs are read.

```
source('ABM Option MCVaR Setup.R')
```

Based on the variable select, the core function is made available (fABMBOVMMCVaRCore).

```
source('ABM OVM MCVaR Core.R')
# Sensitivity Variable
SensitivityVariable = data.frame(matrix(NA, nrow = NumberOfStrategies, ncol = 1))
colnames(SensitivityVariable)  <- 'Parameter'
rownames(SensitivityVariable)  <- seq(1:NumberOfStrategies)
for(i in 1:NumberOfStrategies){
  SensitivityVariable$Parameter[i] <- LowerBound + (i - 1)*Increment
```

```
}
Tab <- "Sensitivity Variable"
SensitivityVariable <- cbind(Parameters = rownames(SensitivityVariable),
SensitivityVariable)
rownames(SensitivityVariable) <- 1:nrow(SensitivityVariable)
wb <- buildWorkbook(SensitivityVariable, sheetName = Tab, gridLines = TRUE)
```
The initial base case is analyzed separately.
```
BASECASE <- FALSE
for(i in 1:NumberOfStrategies){
  SimulationInputData$CorStockVol <- LowerBound + (i - 1)*Increment
  tMCOutput <- fABMOVMMCVaRCore(ABMInputData, InitialValuations, Variable)
  MCVaROutput[,i] <- tMCOutput$VaR
  MCRVaROutput[,i] <- tMCOutput$RVaR
}
# VaR output based on sensitivities
MCVaROutput <- cbind(Strategy = rownames(MCVaROutput), MCVaROutput)
rownames(MCVaROutput) <- 1:nrow(MCVaROutput)
Tab <- "MCVaR Output"
addWorksheet(wb, sheetName = Tab, gridLines = TRUE)
writeData(wb, Tab, MCVaROutput)
# Return VaR output based on sensitivities
MCRVaROutput <- cbind(Strategy = rownames(MCRVaROutput), MCRVaROutput)
rownames(MCRVaROutput) <- 1:nrow(MCRVaROutput)
Tab <- "MCRVaR Output"
addWorksheet(wb, sheetName = Tab, gridLines = TRUE)
writeData(wb, Tab, MCRVaROutput)
WorkBookName <- paste0('ABM MC VaR and RVaR ', Variable, '.xlsx')
saveWorkbook(wb, WorkBookName, overwrite = TRUE)
```
The initial base case is analyzed separately.
```
BASECASE <- TRUE
Valuations <- fABMOVMMCVaRCore(ABMInputData, InitialValuations, Variable)
```

ABM Option MCVaR Setup.R

The file to parameterized both the option model and simulation. Recall the ABMOVM is calibrated to the GBMOVM based on these parameters.
```
# ABM Option MCVaR Setup.R
# Stock 1
inputName <- 'ABC'
inputUnits <- 100 # Number of units (e.g., Shares)
inputStockPrice1 <- 100.0        # In dollars per share
inputStrikePrice1 <- inputStockPrice1 # In dollars per share, at-the-money assumed
inputInterestRate <- 5.0         # In percent, annualized risk-free interest
rate,continuously compounded
inputDividendYield1 <- 0.0       # In percent, continuously compounded, annualized
inputImpliedVolatility1 <- 29.884768295208  # In percent, current implied volatility used
to compute current option prices
inputTimeToMaturity1 = 1         # In years, for both puts and calls
inputType <- 1L                  # 1 for call, -1 for put
inputStrikeIncrement <- 10.0   # Percentage change to compute XL and XH
#
# ABM list
#
ABMInputData <- list(inputName, inputUnits, inputStockPrice1, inputStrikePrice1,
  inputInterestRate, inputDividendYield1, inputImpliedVolatility1, inputTimeToMaturity1,
  inputType, inputStrikeIncrement)
names(ABMInputData) <- c("Name", "Units", "StockPrice", "StrikePrice", "InterestRate",
  "DividendYield", "Volatility", "TimeToMaturity", "Type", "StrikeIncrement")
OptionInputData <- as.data.frame(ABMInputData) # Used later to track inputs
# Stock 2 (eventually build out, or make assignment)

# Similiation input (only one stock for now)
inputHorizon <- 1/12 # Default (1 Week), must be less than TimeToMaturity
inputConfidenceLevel <- 95.0 # Percent
```

```
inputNumberOfSimulations <- 5000 # Default: 5,000
inputStockMean <- 5.0     # In percent, expected percentage change in stock, continuously
compounded, annualized
inputInterestRateMean <- 0.0    # In percent, expected percentage change in interest
rate, cc
inputSDMean <- 0.0    # In percent, expected percentage change in SD, cc
inputStockSD <- 30.0      # In percent, continuously compounded, annualized
inputInterestRateSD <- 10.0      # In percent, continuously compounded, annualized
inputSDSD <- 10.0      # In percent, continuously compounded, annualized
inputCorRateVol <- 0.0  # Correlation between changes in rates and changes in vol
inputCorStockRate <- -0.3 # Correlation between stock returns and changes in rates
inputCorStockVol <- -0.5  # Correlation between stock returns and changes in SD
SimulationInputData <- list(inputName, inputHorizon, inputConfidenceLevel,
  inputNumberOfSimulations, inputStockMean, inputInterestRateMean,
  inputSDMean, inputStockSD, inputInterestRateSD,
  inputSDSD, inputCorRateVol, inputCorStockRate, inputCorStockVol)
names(SimulationInputData) <- c("Name", "Horizon", "ConfidenceLevel",
  "NumberOfSimulations", "StockMean", "InterestRateMean",
  "SDMean", "StockSD", "InterestRateSD",
  "SDSD", "CorRateVol", "CorStockRate", "CorStockVol")
Increment <- (UpperBound - LowerBound)/(NumberOfStrategies - 1)
MCVaROutput = data.frame(matrix(NA, nrow = NumberOfStrategies, ncol =
NumberOfStrategies))
colnames(MCVaROutput)  <- seq(1:NumberOfStrategies)
rownames(MCVaROutput)  <- c("LS", # Long stock
  "LCXL", "LCX", "LCXH", # Long call
  "LPXL", "LPX", "LPXH", # Long put
  "LCCWXL", "LCCWX", "LCCWXH", # Long covered call writing (long stock, short call)
  "LPPBXL", "LPPBX", "LPPBXH", # Long protective put buying (long stock, long call)
  "LLCXL", "LLCX", "LLCXH", # Long leveraged call (long stock, long call)
  "LLPXL", "LLPX", "LLPXH" # Long leveraged put (Long stock, short put)
)
MCRVaROutput <- MCVaROutput
```

ABM OVM MCVaR Core.R

The core of the dynamic risk analysis is performed here complete with multivariate Monte Carlo simulation.

```
# ABM OVM MCVaR Core.R
fABMOVMMCVaRCore = function(ABMInputData, InitialValuations, Variable){
  Horizon = SimulationInputData$Horizon
  NumberOfSecurities <- 1 # Analysis of only one stock
  NumberOfRandomVariables <- 3.0 # Stock, Rates, Volatility
  ConfidenceLevel <- SimulationInputData$ConfidenceLevel
  # Extract Name, Units, Price, Mean, Standard Deviation, and Correlation inputs
  Name = SimulationInputData$Name
  Units = inputUnits
  Price = inputStockPrice1 # Just one stock for now
  Mean = as.numeric(c(1:NumberOfRandomVariables))
  Mean[1] <- SimulationInputData$StockMean/100.0
  Mean[2] <- SimulationInputData$InterestRateMean/100.0
  Mean[3] <- SimulationInputData$SDMean/100.0
  StandardDeviation = as.numeric(c(1:NumberOfRandomVariables))
  StandardDeviation[1] <- SimulationInputData$StockSD/100.0
  StandardDeviation[2] <- SimulationInputData$InterestRateSD/100.0
# SD IN $
  StandardDeviation[3] <- SimulationInputData$SDSD
  Correlation = matrix(data = 0, nrow = NumberOfRandomVariables, ncol =
NumberOfRandomVariables)
  Correlation[1,1] <- Correlation[2,2] <- Correlation[3,3] <- 1.0
  Correlation[1,2] <- Correlation[2,1] <- SimulationInputData$CorStockRate
  Correlation[1,3] <- Correlation[3,1] <- SimulationInputData$CorStockVol
  Correlation[2,3] <- Correlation[3,2] <- SimulationInputData$CorRateVol
  Covariance = matrix(nrow = NumberOfRandomVariables, ncol = NumberOfRandomVariables)
  Covariance[1,1] <- StandardDeviation[1]^2
  Covariance[2,2] <- StandardDeviation[2]^2
```

```
  Covariance[3,3] <- StandardDeviation[3]^2
  Covariance[1,2] <- Covariance[2,1] <- StandardDeviation[1] * StandardDeviation[2] *
Correlation[1,2]
  Covariance[1,3] <- Covariance[3,1] <- StandardDeviation[1] * StandardDeviation[3] *
Correlation[1,3]
  Covariance[2,3] <- Covariance[3,2] <- StandardDeviation[2] * StandardDeviation[3] *
Correlation[2,3]
# Data frame holding outputs
  InitialValuations = data.frame(matrix(NA, nrow = NumberOfStrategies, ncol =
NumberOfSecurities))
  colnames(InitialValuations)  <- c("InitialValue")
  rownames(InitialValuations)  <- c("LS", # Long stock
    "LCXL", "LCX", "LCXH", # Long call
    "LPXL", "LPX", "LPXH", # Long put
    "LCCWXL", "LCCWX", "LCCWXH", # Long covered call writing (long stock, short call)
    "LPPBXL", "LPPBX", "LPPBXH", # Long protective put buying (long stock, long call)
    "LLCXL", "LLCX", "LLCXH", # Long leveraged call (long stock, long call)
    "LLPXL", "LLPX", "LLPXH" # Long leveraged put (Long stock, short put)
  )
#
# Source all required functions
#
  source("ABM OVM MCVaR Functions.R")
#
# VaR analysis of European-style options
#
  inputStyle = 1  # 1 for European-style, 2 for American-style
  IV <- fInitialValues(ABMInputData, InitialValuations) # NOTE: This function will change
with option models
#
# Set up simulations
#
# Test normal distribution (Normal with mean = Mean and
#   standard deviation = StandardDeviation)
# Sample (s)
  NSim <- SimulationInputData$NumberOfSimulations
  s = mvrnorm(n = NSim, Mean, Covariance, tol = 1e-6, empirical = FALSE, EISPACK = FALSE)
  sMean = as.numeric(c(1:NumberOfRandomVariables))
  sStandardDeviation = as.numeric(c(1:NumberOfRandomVariables))
  sCorrelation = matrix(nrow = NumberOfRandomVariables, ncol = NumberOfRandomVariables)
  for(i in 1:NumberOfRandomVariables){
    sMean[i] = mean(s[,i])
    sStandardDeviation[i] = sd(s[,i])
    for(j in 1:i){
      sCorrelation[i, j] = cor(s[,i],s[,j])
      sCorrelation[j, i] = sCorrelation[i, j]
    }
  }
  TerminalValuations = data.frame(matrix(NA, nrow = NumberOfStrategies, ncol = NSim))
  colnames(TerminalValuations)  <- seq(1:NSim)
  rownames(TerminalValuations)  <- c("LS", # Long stock
    "LCXL", "LCX", "LCXH", # Long call
    "LPXL", "LPX", "LPXH", # Long put
    "LCCWXL", "LCCWX", "LCCWXH", # Long covered call writing (long stock, short call)
    "LPPBXL", "LPPBX", "LPPBXH", # Long protective put buying (long stock, long call)
    "LLCXL", "LLCX", "LLCXH", # Long leveraged call (long stock, long call)
    "LLPXL", "LLPX", "LLPXH" # Long leveraged put (Long stock, short put)
  )
  Horizon <- SimulationInputData$Horizon
  tStockPrice <- ABMInputData$StockPrice
  tInterestRate <- ABMInputData$InterestRate
  tVolatility <- ABMInputData$Volatility
  tTimeToMaturity <- ABMInputData$TimeToMaturity
  ABMInputData$TimeToMaturity <- tTimeToMaturity - Horizon
```

```
  for(i in 1:NSim){
    ABMInputData$StockPrice <- tStockPrice*exp(s[i,1]*Horizon)
    ABMInputData$InterestRate <- tInterestRate*exp(s[i,2]*Horizon)
    ABMInputData$Volatility <- tVolatility + s[i,3]*Horizon
    TerminalValuations[,i] <- fInitialValues(ABMInputData, InitialValuations)
  }
  ABMInputData$TimeToMaturity <- tTimeToMaturity
  ABMInputData$StockPrice <- tStockPrice
  ABMInputData$InterestRate <- tInterestRate
  ABMInputData$Volatility <- tVolatility
#
# Compute VaR for each strategy
#
  TerminalValue <- rep(-99, 19)
  Valuations <- cbind(IV, TerminalValue)
  VaR <- rep(-99, 19)
  Valuations <- cbind(Valuations, VaR)
  RVaR <- rep(-99, 19)
  Valuations <- cbind(Valuations, RVaR)
  for(i in 1:NumberOfStrategies){
    tVector <- as.matrix(TerminalValuations[i,])
    VaRTV <- as.numeric(quantile(tVector, 1-ConfidenceLevel/100))
    Valuations[i,2] <- as.numeric(VaRTV)
    Valuations[i,3] <- -(as.numeric(Valuations[i,2]) - as.numeric(Valuations[i,1]))
    Valuations[i,4] <- (Valuations[i,3]/Valuations[i,1])*100.0
  }
  Valuations <- round(Valuations,2)
# Record base case information
  if(BASECASE){
    # Valuations <- cbind(Strategy = rownames(Valuations), Valuations)
    # rownames(Valuations) <- 1:nrow(Valuations)
    # Tab <- "VaR Output"
    # wb <- buildWorkbook(Valuations, sheetName = Tab, gridLines = TRUE)
    Valuations <- cbind(Strategy = rownames(Valuations), Valuations)
    rownames(Valuations) <- 1:nrow(Valuations)
    Tab <- "VaR Output"
    wb <- buildWorkbook(Valuations, sheetName = Tab, gridLines = TRUE)
# Input option data
    Tab <- "Option Parameters"
    addWorksheet(wb, sheetName = Tab, gridLines = TRUE)
    OptionInputData <- t(OptionInputData)
    OptionInputData <- cbind(Strategy = rownames(OptionInputData), OptionInputData)
    rownames(OptionInputData) <- 1:nrow(OptionInputData)
    writeData(wb, Tab, OptionInputData)
# Input simulation data
    Tab <- "Simulation Parameters"
    addWorksheet(wb, sheetName = Tab, gridLines = TRUE)
    SimulationInputData <- as.data.frame(SimulationInputData)
    SimulationInputData <- t(SimulationInputData)
    SimulationInputData <- cbind(Parameters = rownames(SimulationInputData),
SimulationInputData)
    rownames(SimulationInputData) <- 1:nrow(SimulationInputData)
    writeData(wb, Tab, SimulationInputData)
#
# Build data frame to contain both input simulation parameters as well as output
#
    dfMean <- as.data.frame(Mean)
    colnames(dfMean)  <- c("PopMean")
    dfSMean <- as.data.frame(sMean)
    colnames(dfSMean)  <- c("SamMean")
    dfStandardDeviation <- as.data.frame(StandardDeviation)
    colnames(dfStandardDeviation)  <- c("PopSD")
    dfSStandardDeviation <- as.data.frame(sStandardDeviation)
    colnames(dfSStandardDeviation)  <- c("SamSD")
```

```
    dfCorrelation <- as.data.frame(Correlation)
    colnames(dfCorrelation)  <- c("PopCor1", "PopCor2", "PopCor3")
    dfSCorrelation <- as.data.frame(sCorrelation)
    colnames(dfSCorrelation)  <- c("SamCor1", "SamCor2", "SamCor3")
    dfMean <- dfMean*100
    dfSMean <- dfSMean*100
    dfStandardDeviation <- dfStandardDeviation*100
    dfSStandardDeviation <- dfSStandardDeviation*100
    SimParameters <- cbind(dfMean, dfSMean, dfStandardDeviation, dfSStandardDeviation,
      dfCorrelation, dfSCorrelation)
    SimParameters <- round(SimParameters, 2)
    rownames(SimParameters)  <- c("Stock", "Rate", "Volatililty")
    SimParameters <- cbind(Parameters = rownames(SimParameters), SimParameters)
    Tab <- "Simulation Statistics"
    addWorksheet(wb, sheetName = Tab, gridLines = TRUE)
    writeData(wb, Tab, SimParameters)
    WorkBookName <- paste0('VaR and Valuations ', Variable, '.xlsx')
    saveWorkbook(wb, WorkBookName, overwrite = TRUE)
  }
  return(Valuations)
}
```