Homework Assignment 1
FE 621: Computational Methods in Finance
Instructor: Ionut Florescu
2/20/2019

Rukmal Weerawarana
rweerawa@stevens.edu | 104-307-27
Department of Financial Engineering
Stevens Institute of Technology

Overview

In this Homework Assignment, we explore various numerical optimization methods through the lens of the Black-Scholes-Merton Option pricing model (Shreve 2004). Using this, we calculate explore the implied volatility of options for various assets traded on the market. Furthermore, we also explore numeric methods of differential calculation to compute the Greeks of these candidate options. Finally, we explore numeric integration and the behavior of various quadrature methods.

Unless otherwise stated, the following shorthand notation is used to distinguish between dates:

- **DATA1** - Wednesday, February 6 2019 (*2/6/19*)
- **DATA2** - Thursday, February 7 2019 (*2/7/19*)

The content of this Homework Assignment is divided into three sections; the first discusses data gathering, formatting, and a discussion of the assets being examined. The second contains data analysis, and an exploration of implied volatility through the Black-Scholes-Merton pricing framework and related computations. Finally, the third section discusses numerical integration and the convergence of various quadrature rules.

See Appendix A for specific question implementations, and (Weerawarana 2019) for full source code.

1 Data Overview

1.1 Asset Descriptions

1.1.1 *SPY* - SPDR S&P 500 ETF (State Street Global Advisors 2019)

The S&P 500 (i.e. *Standard & Poor's 500*) is a stock market index tracking the 500 largest companies on the American Stock Exchange by Market Capitalization. In this case, the market capitalization is defined as the number of outstanding shares, multiplied by the current share price. A stock market index is designed to be a metric that can be used by market observers as a benchmark to gauge the relative health of the stock market, by analyzing the aggregate performance of its largest components.

However, this index is not the same as the *SPY* ETF. An ETF (*Exchange Traded Fund*) is a basket of stocks that is designed to track a specific index or benchmark. That is, it provides investors with exposure to a index or benchmark, without having to own all of the underlying assets that constitute a composite asset. In addition to more liquidity, this type of investment also provides lower transaction costs and required minimum investment to gain exposure to a given index or benchmark. It is traded on an exchange, akin to a typical traded asset.

1.1.2 *VIX* - CBOE Volatility Index (CBOE (Chicago Board Options Exchange) 2019)

The CBOE (*Chicago Board Options Exchange*) volatility index, *VIX* is an exchange traded product (*ETP*) designed to give investors exposure to the market's expectation of 30-day volatility. It is priced using a large set of implied volatility of put and call options on the S&P 500 index to gauge investor sentiment. Typically, the price of the VIX has an inverse relationship to the price of the S&P 500 index. Similar to an ETF, an ETP is also traded on an exchange as a typical traded asset.

References

CBOE (Chicago Board Options Exchange). 2019. *VIX: Volatility Index.* Accessed February 20, 2019. `http://www.cboe.com/vix`.

Shreve, Steven E. 2004. *Stochastic Calculus for Finance II.* 153–164. April. Pittsburgh, PA: Springer Finance. ISBN: 0-387-40101-6.

State Street Global Advisors. 2019. *SPY: SPDR S&P 500 ETF Trust.* Accessed February 20, 2019. `https://us.spdrs.com/en/etf/spdr-sp-500-etf-SPY`.

Weerawarana, Rukmal. 2019. *FE 621 Homework - rukmal - GitHub.* Accessed February 20, 2019. `https://github.com/rukmal/FE-621-Homework`.

A Appendix

A.1 Question 1 Implementation

```
library("Rblpapi")

# Connect to Bloomberg Terminal backend service
blpConnect(host = "localhost", port = 8194)


#---------------------------------
# Data Download Functionality
#---------------------------------


getPrice <- function(security, startTime, endTime, timeZone) {
  # Downloads and returns the closing price of a given security
  # for each minute in the trading day.
  #
  # Args:
  #   security: Name of the security to be downloaded.
  #   startTime: Datetime object with the start time.
  #   endTime: Datetime object with the end time.
  #   timeZone: Time zone of the target start and end times.
  #
  # Returns:
  #   DataFrame with the closing price for each minute in the
  #   trading day.

  # Getting price data
  data <- getBars(security = security, barInterval = 1,
                  startTime = startTime, endTime = endTime,
                  tz = timeZone)

  # Isolate time and closing price
  data <- data[c("times", "close")]

  # Rename columns
  colnames(data) <- c("Dates", "Close")

  # Return
  data
}


createOptionName <- function(security, dates, prices, type, suffix) {
  # Creates the Bloomberg-standard option name, given a security, date, price,
  # option type and suffix.
  #
  # Args:
  #   security: Name of the security to be included in the option price.
  #   dates: Dates to be included in option name.
  #   prices: Prices to be included in the option name.
  #   type: Type of the option ("C" or "P").
  #   suffix: Suffix for option name (typically "Index" or "Equity").
  #
  # Returns:
  #   Vector of Bloomberg-compatible option names.

  # Empty vector to store names
```

```
  names <- c()

  # Iterate over each date and price
  for (date in dates) {
    for (price in prices) {
      # Building option name
      name <- paste(security, date, paste(type, price, sep = ""), suffix)

      # Appending to list of option names
      names <- c(names, name)
    }
  }

  # Returning names
  names
}


#---------------------------------
# DATA1
#---------------------------------


# Define Start and End times (DATA1)
data1Start <- ISOdatetime(year = 2019, month = 2, day = 6,
                          hour = 9, min = 30, sec = 0)
data1End <- ISOdatetime(year = 2019, month = 2, day = 6,
                        hour = 16, min = 0, sec = 0)

# Defining time zone
timeZone = "America/New_York"

# Defining top-level securities
securities <- c("SPY US Equity", "AMZN US Equity", "VIX Index")

# Getting prices for each of the top-level securities
for (security in securities) {
  data <- getPrice(security, data1Start, data1End, timeZone)
  write.csv(data, file = paste(security, "DATA1", "csv", sep = "."),
            row.names = FALSE)
}

# Expiration dates
expDates <- c("2/15/19", "3/15/19", "4/18/19")

# Defining put and call prices for SPY and AMZN options
# Grabbing prices for 5% +/- current price

# Current SPY price
spyCurrent <- 270
spyPrices <- c(floor(0.95 * spyCurrent):ceiling(1.05 * spyCurrent))

# Current AMZN price (need to do this manually because of option strikes)
# Closest option price to 95% of price is at $1557.50 and 105% is $1722.50
amznCurrent <- 1640
amznPrices <- seq(1555, 1725, by=5)

# Creating option names for SPY and AMZN
spyOptions <- createOptionName("SPY", expDates, spyPrices, "C", "Equity")
spyOptions <- c(spyOptions, createOptionName("SPY", expDates, spyPrices,
                                             "P", "Equity"))
```

```r

amznOptions <- createOptionName("AMZN", expDates, amznPrices, "C", "Equity")
amznOptions <- c(amznOptions, createOptionName("AMZN", expDates, amznPrices,
                                               "P", "Equity"))

# Getting prices for each of the options
for (option in c(amznOptions, spyOptions)) {
  data <- getPrice(option, data1Start, data1End, timeZone)
  # Only print to file if option exists
  if (all(dim(data) > 0)) {
    optionFileName <- gsub("/", "-", option) # Need to do this for Windows
    write.csv(data, file = paste(optionFileName, "csv", sep = "."),
              row.names = FALSE)
  }
}


#-----------------------------------
# DATA2
#-----------------------------------

# Define Start and End times (DATA2)
data2Start <- ISOdatetime(year = 2019, month = 2, day = 7,
                          hour = 9, min = 30, sec = 0)
data2End <- ISOdatetime(year = 2019, month = 2, day = 7,
                        hour = 16, min = 0, sec = 0)

# Getting prices for each of the top-level securities
for (security in securities) {
  data <- getPrice(security, data2Start, data2End, timeZone)
  write.csv(data, file = paste(security, "DATA2", "csv", sep = "."),
            row.names = FALSE)
}
```

question_solutions/question_1.R