

## 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.*

## References

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

```

1 library("Rblpapi")
2
3 # Connect to Bloomberg Terminal backend service
4 blpConnect(host = "localhost", port = 8194)
5
6
7 #-----
8 # Data Download Functionality
9 #-----
10
11
12 getPrice <- function(security, startTime, endTime, timeZone) {
13   # Downloads and returns the closing price of a given security
14   # for each minute in the trading day.
15   #
16   # Args:
17   #   security: Name of the security to be downloaded.
18   #   startTime: Datetime object with the start time.
19   #   endTime: Datetime object with the end time.
20   #   timeZone: Time zone of the target start and end times.
21   #
22   # Returns:
23   #   DataFrame with the closing price for each minute in the
24   #   trading day.
25
26   # Getting price data
27   data <- getBars(security = security, barInterval = 1,
28                   startTime = startTime, endTime = endTime,
29                   tz = timeZone)
30
31   # Isolate time and closing price
32   data <- data[c("times", "close")]
33
34   # Rename columns
35   colnames(data) <- c("Dates", "Close")
36
37   # Return
38   data
39 }
40
41
42 createOptionName <- function(security, dates, prices, type, suffix) {
43   # Creates the Bloomberg-standard option name, given a security, date, price,
44   # option type and suffix.
45   #
46   # Args:
47   #   security: Name of the security to be included in the option price.
48   #   dates: Dates to be included in option name.
49   #   prices: Prices to be included in the option name.
50   #   type: Type of the option ("C" or "P").
51   #   suffix: Suffix for option name (typically "Index" or "Equity").
52   #
53   # Returns:
54   #   Vector of Bloomberg-compatible option names.
55
56   # Empty vector to store names

```

```

57  names <- c()
58
59  # Iterate over each date and price
60  for (date in dates) {
61    for (price in prices) {
62      # Building option name
63      name <- paste(security, date, paste(type, price, sep = ""), suffix)
64
65      # Appending to list of option names
66      names <- c(names, name)
67    }
68  }
69
70  # Returning names
71  names
72 }
73
74
75 #-----
76 # DATA1
77 #-----
78
79
80 # Define Start and End times (DATA1)
81 data1Start <- ISOdatetime(year = 2019, month = 2, day = 6,
82                           hour = 9, min = 30, sec = 0)
83 data1End <- ISOdatetime(year = 2019, month = 2, day = 6,
84                         hour = 16, min = 0, sec = 0)
85
86 # Defining time zone
87 timeZone = "America/New_York"
88
89 # Defining top-level securities
90 securities <- c("SPY US Equity", "AMZN US Equity", "VIX Index")
91
92 # Getting prices for each of the top-level securities
93 for (security in securities) {
94   data <- getPrice(security, data1Start, data1End, timeZone)
95   write.csv(data, file = paste(security, "DATA1", "csv", sep = "."),
96            row.names = FALSE)
97 }
98
99 # Expiration dates
100 expDates <- c("2/15/19", "3/15/19", "4/18/19")
101
102 # Defining put and call prices for SPY and AMZN options
103 # Grabbing prices for 5% +/- current price
104
105 # Current SPY price
106 spyCurrent <- 270
107 spyPrices <- c(floor(0.95 * spyCurrent):ceiling(1.05 * spyCurrent))
108
109 # Current AMZN price (need to do this manually because of option strikes)
110 # Closest option price to 95% of price is at $1557.50 and 105% is $1722.50
111 amznCurrent <- 1640
112 amznPrices <- seq(1555, 1725, by=5)
113
114 # Creating option names for SPY and AMZN
115 spyOptions <- createOptionName("SPY", expDates, spyPrices, "C", "Equity")
116 spyOptions <- c(spyOptions, createOptionName("SPY", expDates, spyPrices,
117                                             "P", "Equity"))

```

```
118
119 amznOptions <- createOptionName("AMZN", expDates, amznPrices, "C", "Equity")
120 amznOptions <- c(amznOptions, createOptionName("AMZN", expDates, amznPrices,
121                                               "P", "Equity"))
122
123 # Getting prices for each of the options
124 for (option in c(amznOptions, spyOptions)) {
125   data <- getPrice(option, data1Start, data1End, timeZone)
126   # Only print to file if option exists
127   if (all(dim(data) > 0)) {
128     optionFileName <- gsub("/", "-", option) # Need to do this for Windows
129     write.csv(data, file = paste(optionFileName, "csv", sep = "."),
130              row.names = FALSE)
131   }
132 }
133
134
135 # -----
136 # DATA2
137 # -----
138
139 # Define Start and End times (DATA2)
140 data2Start <- ISOdatetime(year = 2019, month = 2, day = 7,
141                           hour = 9, min = 30, sec = 0)
142 data2End <- ISOdatetime(year = 2019, month = 2, day = 7,
143                          hour = 16, min = 0, sec = 0)
144
145 # Getting prices for each of the top-level securities
146 for (security in securities) {
147   data <- getPrice(security, data2Start, data2End, timeZone)
148   write.csv(data, file = paste(security, "DATA2", "csv", sep = "."),
149            row.names = FALSE)
150 }
```

question\_solutions/question\_1.R