Lecture 4: Data Downloading Package

Cheng Lu

Overview

- quantmod package
- Historical Prices
- Quoted Prices
- Option prices
- Write csv File
- Homework

quantmod package

- Use **install.packages()** to install packages, and **library()** to load packages
- Install the package for the first time, load the package *every time* you use it

Example

```
> rm(list = ls()) # remove all variables
> #install.packages("quantmod")
> library(quantmod)
```

- Use package ”quantmod” for functions to download and analyze data
- Use **getSymbols()** to download historical price

Example

```
> getSymbols(Symbols = "MSFT")
[1] "MSFT"
> MSFT <- data.frame(MSFT)
```

quantmod package

Use **head()** to show the first few observations

Example

```
> head(MSFT)# show first few observations
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2007-01-03     29.91     30.25    29.40      29.86    76935100      22.24590
2007-01-04     29.70     29.97    29.44      29.81    45774500      22.20865
2007-01-05     29.63     29.75    29.45      29.64    44607200      22.08199
2007-01-08     29.65     30.10    29.53      29.93    50220200      22.29805
2007-01-09     30.00     30.18    29.73      29.96    44636600      22.32040
2007-01-10     29.80     29.89    29.43      29.66    55017400      22.09690
```

Here adjusted close price is the close price adjusted from stock splitting, dividend paying etc.

Historical Prices

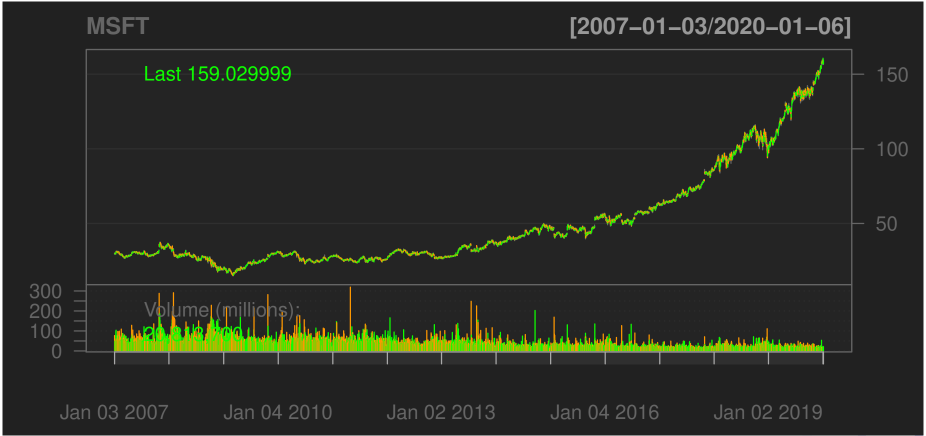
Example

```
getSymbols(Symbols = "INTC", from = "2001-01-01", to = "2001-01-31")
INTC <- data.frame(INTC)
head(INTC) # show first few observations
tail(INTC) # show last few observations

getSymbols(c("SPY","^VIX")) # download multiple symbols

aapl <- getSymbols("AAPL", auto.assign = F)
# By default, from = "2007-01-01", to = Sys.time(), auto.assign = T
# The default source is "yahoo", which is the only source now
```

Historical Prices

Example

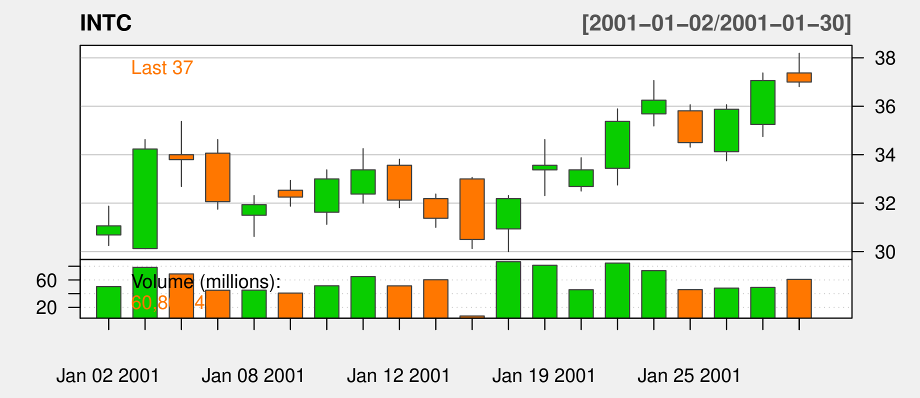
```
> chartSeries(MSFT) # generate chart series
```



Historical Prices

Example

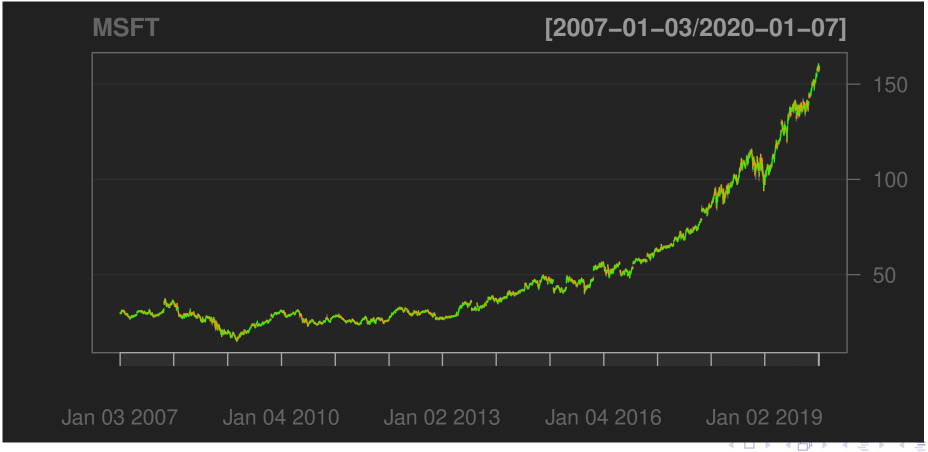
```
> chartSeries(INTC,theme=chartTheme('white'))
```



Historical Prices

Example

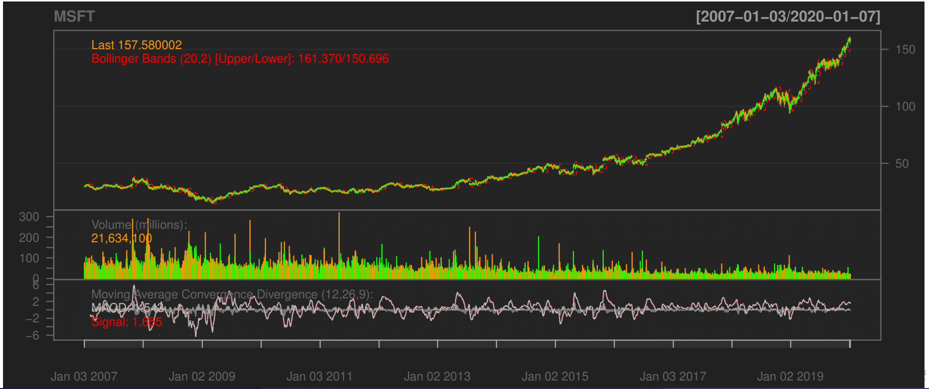
```
> chartSeries(MSFT,TA=NULL) #no volume
```



Historical Prices

Example

```
> chartSeries(MSFT,TA=c(addVo(),addBBands()))
> #add volume and Bollinger Bands from TTR
> addMACD()
```



Quoted Prices

we can use **getQuote()** function to download current quote price (the last transaction price), **getSymbols()** only download historical daily price.

Example

```
> getQuote("MSFT")
              Trade Time   Last    Change   % Change   Open   High    Low
MSFT 2021-09-24 16:00:02 299.35 -0.2099915 -0.07009997 298.23 299.79 296.934
> (LastQuoteMSFT <- getQuote("MSFT")$Last) # Last Quote
[1] 299.35
> nrow(MSFT) # nunber of rows
[1] 3709
> MSFT[nrow(MSFT),] # last observation is the price of the last trading day
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2021-09-24    298.23     299.8   296.93     299.35    14994200        299.35
```

Here "Change" is the difference of the last price and the close price of the last trading day.

Option prices

Use **getOptionChain()** to download option price[1].

Example

```
> AAPL.option <- getOptionChain("AAPL")
> mode(AAPL.option)
[1] "list"
> names(AAPL.option)# use names() to check variables of lists
[1] "calls" "puts"
> is.data.frame(AAPL.option)
[1] FALSE
> is.data.frame(AAPL.option$calls)
[1] TRUE
```

[1] Option is a contract that a investor has right to buy or sell the underlying asset at specific time (expiration date or maturity) with specific price (strike price).

Option prices

Example

```
> head(AAPL.option$calls)
                    Strike  Last        Chg   Bid Ask Vol  OI       LastTradeTime       IV  ITM
AAPL211001C00075000     75 71.89  2.8899994 71.85  72   2   9 2021-09-24 14:54:26 1.687502 TRUE
AAPL211001C00080000     80 66.91  1.0100021 66.90  67  13  49 2021-09-24 14:52:41 1.781251 TRUE
AAPL211001C00085000     85 62.00  1.6500015 61.90  62   1   9 2021-09-24 14:58:30 1.617189 TRUE
AAPL211001C00090000     90 56.95  0.7500000 56.90  57   9 264 2021-09-24 14:55:03 1.460940 TRUE
AAPL211001C00100000    100 45.98 -0.9000015 46.90  47   9 243 2021-09-24 10:07:25 1.171879 TRUE
AAPL211001C00105000    105 42.15  0.9000015 41.90  42   2  47 2021-09-24 15:50:09 1.031255 TRUE
```

For the first observation

- It is a call option that a investor has right buy "AAPL" on "2021-10-01" with price $75.
- The most recent traded (last quoted) price for the contract is $71.89
- The highest price a buyer willing to pay (bid price) for the option is $71.85
- The lowest price a seller willing to sell (ask price) for the option is $72
- The price will change during trading day when investors trade the option

Option prices

- If no expiration date specified, options with *closest* expiration date will be returned
- Set expiration dates with **Exp =**

Example: Setting different expiration dates

```
> # set expiration date
> AAPL.option1 <- getOptionChain("AAPL", Exp = "2021-10-15")
> names(AAPL.option1)
[1] "calls" "puts"
> AAPL.options <- getOptionChain("AAPL", c("2021-10-01","2021-10-15"))
> names(AAPL.options) # expired between "2021-10-01" and "2021-10-08"
[1] "Oct.01.2021" "Oct.08.2021" "Oct.15.2021"
> names(AAPL.options$Oct.08.2021)
[1] "calls" "puts"
```

The options above may be expired (on ”2021-10-01” etc), you may need to find available expiration dates from Yahoo Finance website for replicating the examples.

Option prices

Example Continued

```
> AAPL.options2 <- getOptionChain("AAPL", "2021/2022")
> names(AAPL.options2) # all options expired in 2021 and 2022
 [1] "Oct.01.2021" "Oct.08.2021" "Oct.15.2021" "Oct.22.2021"
 [5] "Oct.29.2021" "Nov.19.2021" "Dec.17.2021" "Jan.21.2022"
 [9] "Mar.18.2022" "Apr.14.2022" "Jun.17.2022" "Sep.16.2022"
> AAPL.options.all <- getOptionChain("AAPL",NULL)
> names(AAPL.options.all) # all options available
 [1] "Oct.01.2021" "Oct.08.2021" "Oct.15.2021" "Oct.22.2021" "Oct.29.2021"
 [6] "Nov.19.2021" "Dec.17.2021" "Jan.21.2022" "Mar.18.2022" "Apr.14.2022"
[11] "Jun.17.2022" "Sep.16.2022" "Jan.20.2023" "Mar.17.2023" "Jun.16.2023"
[16] "Sep.15.2023" "Jan.19.2024"
> AAPL.options.all$Oct.15.2021$calls$Strike # strike of call options expired on "2021-10-15"
 [1]  35  40  45  50  55  60  65  70  75  80  85  90  95 100 105 110 115 120
[19] 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210
[37] 215 220 225 230 235
```

Option prices

We usually use the average of bid and ask as the price for options. So we can create a new column called "Price", and delete all the columns other than "Strike" and "Price".

Example: Averaging Bid and Ask for one expiration date

```
> names(AAPL.options$Oct.08.2021$calls)
 [1] "Strike"        "Last"          "Chg"           "Bid"
 [5] "Ask"           "Vol"           "OI"            "LastTradeTime"
 [9] "IV"            "ITM"
> AAPL.options$Oct.08.2021$calls$Price <- 0.5*(AAPL.options$Oct.08.2021$calls$Bid
                                             + AAPL.options$Oct.08.2021$calls$Ask)
> names(AAPL.options$Oct.08.2021$calls)
 [1] "Strike"        "Last"          "Chg"           "Bid"
 [5] "Ask"           "Vol"           "OI"            "LastTradeTime"
 [9] "IV"            "ITM"           "Price"
> # Too many variables, we only need "Strike" and "Price"
> AAPL.options$Oct.08.2021$calls <- AAPL.options$Oct.08.2021$calls[c("Strike", "Price")]
> names(AAPL.options$Oct.08.2021$calls)
[1] "Strike" "Price"
```

Similarly we can do that for put options, and other expiration dates.

Option prices

How about if we want to save a list which only contains ”Strike” and ”Price” for calls and puts for all expiration dates?

Example: Averaging Bid and Ask for all expiration dates

```
# AAPL.options.all[[1]]$calls$Price <- 0.5*(AAPL.options.all[[1]]$calls$Bid
#                                           + AAPL.options.all[[1]]$calls$Ask)
# AAPL.options.all[[1]]$calls <- AAPL.options.all[[1]]$calls[c("Strike","Price")]
# AAPL.options.all[[1]]$puts$Price <- 0.5*(AAPL.options.all[[1]]$puts$Bid
#                                          + AAPL.options.all[[1]]$puts$Ask)
# AAPL.options.all[[1]]$puts <- AAPL.options.all[[1]]$puts[c("Strike","Price")]
# ...
# similar for 2, 3, 4,...,length(AAPL.options.all)
#
for(i in 1:length(AAPL.options.all)){
  AAPL.options.all[[i]]$calls$Price <- 0.5*(AAPL.options.all[[i]]$calls$Bid
                                           + AAPL.options.all[[i]]$calls$Ask)
  AAPL.options.all[[i]]$calls <- AAPL.options.all[[i]]$calls[c("Strike","Price")]
  AAPL.options.all[[i]]$puts$Price <- 0.5*(AAPL.options.all[[i]]$puts$Bid
                                          + AAPL.options.all[[i]]$puts$Ask)
  AAPL.options.all[[i]]$puts <- AAPL.options.all[[i]]$puts[c("Strike","Price")]
}
```

Option prices

We can also use vectorized operations.

Example

```
> func <- function(x){
+   x$calls$Price <- 0.5*(x$calls$Bid + x$calls$Ask)
+   x$calls <- x$calls[c("Strike", "Price")]
+   x$puts$Price <- 0.5*(x$puts$Bid + x$puts$Ask)
+   x$puts <- x$puts[c("Strike", "Price")]
+   return(x)
+ }
> AAPL.options.all <- getOptionChain("AAPL", NULL)
> AAPL.options.all <- lapply(AAPL.options.all, func)
> names(AAPL.options.all$Oct.08.2021$calls)
[1] "Strike" "Price"
```

Write csv File

- Option prices we obtained from Yahoo finance is the current price
- Use **write.csv()** to save them in csv files for consecutive dates for empirical analysis
- Remember to set/get working directory each time you read/write files

Example

```
setwd("C:/Users/demonew/Documents/Stevens/Graduate Courses/FE515/21f")
write.csv(AAPL.options.all$Oct.08.2021$calls,
          file = "data2021_09_26Exp2021_10_08calls.csv")
```

Here ”2021_09_26” indicates the date of downloading the data, and ”2021_10_08” indicates the expiration date. You can use different format of the file name in practice.

Write csv File

How about if we want to save all the option prices into .csv files? First we need to convert the file names into the format we want.

Example

```
> Sys.Date()# today
[1] "2021-09-26"
> (today <- format(Sys.Date(), "%Y_%m_%d"))
[1] "2021_09_26"
> (Exp <- names(AAPL.options.all))# all expiration dates
 [1] "Oct.01.2021" "Oct.08.2021" "Oct.15.2021" "Oct.22.2021" "Oct.29.2021"
 [6] "Nov.19.2021" "Dec.17.2021" "Jan.21.2022" "Mar.18.2022" "Apr.14.2022"
[11] "Jun.17.2022" "Sep.16.2022" "Jan.20.2023" "Mar.17.2023" "Jun.16.2023"
[16] "Sep.15.2023" "Jan.19.2024"
```

Write csv File

Example

```
> Exp <- as.Date(Exp, format = "%b.%d.%Y") # convert to date object
(Exp <- format(Exp, "%Y_%m_%d")) # convert to chars with certain format
 [1] "2021_10_01" "2021_10_08" "2021_10_15" "2021_10_22" "2021_10_29"
 [6] "2021_11_19" "2021_12_17" "2022_01_21" "2022_03_18" "2022_04_14"
[11] "2022_06_17" "2022_09_16" "2023_01_20" "2023_03_17" "2023_06_16"
[16] "2023_09_15" "2024_01_19"
> for (i in 1:length(Exp)) {
+   write.csv(AAPL.options.all[[i]]$calls,
+             file = paste0("data",today,"Exp",Exp[i],"calls.csv"))
+   write.csv(AAPL.options.all[[i]]$puts,
+             file = paste0("data",today,"Exp",Exp[i],"puts.csv"))
+ }
```

Exercise: Replace the above for loop by vectorized operations. (Hint: use **mapply()**)