FE8828 Programming Web Applications in Finance

Dr. Yang Ye yy@runchee.com

Sep 24, 2019

1. Bank

```
Data frame 1: Account
| AcountNo | Name |

Data frame 2: Transaction
| TransactionNo | Date | AccountNo | TransactionType | Amount | Currency |

Data frame 3: Currency to SGD
| Currency | Conversion | Date |

TransactionType can be one of Withdraw/Deposit/Spend
```

1. Create 10 client accounts with initial random deposit (random in [1000, 2000]) and credit (2000 for all) in SGD and store them in Rda file (Rda: "R data file", use save() and load()).
2. Create 3 currencies: CNY, USD, SGD. Download their conversion rate during 2019-07-01 and 2019-09-30 from MAS website and store in Rda.
3. Generate random transaction data for 10 accounts during 2019-07-01 and 2019-09-30 and store them in Rda.

- Make it more realistic, deposit is 1-2 times per month, a random number between 1000-2000, in any of three currencies.
- Spend/Withdraw can be any times [0, 1000] and any amount (could be a random number from a long-tail distribution), in any currencies
- Spend is increasing from credit
- Withdraw is deducting from balance
- Deposit is positive, Withdraw/Spend is negative.
- Interest payment from bank for end-of-month balance.
- Auto-generation Constraints: You can't withdraw more than the deposit, can't spend more than credit.

4. Bank earns 2% from FX and 1% from SGD from client spending, gives client interest 0.5% (all annualized rates).

Build a Shiny interface for

1. Trigger auto-generation
2. Client view: Month-end statement generation with following input

- Select client account
- Select month

```
{ Client Name }
{ Month }

# Chart of balance: showing credit, deposit, balance along the axis of date

# Transaction History
```

```
Date | TransactionType | Currency | Amount | Amount (in SGD) | Deposit Balance | Credit Balance |
     |                 |          |        |                 |                 |                |
     |                 |          |        |                 |                 |                |
     |                 |          |        |                 |                 |                |
-------------------------------------------------------------------------------------------------
                                             Month-End Balance | Deposit         | Credit         |

# Summary

TransactionType | Amount |
...             |        |
```

3. Bank view, Month report with following input

- Select month

```
# Chart: showing aggregated total deposit and credit along the axis of date

# PnL table: Detail table (for every day)

Date | Total Deposit | Total Credit | PnL from Client Spending |
.... |               |              |                          |

# Risk table as of the end of month (sort descending by largest Credit-Deposit)

Client Name | Deposit | Credit |
...         |         |        |
```

---

2. Option trading - dynamic hedging

Financial derivatives like Options are named so because their value is derived from its underlyings. Greeks are the sensitivity of the price of the derivaties to its underlying parameters. For example, a vanilla European option depends on S (Underlying price), stri$K$e, Vol(atility), r (interest rate), q (cost of carry), T (time).

Greeks:

- $\frac{dV}{dS}$: Delta
- $\frac{d^2V}{dS^2}$: Gamma
- $\frac{dV}{dVol}$: Vega
- $\frac{dV}{dT}$: Theta

To simplify, we set $r = q = 0$ (so Rho is zero).

The daily PnL of option can be approximated into, according to Talyor expansion $\Delta V = Delta * \Delta S + 0.5 * Gamma * \Delta S^2 + Vega * \Delta Vol + Theta * (dT) + SmallRemainder$

Option trading with dynamtic hedging uses two positions: one position in option and one position in underlying. The position on option is established at the beginning of the trade. The position in underlying is adjusted periodically (e.g. daily) to the opposite of the option delta so the strategy-delte is zero. The objective is to realize the difference between realized volatility and implied volatility.

For example, there is a 30-day ATM call option, K = S = 100, r = q = 0, vol = 0.3 constant till option expiry.

```r
library(tidyverse)
library(fOptions) # Note there are conflicts in name for functions like "filter()"
```

```
days <- 20.0
vega0 <- GBSGreeks("Vega", "c", S = 100, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)
delta0 <- GBSGreeks("Delta", "c", S = 100, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)
gamma0 <- GBSGreeks("Gamma", "c", S = 100, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)
theta0 <- GBSGreeks("Theta", "c", S = 100, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)
price0 <- GBSOption("c", S = 100, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)@price
```

If on day 1, S1 = 98, Volatility is unchanged.

```
S1 <- 98
option_pnl <- GBSOption("c", S = S1, X = 100,
                        Time = (days - 1) / 250, r = 0, b = 0, sigma = 0.3)@price - price0
approx_pnl <- delta0 * (S1 - 100) + 0.5 * gamma0 * (S1 - 100) ** 2 + vega0 * 0 + theta0 * 1/250
cat(paste0("option_pnl: ", option_pnl, ", ", approx_pnl, "\n"))
```

```
## option_pnl: -1.02175324971442, -1.02444665039741
```

```
cat(paste0("diff with approx.: ", option_pnl - approx_pnl, "\n"))
```

```
## diff with approx.: 0.00269340068298818
```

Generate random price series with given S and *volatility*.

```
    S <- 100
    K <- 100
    sigma <- 0.3 # realized vol can be 0.3 for [2] or 0.5 for [3]
    drift <- 0 # drift = r - q
    timestep <- 1 / 250
    days <- 20 / 250
    N <- days / timestep

    p1 <- (drift - 0.5 * sigma * sigma) * timestep
    p2 <- sigma * sqrt(timestep)

    # ss is the simulated price movement for N days
    ss <- rep(S, N) * c(1, cumprod(rlnorm(N - 1, mean = p1, sd = p2)))
```

Questions:

[1] Please show how good Taylor's approximation to actual PnL.

We can also have the following conclusions from option hedging.

[2] If realized volatility and implied volatility are the same, Gamma and Theta can neutralize each other, i.e. $\Sigma(0.5 * Gamma * \Delta S^2 * dt + Theta * dt) \approx 0$

[3] If realized volatility > implied volatility, "Overall PnL" (profit and loss) for option hedging is $\approx \Sigma(\frac{1}{2} * Gamma * (RealizedVol^2 - ImpliedVol^2) * dt * S)^2$.

Your task is to use simulation to prove above three points with the above-mentioned 30-day ATM call option and some random-generated price series.

Note 1: that simulation would show some acceptable level of difference. Note 2: The overall PnL = the cost of buying the option + final payoff + delta PnL.

```
df <- tibble(S = ss, days = 1:days)
opt <- rowwise(df) %>% mutate(
  price = GBSOption("c", S = S, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3)@price,
  delta = GBSGreeks("Delta", "c", S = S, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3),
  gamma = GBSGreeks("Gamma", "c", S = S, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3),
```

```
  theta = GBSGreeks("Theta", "c", S = S, X = 100, Time = days / 250, r = 0, b = 0, sigma = 0.3) / 250
)
...
```

---

3. Option Delta Hedging (Real-world)

Show case how delta hedging works as a trading strategy.

- Show how one trade works
- Backtest for available history (12-months, each month as) - one-year history from the website.

Assumption:

- You hold 100 ATM call/put option which expires in 30 days (calendar days). You just need to do either Call or Put. After finishing the tasks below, duplicate the R Notebook and carry out 2nd-round analysis using an option of ~25% delta, take a guess for the strike level.
- You start to do delta hedging daily immediately till 2nd last day. You close stock position in the last day. Option expires on the last day as well (either ITM with positive PnL or ATM/OTM with zero PnL)
- Delta hedging: calculate the delta from option, negate it, that's the quantity what you need to hold over 1 day. Repeat for every trading day.
- Daily PnL: (option premium change) + (stock holding quantity * price change).
- You can get your favorite stocks here. There is one year of data.
  - Many website started charging data download. Luckily we can have CBOE pushing VIX Index for top five active trading stocks.
  - http://www.cboe.com/vix
  - "The VIX Index is a calculation designed to produce a measure of constant, 30-day expected volatility of the U.S. stock market, derived from real-time, mid-quote prices of S&P 500® Index (SPXSM) call and put options. On a global basis, it is one of the most recognized measures of volatility – widely reported by financial media and closely followed by a variety of market participants as a daily market indicator."
  - Data is available to download from here. Pick one of five stocks. https://fred.stlouisfed.org/search/?st=%20CBOE%20Equity%20VIX
  - Combine with Yahoo's stock price series (Use Close)
- Daily IV 30 is provided, IV 30 is implied volatility for 30 days. We flat-extrapolate it to be for < 30 days (which means don't change it).
- As underlying is equity, dividend yield is applicable for B-S valuation
- US risk-free rate for 1M: 0.8% (annualized)

Analysis: For one trade

- Daily PnL v.s. Time to expiry: split into option and stock.
- Final PnL: accumulative of Daily PnL. split into option and stock
- Max Drawdown: accumulative of Daily PnL, max - min.
- Sharpe ratio: Sharpe ratio = (Mean of Daily PnL - Risk-free rate)/Standard deviation of Daily PnL

Analysis: For backtest

- Distribution of Final PnL

- Distribution of Max Drawdown
- Final PnL v.s. Option Expiry Date
- ...
- More analysis
- in R Markdown

Example flow of analysi:

- Create xts object from the data from website.
- One trade analysis
  - Pick a date range using xts object.
  - Get starting date and end date.

```
dates <- index(xts_obj)
start_date <- min(dates)
end_date <- max(dates)
start_price <- xts_obj[start_date, "Close"]
start_volatility <- xts_obj[start_date, "IV30"]
```

  - Create a df with date column

```
df <- tibble(date = dates)
df$Close <- coredata(xts_obj[, "Close"])
```

  - Daily Profit and Loss ("DoD PnL")
    * Option side:

```
X <- start_price
sigma = start_volatility
r <- 0.8 / 100
# Vary S and Time everyday
S <- Close
Time <- (end_date - date) / 365
GBSOption(TypeFlag, S, X, Time, r, b, sigma)@price

df_opt <- rowwise(df) %>%
  mutate(premium = GBSOption(TypeFlag = "...",
                             S = Close,
                             X = start_price,
                             Time = (end_date - date) / 365,
                             r = ..., # interest rate
                             b = ..., # dividend yield
                             sigma = start_volatility)@price) %>%
  ungroup %>%
  mutate(Option_DoD_PnL = ifelse(date == start_date,
  # On the 1st date, we count the cost of buying the option
                                 premium * (-1),
                                 premium - lag(premium)))
```

    * Hedging side:

```
df %>%
rowwise() %>%
mutate(delta_hedge = GBSGreeks("delta", TypeFlag, S, X, Time, r, b, sigma) *
                     quantity * (-1)) %>%
ungroup() %>%
```

```
mutate(Hedging_DoD_Pnl = ifelse(date == start_date,
                                0,
                                delta_hedge * (Close - lag(Close))))
```

  * Daily PnL (combined):

```
mutate(DoD_PnL = Option_DoD_PnL + Hedging_DoD_Pnl)
```

- Max Drawdown: accumulative of Daily PnL, max - min.

```
ungroup() %>%
mutate(PnL = cumsum(DoD_PnL)) %>%
{
  xs <- .$PnL
  max(cummax(xs) - cummin(xs))
}
```