Black-Litterman Portfolio Optimization

Arunabha Sarkar

13 August 2019

Introduction: Black-Litterman model

The Black-Litterman model of portfolio optimization was published by Fisher Black and Robert Litterman in 1992 (https://jfi.pm-research.com/content/1/2/7). This model combines market equilibrium concept with custom future expectation to arrive at optimal portfolio allocation. In this project, we use historical trends of accounting values ‘Return on Investment’ and ‘Profit/Loss Figures’ of banking firms to predict future expectation returns and then optimize portfolio allocation using the ‘Black-Litterman model’.

Preliminary Work

Stock price time series information is easy to collect using existing APIs and packages. Accounting time series of Indian stocks, although available on respectively quarterly reports and annual reports of various company websites, they are not available at command on any central repository. For generating this data, a dynamic webscrapper was built to scrap accounting data (quarterly reports) for 100s of companies from upto 25 years. The technology can be used for a plethora of other applications as well. The code and construct for the same can be found in my Github here (https://github.com/NV2017/Mo_Co_Q_Report_Full_Time_Series_Scrap). The scrapped data (primary data, partly unstructured) is converted to structured data using an additionally built custom tool. This too can be found in my Github here (https://github.com/NV2017/Cleaning_raw_Quarterly_Reports_raw_money_control_scrap_data).

Portfolio modelling of the Black-Litterman model

Once our accounting time series is collected, processed and the stock time series data also arrange, we start our Black Litterman modelling. For the sake of simplicity, we only choose 4 common banks: Kotak Bank, SBI Bank, Axis Bank and HDFC Bank. First step, we clean the working directory and load the necessary packages:

```
rm(list=ls())

setwd("~/NISM/3 Portfolio Optimization/BL Project")

libraries_required = c('BLModel','treemap')

for(i in seq(libraries_required))
{
  if(!(libraries_required[i] %in% rownames(installed.packages())))
  {
    try(expr = install.packages('libraries_required[i]'), silent = T)
  }
  try(expr = library(libraries_required[i], character.only = T), silent = T)
}
```

Next, we load the necessary bank company stock market returns and visualize it:

```
Bank_Raw_Data <- read.csv('Bank Data.csv', header = T)

colnames(Bank_Raw_Data) <- c("KotakBankDate","KotakBankClose",
                             "SBIBankDate","SBIBankClose",
                             "AxisBankDate","AxisBankClose",
                             "HDFCBankDate","HDFCBankClose")

head(Bank_Raw_Data)
```

```
##   KotakBankDate KotakBankClose SBIBankDate SBIBankClose AxisBankDate
## 1    16/07/2009        155.613  17/07/2009      167.395   2009-07-17
## 2    17/07/2009        160.425  20/07/2009      172.295   2009-07-20
## 3    20/07/2009        168.375  21/07/2009      171.155   2009-07-21
## 4    21/07/2009        160.637  22/07/2009      169.390   2009-07-22
## 5    22/07/2009        156.038  23/07/2009      172.715   2009-07-23
## 6    23/07/2009        160.262  24/07/2009      169.860   2009-07-24
##   AxisBankClose HDFCBankDate HDFCBankClose
## 1      120.8636   17/07/2009        287.33
## 2      125.8694   20/07/2009        294.75
## 3      125.5924   21/07/2009        292.18
## 4      122.2268   22/07/2009        289.03
## 5      126.9486   23/07/2009        291.81
## 6      126.9486   24/07/2009        289.99
```

Next, we process this raw data into the required time series. We also recombine the individual component time series to a single data frame for the model.

```
Kotak_Bank <- Bank_Raw_Data[,1:2]

SBI_Bank <- Bank_Raw_Data[,3:4]

Axis_Bank <- Bank_Raw_Data[,5:6]

HDFC_Bank <- Bank_Raw_Data[,7:8]


Kotak_Bank[,1] <- as.Date(Kotak_Bank[,1], format="%d/%m/%Y")
colnames(Kotak_Bank)[1] <- "Date"

SBI_Bank[,1] <- as.Date(SBI_Bank[,1], format="%d/%m/%Y")
colnames(SBI_Bank)[1] <- "Date"

Axis_Bank[,1] <- as.Date(Axis_Bank[,1], format="%Y-%m-%d")
colnames(Axis_Bank)[1] <- "Date"

HDFC_Bank[,1] <- as.Date(HDFC_Bank[,1], format="%d/%m/%Y")
colnames(HDFC_Bank)[1] <- "Date"

List_of_Individuals <- list(Kotak_Bank, SBI_Bank, Axis_Bank, HDFC_Bank)

Combined_data <- List_of_Individuals[[1]]

for(i in 2:length(List_of_Individuals))
{
  Combined_data <- merge(Combined_data, List_of_Individuals[[i]],  by= "Date")
}


head(Combined_data)
```

```
##         Date KotakBankClose SBIBankClose AxisBankClose HDFCBankClose
## 1 2009-07-17        160.425      167.395      120.8636        287.33
## 2 2009-07-20        168.375      172.295      125.8694        294.75
## 3 2009-07-21        160.637      171.155      125.5924        292.18
## 4 2009-07-22        156.038      169.390      122.2268        289.03
## 5 2009-07-23        160.262      172.715      126.9486        291.81
## 6 2009-07-24        158.863      169.860      126.9486        289.99
```

Next we find the returns of this time series.

```
Returns_Data <- data.frame(na.omit(cbind( as.character(as.Date(Combined_data[,1])),
                               c(NA,diff(log(Combined_data[,2]))),
                               c(NA,diff(log(Combined_data[,3]))),
                               c(NA,diff(log(Combined_data[,4]))),
                               c(NA,diff(log(Combined_data[,5]))) )),
                           stringsAsFactors = F)

colnames(Returns_Data) <- c("Date","Kotak_Bank_Returns","SBI_Bank_Returns",
                            "Axis_Bank_Returns","HDFC_Bank_Returns")

head(Returns_Data)
```

```
##         Date   Kotak_Bank_Returns     SBI_Bank_Returns
## 1 2009-07-20   0.0483670723936225   0.0288517994255608
## 2 2009-07-21  -0.0470465172768666 -0.00663853950572513
## 3 2009-07-22  -0.0290475885758719  -0.0103658301517022
## 4 2009-07-23   0.0267104034362404   0.0194390713459303
## 5 2009-07-24 -0.00876769267179611  -0.0166682398016196
## 6 2009-07-27   0.0510767120229874  0.00744895656784905
##      Axis_Bank_Returns    HDFC_Bank_Returns
## 1   0.0405823501539677   0.0254962036614081
## 2 -0.00220257791161504 -0.00875751267094671
## 3  -0.0271634644755991  -0.0108395412530697
## 4   0.0379037624041274  0.00957241320222035
## 5                    0 -0.00625649355227775
## 6    0.048473229814749  -0.0114798184351574
```

Now that we have the returns data of the stock, we start with the downloaded accounting values of the same stocks. Keeping within the scope of this project, we only assess the ‘Return on Investment’ and ‘Profit and Loss’ from the acconting statements. To compare and operate on these numbers which are of different scales, we ‘scale’ them as well.

```
Kotak_Bank_Q_Rep <- read.csv('Kotak Mahindra Bank Q_Report_Time_Series_2019_07_13_17_40_58.cs
v', header = T)

SBI_Bank_Q_Rep <- read.csv('State Bank of India Q_Report_Time_Series_2019_07_13_17_41_01.csv'
, header = T)

Axis_Bank_Q_Rep <- read.csv('Axis Bank Q_Report_Time_Series_2019_07_13_17_21_49.csv', header
 = T)

HDFC_Bank_Q_Rep <- read.csv('HDFC Bank Q_Report_Time_Series_2019_07_13_17_21_53.csv', header
 = T)

Kotak_Bank_ROI <- scale(na.omit(Kotak_Bank_Q_Rep$Return.on.Assets..))
SBI_Bank_ROI <- scale(na.omit(SBI_Bank_Q_Rep$Return.on.Assets..))
Axis_Bank_ROI <- scale(na.omit(Axis_Bank_Q_Rep$Return.on.Assets..))
HDFC_Bank_ROI <- scale(na.omit(HDFC_Bank_Q_Rep$Return.on.Assets..))

Kotak_Bank_PL <- scale(na.omit(Kotak_Bank_Q_Rep$P.L.After.Tax.from.Ordinary.Activities))
SBI_Bank_PL <- scale(na.omit(SBI_Bank_Q_Rep$P.L.After.Tax.from.Ordinary.Activities))
Axis_Bank_PL <- scale(na.omit(Axis_Bank_Q_Rep$P.L.After.Tax.from.Ordinary.Activities))
HDFC_Bank_PL <- scale(na.omit(HDFC_Bank_Q_Rep$P.L.After.Tax.from.Ordinary.Activities))
```

Within the scope of this project, we define the outlook for each of the four stocks as average of excess of scaled version of ‘Return on Investment’ and ‘Profit and Loss’ over it’s last year average. Though not idea utilisation of the extensive downloaded resources, this still serves as working demonstration of the Black Litterman model implementation in R.

```
Kotak_Bank_Outlook <- (Kotak_Bank_ROI[1] - mean(Kotak_Bank_ROI[1:4]) + Kotak_Bank_PL[1] - mea
n(Kotak_Bank_PL[1:4]))/2

SBI_Bank_Outlook <- (SBI_Bank_ROI[1] - mean(SBI_Bank_ROI[1:4]) + SBI_Bank_PL[1] - mean(SBI_Ba
nk_PL[1:4]))/2

Axis_Bank_Outlook <- (Axis_Bank_ROI[1] - mean(Axis_Bank_ROI[1:4]) + Axis_Bank_PL[1] - mean(Ax
is_Bank_PL[1:4]))/2

HDFC_Bank_Outlook <- (HDFC_Bank_ROI[1] - mean(HDFC_Bank_ROI[1:4]) + HDFC_Bank_PL[1] - mean(HD
FC_Bank_PL[1:4]))/2
```

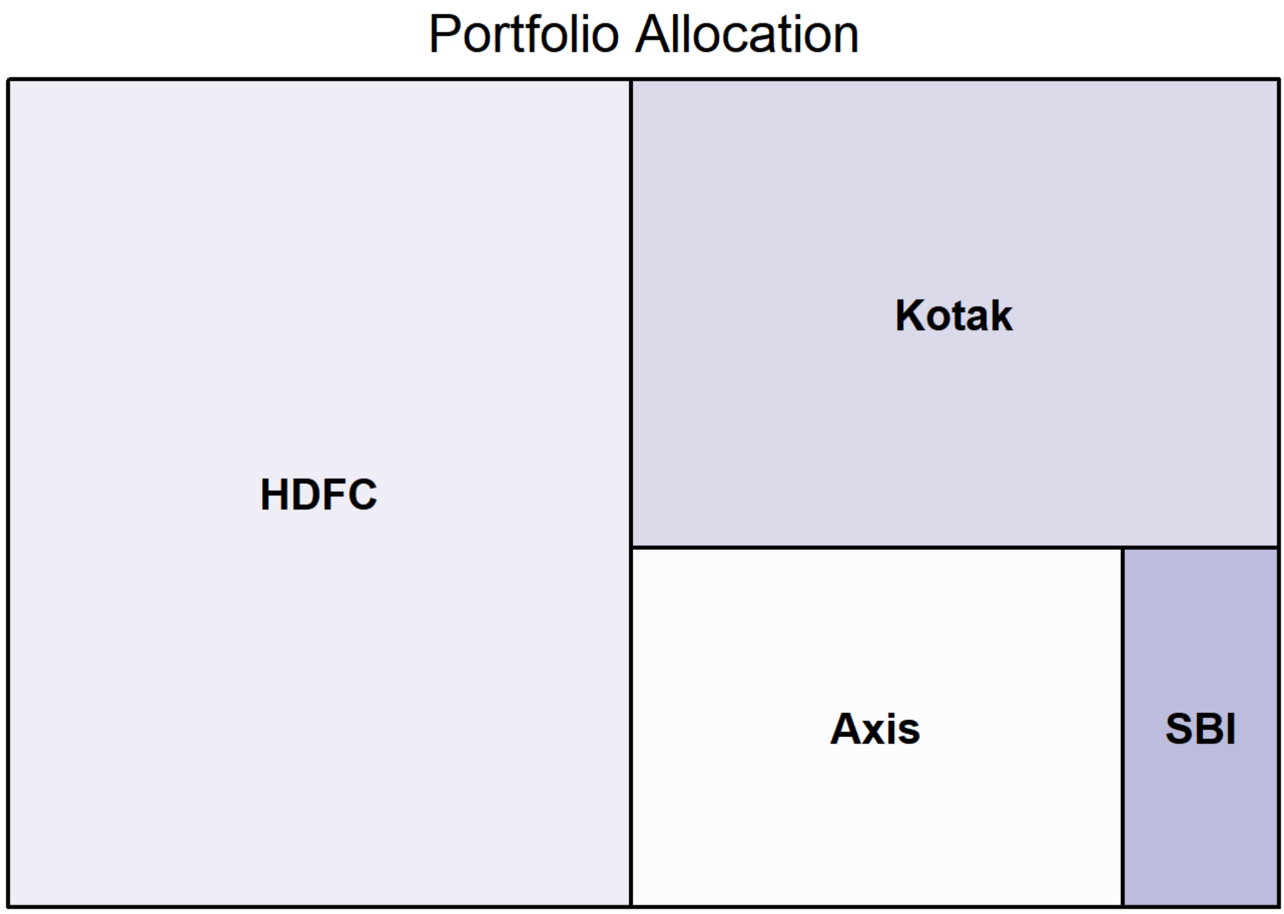
Now that the range of raw data is organised, we implement the Black Litterman model using the **BLModel** package. We choose ‘Mean Absolute Deviation’ as measure of our risk in the model.

```
dat <- cbind(Combined_data, matrix(1/dim(Combined_data)[1], dim(Combined_data)[1], 1) )

returns_freq <- 250 # Daily

SR <- 0.5

Pe <- diag(4)

qe <- c(Kotak_Bank_Outlook, SBI_Bank_Outlook, Axis_Bank_Outlook, HDFC_Bank_Outlook)
qe <- (qe+1)^(1/62)-1 # Converting to daily

tau <- 0.02

BL_Model <- BL_post_distr(dat = dat[,-1], returns_freq = returns_freq, prior_type = NULL,
              market_portfolio = rep(1/4,4), SR = SR, P = Pe, q = qe, tau = tau,
              risk = "MAD", alpha = 0, views_distr = observ_normal, "diag", cov_matrix = NULL
)
```

Conclusion: Black-Litterman postfolio optimization visualized as portfolio allocation treemap

From the onset, our outlook was most positive for HDFC and least for SBI. From visualization of this Black Litterman portfolio optimization visualization, we find the same outlook is furnished by the model as well, albeit, after considering and quantitative comparison with the daily returns time series as well.

```
df_tree <- data.frame(Asset_Names = c('Kotak','SBI','Axis','HDFC'),
   Asset_Value = unlist(abs(head(BL_Model$post_distr,1)[,1:4])),stringsAsFactors = F)

treemap(dtf = df_tree, index = 'Asset_Names', vSize = 'Asset_Value',
        fontsize.title = 20, fontsize.labels = c(16, 16),
        palette = 'Purples', title = 'Portfolio Allocation')
```



In the end, we conclude with a successful implementation of Bloack Litterman model while utilising accounting time series data in a limited fashion. The data collected and the technique/code utilised can be modified for custom needs. We finish off by clearing the workspace.

```
rm(list=ls())
```