Bond Performance Analysis using Flattener Strategy

Executive Summary

The purpose of this report is to evaluate the performance of the 10 year and 2 year U.S. Treasury yield curve spread trade by using the flattener strategy.
Using the flattener strategy we shorted the front leg and longed the back leg) using our initial capital (of \$1 million). We have set up a DV01-neutral yield curve spread trade and each week we rebalanced it to maintain the DV01-neutral position and examined the cash position, total capital and weekly returns.

Introduction

A yield curve spread is the yield differential between two different maturities of a bond issuer. The later maturity leg of the trade is referred to as the back leg and the trade leg maturing earlier is called the front leg. The two primary yield curve spread strategies are the 'flattener' strategy (in this strategy we short the front leg and long the back leg) and the 'steepener' strategy (in this strategy we long the front leg and short the back leg). The risk measure for yield curve spread trades is DV01 (dollar value of a basis point). As the back leg DV01 is greater than the front leg DV01, we calculate a hedge ratio to result in a DV01 neutral position. In this report we have considered the yield differential between the 10 year U.S. Treasury bond and the 2 year U.S. Treasury bond and we have evaluated the performance of the treasury yield curve spread trade by using the flattener strategy. We used the Nelson-Siegel-Svensson's yield curve model to calculate the yield rates. These yield rates were used to compute the bond prices. For simplicity, we have only used zero coupon bonds and assumed that bonds of any maturity are available. We have also considered the capital margin requirements (which is 10% of the trading positions) while deploying our strategy. We then set up a DV01-neutral yield curve spread trade and each week we rebalanced it to maintain the DV01-neutral position and then analyzed the cash position, total capital and returns.

Initial Setup

We start by importing data and changing the Date column into the date format as recognized by R; followed by creating two functions namely, `nss` and `calculate_bond_price`. Here `nss` is used to calculate the interest rates using the Nelson-Siegel-Svensson model using the inputs β_0, β_1, β_2, β_3, τ_1, and τ_2. These inputs are found from a file called `gsw_yields.csv`, from the Board of Governors of the Federal Reserve System, that contains historic data collected by the U.S. treasury. The second function called `calculate_bond_price` uses inputs, `maturity`(time to maturity), `par`(par-value of the bond), and `rate`(interest rate), to calculate the price of the bond with those properties. We then initiate a number of variables that will be used in further code to facilitate further analysis.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data <- read.csv('~/Documents/Documents/Investments/gsw_yields.csv',
                 header = TRUE)
data$Date <- as.Date(data$Date, format = "%m/%d/%Y")

nss <- function(t, b0, b1, b2, b3, t1, t2) {
    r <- b0 + b1*((1-exp(-t/t1))/(t/t1)) + (b2*(((1-exp(-t/t1))/(t/t1))-(exp(-t/t1)))) +
      (b3*(((1-exp(-t/t2))/(t/t2))-(exp(-t/t2))))
    return(r)
}

calculate_bond_price <- function(maturity, par, rate) {
  bond_price <- par/((1+(rate/100))^maturity)
  return(bond_price)
}
Date1 <- as.Date("2020-11-11")
Convexity <- 0;
Delta_Bond_Price <- 0;
yield <- 0;
temp2 <- 0;
r2_new <- 0
r2_old <- 0
r10_new <- 0
r10_old <- 0
position_2_new <- 0
position_10_new <- 0
position_2_old <- 0
position_10_old <- 0
cash_position_old <- 0
capital <- 0
weekly_return <- 0
cash_position_new <- 0;
return <- 0;
cummulative_return <- 0;
Spread_Return <- 0
Convexity_10 <- 0
Convexity_2 <-0
Delta_Bond_Price_2 <- 0
Delta_Bond_Price_10 <- 0
Convexity_Return <- 0
Time_Return <- 0
Residual <- 0
C_Residual <- 0;
C_Convex <- 0;
C_Time <- 0;
C_Spread <- 0;
```

First part of the computations

Initially, a `for` loop is created; this is done to first filter out the coefficient values on a weekly basis starting 1983-12-30. The `r_2_new` and `r_10_new` values are interest rates calculated using the `nss` function for zero coupon 2 year and 10 year bonds respectively assuming the bonds are issued on that day. The `r_2_old` and `r_10_old` values are interest rates calculated using the `nss` function for zero coupon 2 year and 10 year bonds respectively assuming the bonds are issued on that day one week earlier, thus reducing the time to maturity by a week.

```
for (i in 0:1904) {
  temp <- data %>% filter(Date == (as.Date("1983-12-30")+(i*7)))
  Date1[i+1] <- as.Date("1983-12-30")+(i*7)
  r2_old[i+1] <- nss((2-(1/52)), temp$BETA0, temp$BETA1, temp$BETA2, temp$BETA3,
                     temp$TAU1, temp$TAU2)
  r2_new[i+1] <- nss(2, temp$BETA0, temp$BETA1, temp$BETA2, temp$BETA3, temp$TAU1,
                     temp$TAU2)
  r10_old[i+1] <- nss((10-(1/52)), temp$BETA0, temp$BETA1, temp$BETA2, temp$BETA3,
                      temp$TAU1, temp$TAU2)
  r10_new[i+1] <- nss(10, temp$BETA0, temp$BETA1, temp$BETA2, temp$BETA3, temp$TAU1,
                      temp$TAU2)
}
```

Second part of the computations

A `final_new` dataframe is then created and further data manipulation is done using the `mutate()` function from the `dplyr` library. This is done to find the following
1. `price_2_new` and `price_10_new`: These bond prices are calculated using the `calculate_bond_price()` function created above and using the `r_2_new` and `r_10_new` interest rates.
2. `mod_duration_2_new` and `mod_duration_10_new`: These modified durations are calculated using the respective durations of 2 and 10 years and using the `r_2_new` and `r_10_new` interest rates.
3. `dv01_2_new` and `dv01_10_new`: These DV01s are calculated using the `mod_duration_2_new` and `mod_duration_10_new` and the `price_2_new` and `price_10_new`.
4. `x_new` is calculated by dividing `dv01_2_new` by `dv01_10_new`.
5. `price_2_old` and `price_10_old`: These bond prices are calculated using the `calculate_bond_price()` function created above and using the `r_2_old` and `r_10_old` interest rates.
The NA values are removed for ease of computation using the `complete.cases()` function.

```
final_new <- data.frame("Date" = Date1[1:1905], r2_old, r2_new, r10_old, r10_new) %>%
  mutate(price_2_new = calculate_bond_price(2, 1, r2_new), price_10_new =
           calculate_bond_price(10, 1, r10_new)) %>%
  mutate(mod_duration_2_new = (2)/(1+(r2_new/100)), mod_duration_10_new =
           (10)/(1+(r10_new/100))) %>%
  mutate(dv01_2_new = mod_duration_2_new*price_2_new, dv01_10_new =
           mod_duration_10_new*price_10_new) %>%
  mutate(x_new = dv01_2_new/dv01_10_new) %>%
  mutate(price_2_old = calculate_bond_price(2, 1, r2_old), price_10_old =
           calculate_bond_price(10, 1, r10_old))

final_new <- final_new[complete.cases(final_new), ]
```

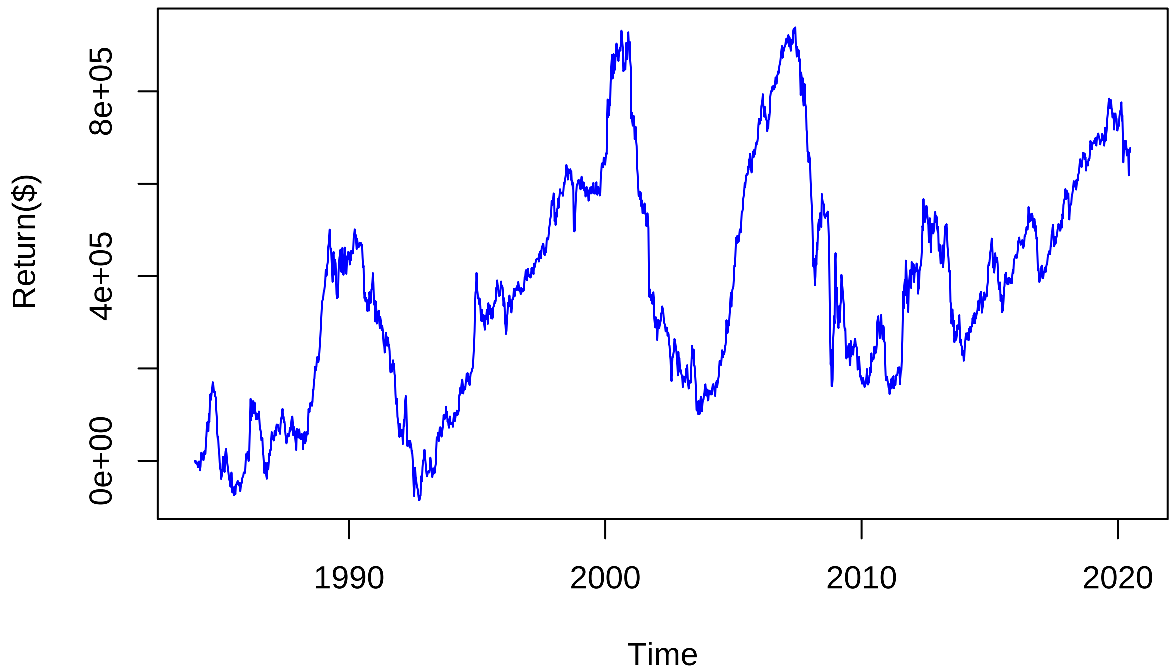
Third part of the computations

A `for` loop is created to calculate a number of other elements of the computation. Firstly, the `position_2_old` and `position_10_old` (current values of our positions) are computed using the current price of the bonds that were issued one week earlier, multiplied by the number of bonds that were bought last week when the bonds were issued. Secondly, `cash_position_old` is the cash position once our trades from last week are closed, including the capital help during this time and any interest earned or paid on this position. The `capital` held at any point is the same as the `cash_position_old`. The `n_2_new` and `n_10_new` are then computed to find the number of bonds held using the `x_new` ratio computed above for the current period as well as the `capital` we now have to make the new deal and keeping in mind the `10%` capital requirement. Finally the `return` and `cummulative_return` were calculated using the change in `capital`.

```r
for (i in 1:length(final_new$Date)) {
  if(i==1){
    cap <- 1000000;
    n_2_new <- 0;
    n_10_new <- 0;
  }else{
    cap <- capital[i-1];
  }
  if(i==1){
    position_2_old[i] <- 0;
    position_10_old[i] <- 0;
    cash_position_old[i] <- 1000000;
    capital[i] <- cash_position_old[i];
  }else{
    position_2_old[i] <- n_2_new[i-1]*final_new$price_2_old[i];
    position_10_old[i] <- n_10_new[i-1]*final_new$price_10_old[i];
    temp <- data %>% filter(Date == final_new$Date[i]);
    ## Cash position after sale of position from last week
    cash_position_old[i] <- ((cap - position_2_old[i] + position_10_old[i] +
                                 position_2_new[i-1] - position_10_new[i-1]) *
                                (1 + (nss((1/52), temp$BETA0, temp$BETA1,
                                          temp$BETA2, temp$BETA3, temp$TAU1,
                                          temp$TAU2)/(100*52))));
    capital[i] <- cash_position_old[i];
  }
  n_2_new[i] <- capital[i]*10 / (final_new$price_2_new[i] + final_new$x_new[i]*
                                   final_new$price_10_new[i]);
  n_10_new[i] <- final_new$x_new[i]*n_2_new[i];
  position_2_new[i] <- n_2_new[i]*final_new$price_2_new[i];
  position_10_new[i] <- n_10_new[i]*final_new$price_10_new[i];
  cash_position_new[i] <- position_2_new[i] - position_10_new[i];
  if(i==1){
    weekly_return[i] <- 0;
    return[i] <- 0;
  }else{
    ## weekly_return is calculated as fractional value
    weekly_return[i] <- (capital[i]-capital[i-1])/capital[i-1];
    ## return and cumulative return are dollar values
    return[i] <- capital[i]-capital[i-1];
    cummulative_return[i] <- sum(return[1:i])
  }
}
```

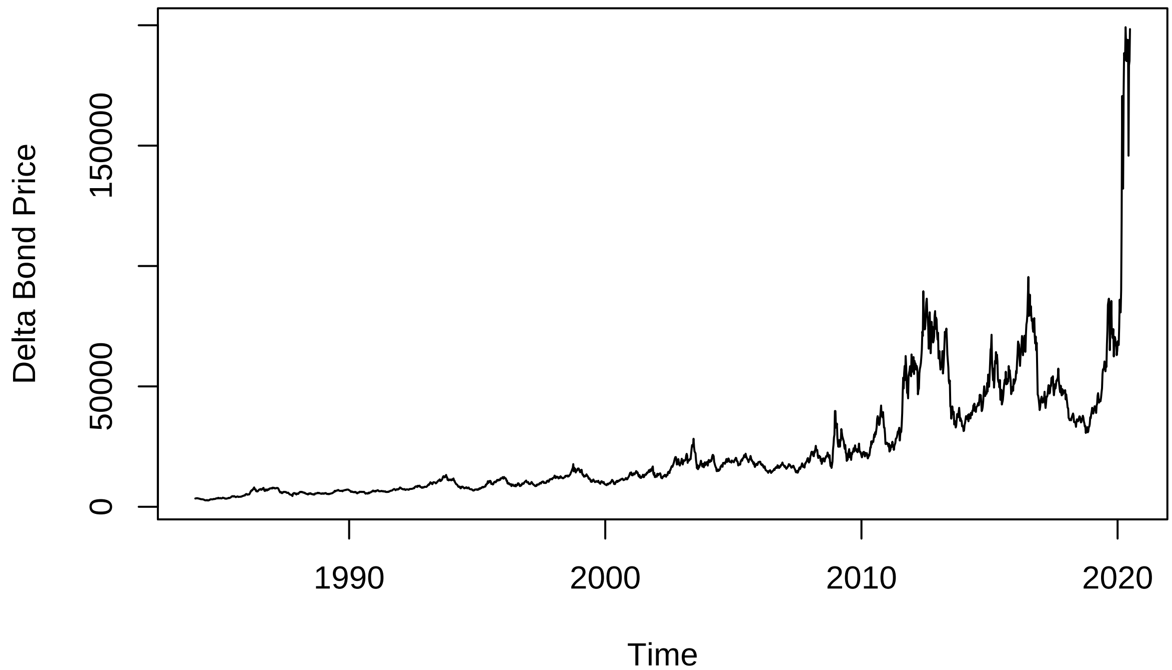
Question 1: Plot the cumulative return for your trading strategy.

```r
plot(final_new$Date, cummulative_return, type='l', xlab = "Time", ylab = "Cummulative
     Return($)", col = 'blue')
```



Question 2: Although the spread trade is DV01-neutral, there is unhedged convexity. Calculate the convexity risk of the spread trade for a 10 basis point change in yields for a constant $1mm (in terms of face value) position in the 10 yr Treasury. Plot the convexity risk over time.

```r
for (i in 0:1904){
  temp2 <- data %>% filter(Date == (as.Date("1983-12-30")+(i*7)))
  yield[i+1] <- nss(10, temp2$BETA0, temp2$BETA1, temp2$BETA2, temp2$BETA3,
                    temp2$TAU1, temp2$TAU2)
  Convexity[i+1] = 10*11/(1 + yield[i+1])^2
  Delta_Bond_Price[i+1] = 0.5 * 1000000 * Convexity[i+1] * (0.1)^2
}
plot(final_new$Date, Delta_Bond_Price[complete.cases(Delta_Bond_Price)], type='l',
     xlab = "Time", ylab = "Delta Bond Price")
```



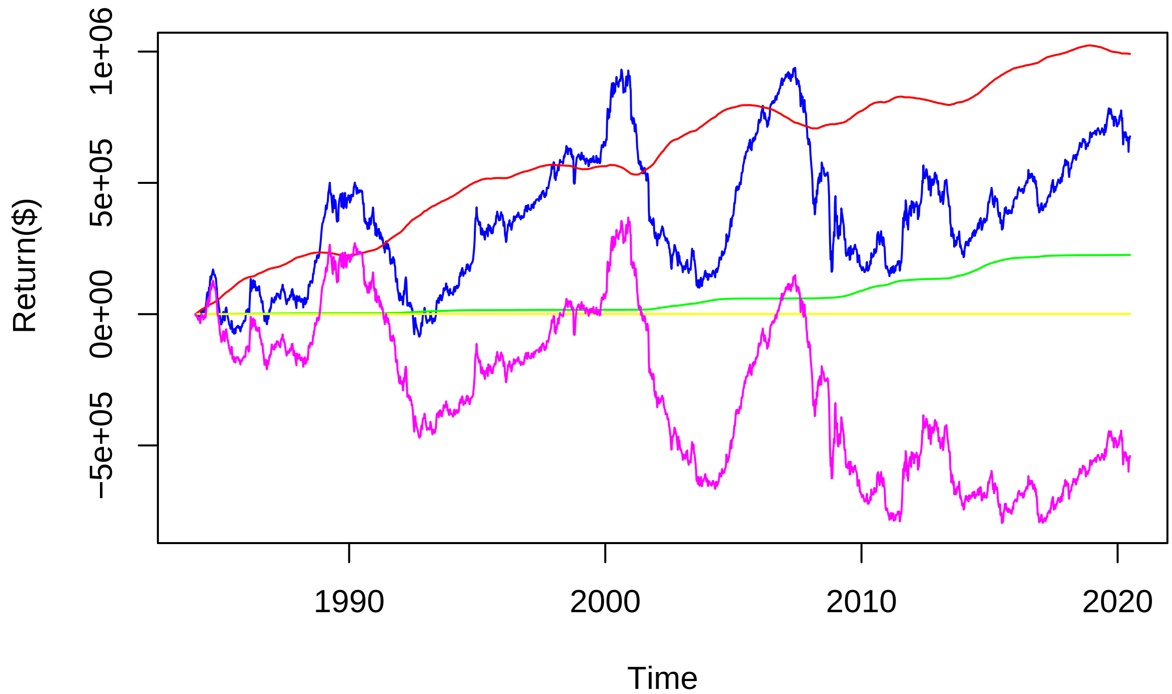
##
Question 3: Each week, calculate the duration and convexity for each leg of your trading strategy. Given your risk metrics, the changes in yields, and the size of your positions, decompose the weekly return into the following components:

```
for (i in 1:length(final_new$Date)) {
  if(i==1){

  }else{
    temp <- data %>% filter(Date == final_new$Date[i]);
    ##Computation of Spread Return
    Spread_Return[i] <- (-position_2_new[i] * final_new$mod_duration_2_new[i] *
                           (final_new$r2_old[i] - final_new$r2_new[i])/100) +
      (position_10_new[i] * final_new$mod_duration_10_new[i] *
         (final_new$r10_old[i] - final_new$r10_new[i])/100)
    C_Spread[i] <- sum(Spread_Return[1:i])
    ## Computation of Convexity Return
    Convexity_10[i] <- 10*11/(1 + final_new$r10_new[i])^2
    Convexity_2[i] <-  2*3/(1 + final_new$r2_new[i])^2
    Delta_Bond_Price_2[i] <-  0.5 * position_2_new[i] * Convexity_2[i] *
      (final_new$r2_old[i] - final_new$r2_new[i])^2
    Delta_Bond_Price_10[i] <-  0.5 * position_10_new[i] * Convexity_2[i] *
      (final_new$r2_old[i] - final_new$r2_new[i])^2
    Convexity_Return[i] <-  Delta_Bond_Price_2[i] - Delta_Bond_Price_10[i]
    C_Convex[i] <- sum(Convexity_Return[1:i])
    ## Computation of Time Return
    Time_Return[i] <- Spread_Return[i] * (nss((1/52), temp$BETA0, temp$BETA1, temp$BETA2,
                                              temp$BETA3, temp$TAU1, temp$TAU2)/(100*52))
    C_Time[i] <- sum(Time_Return[1:i])
    ## Computation of Residual Return
    Residual[i] <- (capital[i] - capital[i-1]) - (Spread_Return[i] +
                                                    Convexity_Return[i] + Time_Return[i])
    C_Residual[i] <- sum(Residual[1:i])
  }
```

```
}
```

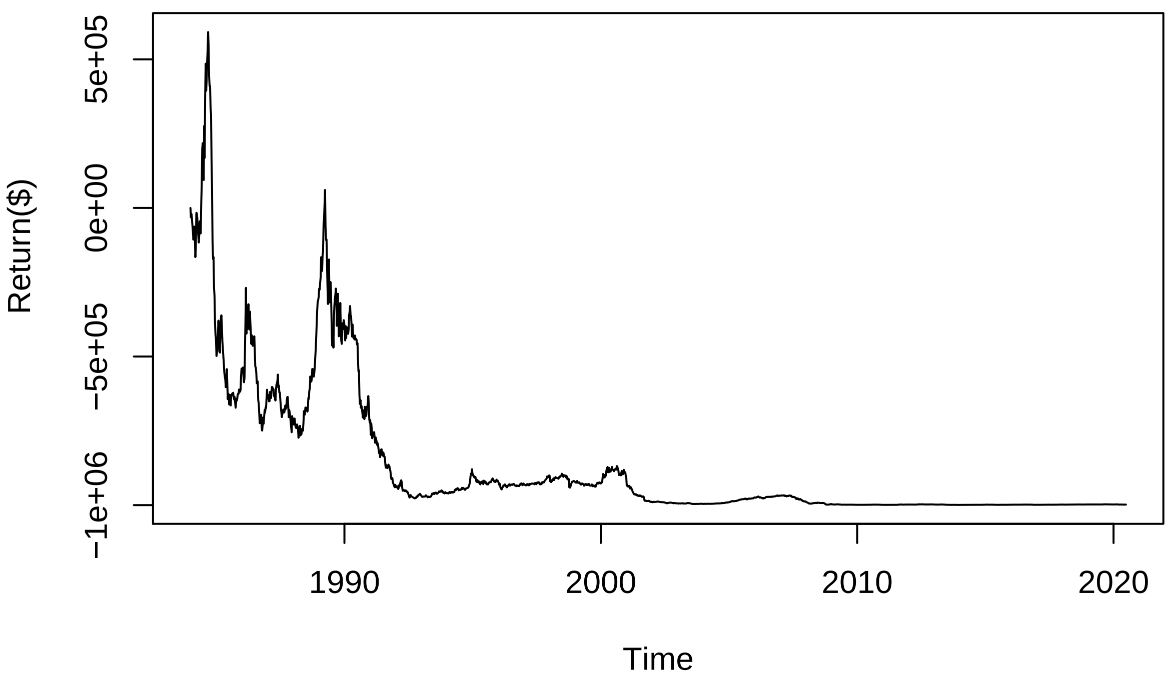
```
plot(final_new$Date, cummulative_return, type='l', xlab = "Time", ylab = "Cummulative
     Return($)", col = 'blue', ylim=c(-8e05, 10e+05))
lines(final_new$Date, C_Spread, col='red')
lines(final_new$Date, C_Convex, col='green')
lines(final_new$Date, C_Time, col='yellow')
lines(final_new$Date, C_Residual, col='magenta')
```



Question 4: How does a 2% margin requirement impact the cumulative return of your trading strategy? Plot the cumulative total return of the 2% margin requirement compared to the 10% margin requirement.

```
for (i in 1:length(final_new$Date)) {
  if(i==1){
    cap <- 1000000;
    n_2_new <- 0;
    n_10_new <- 0;
  }else{
    cap <- capital[i-1];
  }
  if(i==1){
    position_2_old[i] <- 0;
    position_10_old[i] <- 0;
    cash_position_old[i] <- 1000000;
    capital[i] <- cash_position_old[i];
  }else{
    position_2_old[i] <- n_2_new[i-1]*final_new$price_2_old[i];
    position_10_old[i] <- n_10_new[i-1]*final_new$price_10_old[i];
    temp <- data %>% filter(Date == final_new$Date[i]);
    ## Cash position after sale of position from last week
```

```r
    cash_position_old[i] <- ((cap - position_2_old[i] + position_10_old[i] +
                              position_2_new[i-1] - position_10_new[i-1]) *
                             (1 + (nss((1/52), temp$BETA0, temp$BETA1,
                                       temp$BETA2, temp$BETA3, temp$TAU1,
                                       temp$TAU2)/(100*52))));
    capital[i] <- cash_position_old[i];
  }
  n_2_new[i] <- capital[i]*50 / (final_new$price_2_new[i] + final_new$x_new[i]*
                                   final_new$price_10_new[i]);
  n_10_new[i] <- final_new$x_new[i]*n_2_new[i];
  position_2_new[i] <- n_2_new[i]*final_new$price_2_new[i];
  position_10_new[i] <- n_10_new[i]*final_new$price_10_new[i];
  cash_position_new[i] <- position_2_new[i] - position_10_new[i];
  if(i==1){
    weekly_return[i] <- 0;
    return[i] <- 0;
  }else{
    ## weekly_return is calculated as fractional value
    weekly_return[i] <- (capital[i]-capital[i-1])/capital[i-1];
    ## return and cumulative return are dollar values
    return[i] <- capital[i]-capital[i-1];
    cummulative_return[i] <- sum(return[1:i])
  }
}
plot(final_new$Date, cummulative_return, type='l', xlab = "Time", ylab = "Cummulative
     Return($)")
```



With a 2% margin requirement, the cumulative return is observed to finally tend towards zero. This result is quite different compared to 10% margin requirement where we had a positive cumulative return. This difference is observed as smaller margins are more sensitive and can incur huge losses when the interest rates are highly volatile and this can reduce the initial capital drastically. As this effect can get reflected on the value of positions we hold, we may see that the cumulative return gradually tend to zero, as the huge losses incurred is very tough to get recovered from smaller cash positions.

Conclusion

The flattener strategy, also known as the ‘Carry’ strategy can mostly only be used by banks and other large firms due to the huge positions and exposures involved.

Some results that we observed from our analysis are:

1. We can observe that with 10% margin requirement, the 2 yr - 10 yr U.S Treasury yield curve spread trade gave positive returns.

2. We see a major difference between the cumulative returns over the time frame when we compare the return with 10% margin with that of 2% margin.

3. We also observe that the returns are highly volatile whenever the interest rates are volatile, majorly volatility contributed from the 10-yr bond.

4. We can observe that the spread return is a good contributor of the total return from our results.

5. We can see that the long term trend based on observed values from 1983 to 2020, yield is decreasing with time.