Optimized P_MA list and graphs.pdf

Tianyu Zhang

2025-03-29

Test for NVDA data

```r
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

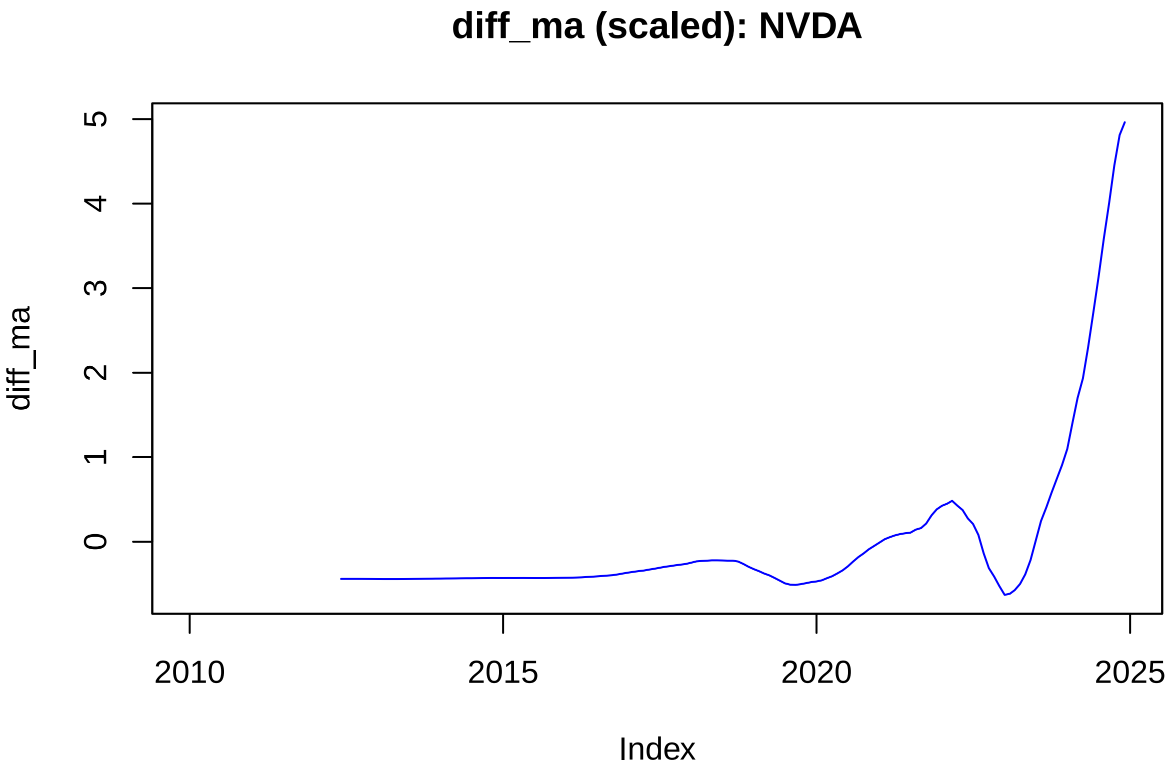
```r
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: TTR
```

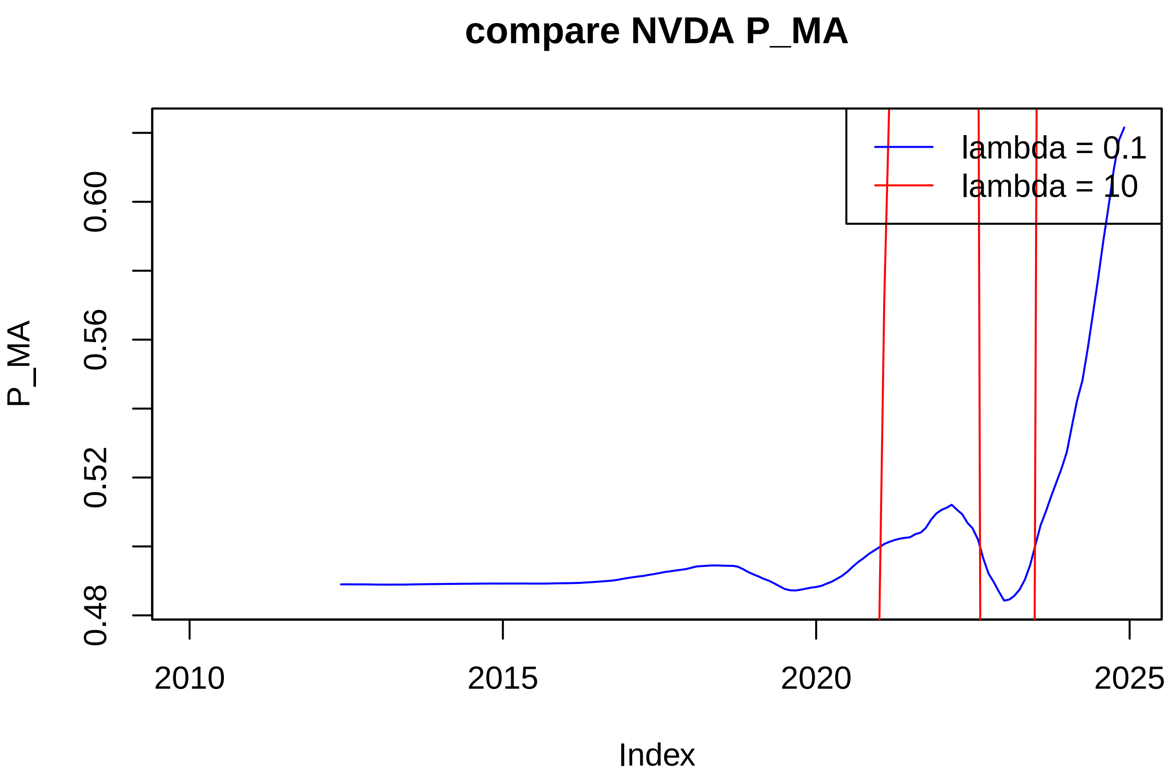
```r
library(TTR)

symbol <- "NVDA"
stock_data <- get.hist.quote(instrument = symbol,
                             start = "2010-01-01",
                             end = "2024-12-31",
                             quote = "AdjClose",
                             compression = "m")
```

```
## time series ends   2024-12-01
```

```r
stock_data <- zoo(stock_data, order.by = as.Date(time(stock_data)))
colnames(stock_data) <- "AdjClose"

# MA
ma_short <- rollmean(stock_data$AdjClose, k = 10, fill = NA, align = "right")
ma_long  <- rollmean(stock_data$AdjClose, k = 30, fill = NA, align = "right")
diff_ma  <- scale(ma_short - ma_long)

# DIFF_MA:
plot(diff_ma, type = "l", col = "blue", main = "diff_ma (scaled): NVDA")
```




```r
sigma <- function(x) {1 / (1 + exp(-x))}

lambda1 <- 0.1
lambda2 <- 10

p_ma_01 <- sigma(lambda1 * diff_ma)
p_ma_10 <- sigma(lambda2 * diff_ma)

plot(p_ma_01, type = "l", col = "blue", main = "compare NVDA P_MA", ylab = "P_MA")
lines(p_ma_10, col = "red")
legend("topright", legend = c("lambda = 0.1", "lambda = 10"), col = c("blue", "red"), lty = 1)
```



We can see there is a difference between lagre and small lambda, the data is valid, next put a pre-retuning stategy for grid search:

NOTE: We are using X=pre_T_adaptive(same logic as T_adaptive, but it's a easier version only for one model, T_adaptive is more suitable for hypermodel) for each stocks:

If the signal (P_MA) is greater than X → Go Long (expecting price to rise)

If the signal is less than X-0.2 → Go Short (expecting price to fall)

If the signal is between X-0.2 and X → Hold (no trade; uncertainty zone)

Using Grid search oprimazation logic for NVDA:

```
library(tseries)
library(zoo)
library(quantmod)
library(TTR)
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(xts)

symbol <- "NVDA"
stock_data <- get.hist.quote(instrument = symbol,
                             start = "2010-01-01",
```

```r
                            end = "2024-12-31",
                            quote = "AdjClose",
                            compression = "m")
```

```
## time series ends   2024-12-01
```

```r
stock_data <- zoo(stock_data, order.by = as.Date(time(stock_data)))
colnames(stock_data) <- "AdjClose"

ma_short <- rollmean(stock_data$AdjClose, k = 12, fill = NA, align = "right")
ma_long  <- rollmean(stock_data$AdjClose, k = 36, fill = NA, align = "right")
diff_ma  <- scale(ma_short - ma_long)

#sigmoid
sigma <- function(x) {1 / (1 + exp(-x))}

# Range for lambda
lambda_values <- seq(0.1, 10, by = 0.1)
score_list <- data.frame()

for (lambda in lambda_values) {
  p_ma <- sigma(lambda * diff_ma)
  valid_index <- which(!is.na(p_ma))

  df <- data.frame(Date = index(stock_data)[valid_index],
                   AdjClose = coredata(stock_data$AdjClose)[valid_index],
                   P_MA = coredata(p_ma[valid_index]))

  # log return
  returns <- diff(log(df$AdjClose))
  signals <- df$P_MA[-1]

  # Trading logic:
  strategy_returns <- ifelse(signals > 0.46, returns,
                             ifelse(signals < 0.25, -returns, 0))
  strategy_returns <- xts(strategy_returns, order.by = df$Date[-1])

  # This step is for checking the data's sufficiency
  if (length(strategy_returns) < 5 || sd(strategy_returns, na.rm = TRUE) == 0) next

  sharpe <- as.numeric(SharpeRatio.annualized(strategy_returns, scale = 12, geometric = FALSE))
  wins <- sum(strategy_returns > 0, na.rm = TRUE)
  losses <- sum(strategy_returns < 0, na.rm = TRUE)
  win_loss <- ifelse(losses == 0, wins, wins / losses)

  # Keep each sharpe + win_loss score
  score <- sharpe + win_loss
  score_list <- rbind(score_list, data.frame(Lambda = lambda, Sharpe = sharpe, WinLoss = win_loss, Scor
}

# Find the best lambda for NVDA:
best_row <- score_list[which.max(score_list$Score), ]
print(score_list)
```

```
##     Lambda      Sharpe   WinLoss       Score
## 1      0.1  1.21987528 1.9183673  3.13824263
## 2      0.2  1.21987528 1.9183673  3.13824263
## 3      0.3  1.21987528 1.9183673  3.13824263
## 4      0.4  0.65661177 1.6551724  2.31178418
## 5      0.5  0.34782114 1.2857143  1.63353543
## 6      0.6  0.30178441 1.3333333  1.63511775
## 7      0.7  0.48994262 1.6111111  2.10105373
## 8      0.8  0.44591276 1.5294118  1.97532453
## 9      0.9  0.44591276 1.5294118  1.97532453
## 10     1.0  0.42620255 1.4705882  1.89679078
## 11     1.1  0.42620255 1.4705882  1.89679078
## 12     1.2  0.43452515 1.5625000  1.99702515
## 13     1.3  0.43452515 1.5625000  1.99702515
## 14     1.4  0.37533500 1.5000000  1.87533500
## 15     1.5  0.37533500 1.5000000  1.87533500
## 16     1.6  0.37533500 1.5000000  1.87533500
## 17     1.7  0.37684393 1.6000000  1.97684393
## 18     1.8  0.37684393 1.6000000  1.97684393
## 19     1.9  0.37684393 1.6000000  1.97684393
## 20     2.0  0.37684393 1.6000000  1.97684393
## 21     2.1  0.34718074 1.5333333  1.88051407
## 22     2.2  0.23349105 1.2105263  1.44401736
## 23     2.3  0.03614806 0.8857143  0.92186234
## 24     2.4 -0.27389456 0.8163265  0.54243197
## 25     2.5 -0.38238451 0.7592593  0.37687475
## 26     2.6 -0.55613273 0.7068966  0.15076382
## 27     2.7 -0.59863744 0.6833333  0.08469589
## 28     2.8 -0.53873608 0.6774194  0.13868328
## 29     2.9 -0.52801764 0.6935484  0.16553075
## 30     3.0 -0.61144910 0.6615385  0.05008936
## 31     3.1 -0.66510332 0.6417910 -0.02331228
## 32     3.2 -0.76729551 0.6142857 -0.15300979
## 33     3.3 -0.77632164 0.6056338 -0.17068783
## 34     3.4 -0.80278453 0.5890411 -0.21374343
## 35     3.5 -0.84277284 0.5810811 -0.26169176
## 36     3.6 -0.87057393 0.5733333 -0.29724059
## 37     3.7 -0.87896478 0.5789474 -0.30001741
## 38     3.8 -0.87060932 0.5921053 -0.27850406
## 39     3.9 -0.87060932 0.5921053 -0.27850406
## 40     4.0 -0.91055447 0.5844156 -0.32613889
## 41     4.1 -0.82059683 0.6103896 -0.21020722
## 42     4.2 -0.81966051 0.6153846 -0.20427590
## 43     4.3 -0.80994703 0.6282051 -0.18174191
## 44     4.4 -0.74356681 0.6410256 -0.10254117
## 45     4.5 -0.73728743 0.6538462 -0.08344128
## 46     4.6 -0.73728743 0.6538462 -0.08344128
## 47     4.7 -0.68572662 0.6582278 -0.02749877
## 48     4.8 -0.67911221 0.6625000 -0.01661221
## 49     4.9 -0.70376954 0.6543210 -0.04944856
## 50     5.0 -0.72467726 0.6385542 -0.08612304
## 51     5.1 -0.70834867 0.6506024 -0.05774626
## 52     5.2 -0.70834867 0.6506024 -0.05774626
## 53     5.3 -0.70834867 0.6506024 -0.05774626
```

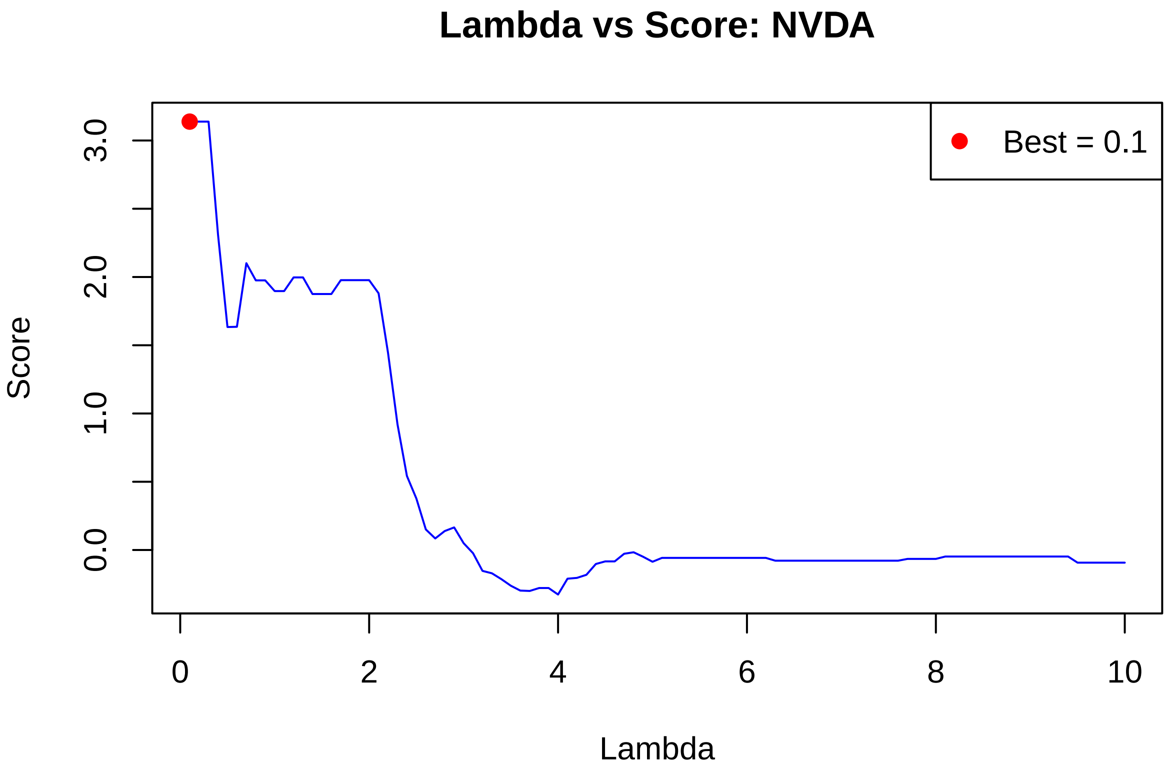
```
## 54     5.4 -0.70834867 0.6506024 -0.05774626
## 55     5.5 -0.70834867 0.6506024 -0.05774626
## 56     5.6 -0.70834867 0.6506024 -0.05774626
## 57     5.7 -0.70834867 0.6506024 -0.05774626
## 58     5.8 -0.70834867 0.6506024 -0.05774626
## 59     5.9 -0.70834867 0.6506024 -0.05774626
## 60     6.0 -0.70834867 0.6506024 -0.05774626
## 61     6.1 -0.70834867 0.6506024 -0.05774626
## 62     6.2 -0.70834867 0.6506024 -0.05774626
## 63     6.3 -0.72111918 0.6428571 -0.07826204
## 64     6.4 -0.72111918 0.6428571 -0.07826204
## 65     6.5 -0.72111918 0.6428571 -0.07826204
## 66     6.6 -0.72111918 0.6428571 -0.07826204
## 67     6.7 -0.72111918 0.6428571 -0.07826204
## 68     6.8 -0.72111918 0.6428571 -0.07826204
## 69     6.9 -0.72111918 0.6428571 -0.07826204
## 70     7.0 -0.72111918 0.6428571 -0.07826204
## 71     7.1 -0.72111918 0.6428571 -0.07826204
## 72     7.2 -0.72111918 0.6428571 -0.07826204
## 73     7.3 -0.72111918 0.6428571 -0.07826204
## 74     7.4 -0.72111918 0.6428571 -0.07826204
## 75     7.5 -0.72111918 0.6428571 -0.07826204
## 76     7.6 -0.72111918 0.6428571 -0.07826204
## 77     7.7 -0.71577891 0.6506024 -0.06517650
## 78     7.8 -0.71577891 0.6506024 -0.06517650
## 79     7.9 -0.71577891 0.6506024 -0.06517650
## 80     8.0 -0.71577891 0.6506024 -0.06517650
## 81     8.1 -0.71039033 0.6626506 -0.04773973
## 82     8.2 -0.71039033 0.6626506 -0.04773973
## 83     8.3 -0.71039033 0.6626506 -0.04773973
## 84     8.4 -0.71039033 0.6626506 -0.04773973
## 85     8.5 -0.71039033 0.6626506 -0.04773973
## 86     8.6 -0.71039033 0.6626506 -0.04773973
## 87     8.7 -0.71039033 0.6626506 -0.04773973
## 88     8.8 -0.71039033 0.6626506 -0.04773973
## 89     8.9 -0.71039033 0.6626506 -0.04773973
## 90     9.0 -0.71039033 0.6626506 -0.04773973
## 91     9.1 -0.71039033 0.6626506 -0.04773973
## 92     9.2 -0.71039033 0.6626506 -0.04773973
## 93     9.3 -0.71039033 0.6626506 -0.04773973
## 94     9.4 -0.71039033 0.6626506 -0.04773973
## 95     9.5 -0.74750396 0.6547619 -0.09274206
## 96     9.6 -0.74750396 0.6547619 -0.09274206
## 97     9.7 -0.74750396 0.6547619 -0.09274206
## 98     9.8 -0.74750396 0.6547619 -0.09274206
## 99     9.9 -0.74750396 0.6547619 -0.09274206
## 100   10.0 -0.74750396 0.6547619 -0.09274206
```

```r
print(best_row)
```

```
##   Lambda   Sharpe  WinLoss    Score
## 1    0.1 1.219875 1.918367 3.138243
```

```r
# plot Lambda vs Score
plot(score_list$Lambda, score_list$Score, type = "l", col = "blue",
```

```r
     main = "Lambda vs Score: NVDA", xlab = "Lambda", ylab = "Score")
points(best_row$Lambda, best_row$Score, col = "red", pch = 19)
legend("topright", legend = paste("Best =", round(best_row$Lambda, 2)), col = "red", pch = 19)
```



Same logic, put SOXL into the loop:

```r
# Same logic, put SOXL into the code:
symbol2 <- "SOXL"
stock_data2 <- get.hist.quote(instrument = symbol2,
                              start = "2010-01-01",
                              end = "2024-12-31",
                              quote = "AdjClose",
                              compression = "m")
```

```
## time series starts 2010-03-01
## time series ends   2024-12-01
```

```r
stock_data2 <- zoo(stock_data2, order.by = as.Date(time(stock_data2)))
colnames(stock_data2) <- "AdjClose"

ma_short2 <- rollmean(stock_data2$AdjClose, k = 12, fill = NA, align = "right")
ma_long2  <- rollmean(stock_data2$AdjClose, k = 36, fill = NA, align = "right")
diff_ma2  <- scale(ma_short2 - ma_long2)

score_list2 <- data.frame()
for (lambda in lambda_values) {
  p_ma <- sigma(lambda * diff_ma2)
  valid_index <- which(!is.na(p_ma))
  df <- data.frame(Date = index(stock_data2)[valid_index],
```

```r
                   AdjClose = coredata(stock_data2$AdjClose)[valid_index],
                   P_MA = coredata(p_ma[valid_index]))
  returns <- diff(log(df$AdjClose))
  signals <- df$P_MA[-1]
  strategy_returns <- ifelse(signals > 0.35, returns,
                             ifelse(signals < 0.15, -returns, 0))
  strategy_returns <- xts(strategy_returns, order.by = df$Date[-1])
  if (length(strategy_returns) < 5 || sd(strategy_returns, na.rm = TRUE) == 0) next
  sharpe <- as.numeric(SharpeRatio.annualized(strategy_returns, scale = 12, geometric = FALSE))
  wins <- sum(strategy_returns > 0)
  losses <- sum(strategy_returns < 0)
  win_loss <- ifelse(losses == 0, wins, wins / losses)
  score <- sharpe + win_loss
  score_list2 <- rbind(score_list2, data.frame(Symbol = symbol2, Lambda = lambda, Sharpe = sharpe, WinLo
}
best2 <- score_list2[which.max(score_list2$Score), ]
print(score_list2)
```

```
##     Symbol Lambda      Sharpe   WinLoss     Score
## 1     SOXL    0.1  0.42944795 1.3278689 1.7573168
## 2     SOXL    0.2  0.42944795 1.3278689 1.7573168
## 3     SOXL    0.3  0.38404348 1.3050847 1.6891282
## 4     SOXL    0.4  0.31829478 1.2807018 1.5989965
## 5     SOXL    0.5  0.31540185 1.2631579 1.5785597
## 6     SOXL    0.6  0.36213810 1.2857143 1.6478524
## 7     SOXL    0.7  0.32949209 1.2678571 1.5973492
## 8     SOXL    0.8  0.24205848 1.2000000 1.4420585
## 9     SOXL    0.9  0.30512432 1.2333333 1.5384577
## 10    SOXL    1.0  0.25056719 1.1935484 1.4441156
## 11    SOXL    1.1  0.25056719 1.1935484 1.4441156
## 12    SOXL    1.2  0.11579150 1.1406250 1.2564165
## 13    SOXL    1.3  0.11302134 1.1230769 1.2360983
## 14    SOXL    1.4  0.11302134 1.1230769 1.2360983
## 15    SOXL    1.5  0.15219985 1.1384615 1.2906614
## 16    SOXL    1.6  0.17495290 1.1562500 1.3312029
## 17    SOXL    1.7  0.17495290 1.1562500 1.3312029
## 18    SOXL    1.8  0.11596134 1.1093750 1.2253363
## 19    SOXL    1.9  0.01955101 1.0322581 1.0518091
## 20    SOXL    2.0  0.01282426 1.0333333 1.0461576
## 21    SOXL    2.1 -0.07204236 0.9661017 0.8940593
## 22    SOXL    2.2 -0.08138165 0.9482759 0.8668942
## 23    SOXL    2.3 -0.04570876 0.9818182 0.9361094
## 24    SOXL    2.4 -0.09094988 0.9444444 0.8534946
## 25    SOXL    2.5 -0.12021753 0.9433962 0.8231787
## 26    SOXL    2.6 -0.14642091 0.9423077 0.7958868
## 27    SOXL    2.7 -0.11436240 0.9600000 0.8456376
## 28    SOXL    2.8 -0.07777808 1.0217391 0.9439610
## 29    SOXL    2.9 -0.10757151 1.0000000 0.8924285
## 30    SOXL    3.0 -0.13452682 0.9565217 0.8219949
## 31    SOXL    3.1 -0.13452682 0.9565217 0.8219949
## 32    SOXL    3.2 -0.13452682 0.9565217 0.8219949
## 33    SOXL    3.3 -0.17271486 0.9565217 0.7838069
## 34    SOXL    3.4 -0.17271486 0.9565217 0.7838069
## 35    SOXL    3.5 -0.16976685 0.9777778 0.8080109
```

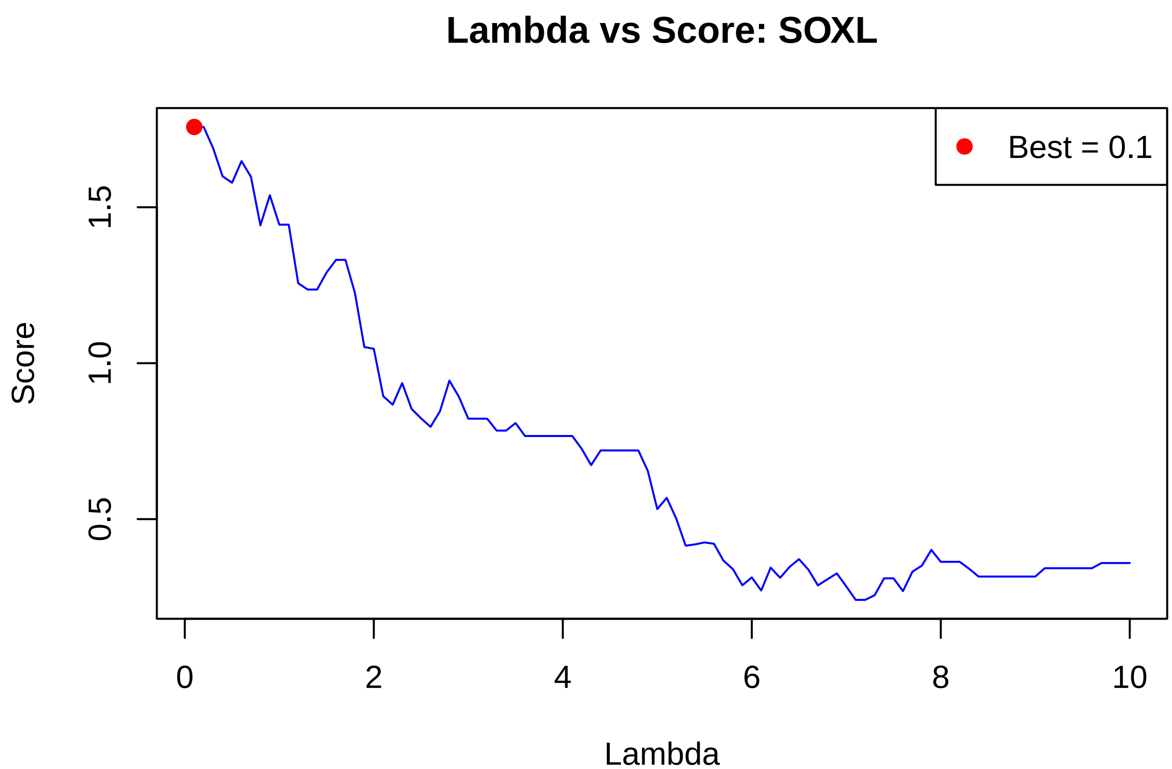
```
## 36    SOXL    3.6 -0.18894774 0.9555556 0.7666078
## 37    SOXL    3.7 -0.18894774 0.9555556 0.7666078
## 38    SOXL    3.8 -0.18894774 0.9555556 0.7666078
## 39    SOXL    3.9 -0.18894774 0.9555556 0.7666078
## 40    SOXL    4.0 -0.18894774 0.9555556 0.7666078
## 41    SOXL    4.1 -0.18894774 0.9555556 0.7666078
## 42    SOXL    4.2 -0.20786924 0.9333333 0.7254641
## 43    SOXL    4.3 -0.23806258 0.9111111 0.6730485
## 44    SOXL    4.4 -0.21306814 0.9333333 0.7202652
## 45    SOXL    4.5 -0.21306814 0.9333333 0.7202652
## 46    SOXL    4.6 -0.21306814 0.9333333 0.7202652
## 47    SOXL    4.7 -0.21306814 0.9333333 0.7202652
## 48    SOXL    4.8 -0.21306814 0.9333333 0.7202652
## 49    SOXL    4.9 -0.25688148 0.9111111 0.6542296
## 50    SOXL    5.0 -0.30106858 0.8333333 0.5322648
## 51    SOXL    5.1 -0.28895481 0.8571429 0.5681881
## 52    SOXL    5.2 -0.34081016 0.8431373 0.5023271
## 53    SOXL    5.3 -0.38133957 0.7962963 0.4149567
## 54    SOXL    5.4 -0.37723771 0.7962963 0.4190586
## 55    SOXL    5.5 -0.37475627 0.8000000 0.4252437
## 56    SOXL    5.6 -0.38257363 0.8035714 0.4209978
## 57    SOXL    5.7 -0.42621070 0.7931034 0.3668927
## 58    SOXL    5.8 -0.43993558 0.7796610 0.3397254
## 59    SOXL    5.9 -0.46615229 0.7540984 0.2879461
## 60    SOXL    6.0 -0.45740423 0.7704918 0.3130876
## 61    SOXL    6.1 -0.47492624 0.7460317 0.2711055
## 62    SOXL    6.2 -0.41355647 0.7580645 0.3445081
## 63    SOXL    6.3 -0.43413053 0.7460317 0.3119012
## 64    SOXL    6.4 -0.41503974 0.7619048 0.3468650
## 65    SOXL    6.5 -0.40607910 0.7777778 0.3716987
## 66    SOXL    6.6 -0.42804511 0.7656250 0.3375799
## 67    SOXL    6.7 -0.45484945 0.7424242 0.2875748
## 68    SOXL    6.8 -0.45037249 0.7575758 0.3072033
## 69    SOXL    6.9 -0.44685194 0.7727273 0.3258753
## 70    SOXL    7.0 -0.47703495 0.7611940 0.2841591
## 71    SOXL    7.1 -0.50887189 0.7500000 0.2411281
## 72    SOXL    7.2 -0.50887189 0.7500000 0.2411281
## 73    SOXL    7.3 -0.50870747 0.7647059 0.2559984
## 74    SOXL    7.4 -0.47238393 0.7826087 0.3102248
## 75    SOXL    7.5 -0.47238393 0.7826087 0.3102248
## 76    SOXL    7.6 -0.50216617 0.7714286 0.2692624
## 77    SOXL    7.7 -0.46846858 0.8000000 0.3315314
## 78    SOXL    7.8 -0.46333367 0.8142857 0.3509520
## 79    SOXL    7.9 -0.42736090 0.8285714 0.4012105
## 80    SOXL    8.0 -0.45380333 0.8169014 0.3630981
## 81    SOXL    8.1 -0.45380333 0.8169014 0.3630981
## 82    SOXL    8.2 -0.45380333 0.8169014 0.3630981
## 83    SOXL    8.3 -0.46468377 0.8055556 0.3408718
## 84    SOXL    8.4 -0.47872593 0.7945205 0.3157946
## 85    SOXL    8.5 -0.47872593 0.7945205 0.3157946
## 86    SOXL    8.6 -0.47872593 0.7945205 0.3157946
## 87    SOXL    8.7 -0.47872593 0.7945205 0.3157946
## 88    SOXL    8.8 -0.47872593 0.7945205 0.3157946
## 89    SOXL    8.9 -0.47872593 0.7945205 0.3157946
```

```
## 90     SOXL    9.0 -0.47872593 0.7945205 0.3157946
## 91     SOXL    9.1 -0.46563680 0.8082192 0.3425824
## 92     SOXL    9.2 -0.46563680 0.8082192 0.3425824
## 93     SOXL    9.3 -0.46563680 0.8082192 0.3425824
## 94     SOXL    9.4 -0.46563680 0.8082192 0.3425824
## 95     SOXL    9.5 -0.46563680 0.8082192 0.3425824
## 96     SOXL    9.6 -0.46563680 0.8082192 0.3425824
## 97     SOXL    9.7 -0.46272148 0.8219178 0.3591963
## 98     SOXL    9.8 -0.46272148 0.8219178 0.3591963
## 99     SOXL    9.9 -0.46272148 0.8219178 0.3591963
## 100    SOXL   10.0 -0.46272148 0.8219178 0.3591963
```

```
print(best2)
```

```
##   Symbol Lambda   Sharpe  WinLoss    Score
## 1   SOXL    0.1 0.429448 1.327869 1.757317
```

```
plot(score_list2$Lambda, score_list2$Score, type = "l", col = "blue",
     main = "Lambda vs Score: SOXL", xlab = "Lambda", ylab = "Score")
points(best2$Lambda, best2$Score, col = "red", pch = 19)
legend("topright", legend = paste("Best =", round(best2$Lambda, 2)), col = "red", pch = 19)
```



Same logic, put XOM into the loop:

```
symbol3 <- "XOM"
stock_data3 <- get.hist.quote(instrument = symbol3,
                              start = "2010-01-01",
                              end = "2024-12-31",
                              quote = "AdjClose",
                              compression = "m")
```

```
## time series ends   2024-12-01
```

```
stock_data3 <- zoo(stock_data3, order.by = as.Date(time(stock_data3)))
colnames(stock_data3) <- "AdjClose"

ma_short3 <- rollmean(stock_data3$AdjClose, k = 12, fill = NA, align = "right")
ma_long3  <- rollmean(stock_data3$AdjClose, k = 36, fill = NA, align = "right")
diff_ma3  <- scale(ma_short3 - ma_long3)

score_list3 <- data.frame()
for (lambda in lambda_values) {
  p_ma <- sigma(lambda * diff_ma3)
  valid_index <- which(!is.na(p_ma))
  df <- data.frame(Date = index(stock_data3)[valid_index],
                   AdjClose = coredata(stock_data3$AdjClose)[valid_index],
                   P_MA = coredata(p_ma[valid_index]))
  returns <- diff(log(df$AdjClose))
  signals <- df$P_MA[-1]
  strategy_returns <- ifelse(signals > 0.38, returns,
                             ifelse(signals < 0.18, -returns, 0))
  strategy_returns <- xts(strategy_returns, order.by = df$Date[-1])
  if (length(strategy_returns) < 5 || sd(strategy_returns, na.rm = TRUE) == 0) next
  sharpe <- as.numeric(SharpeRatio.annualized(strategy_returns, scale = 12, geometric = FALSE))
  wins <- sum(strategy_returns > 0)
  losses <- sum(strategy_returns < 0)
  win_loss <- ifelse(losses == 0, wins, wins / losses)
  score <- sharpe + win_loss
  score_list3 <- rbind(score_list3, data.frame(Symbol = symbol3, Lambda = lambda, Sharpe = sharpe, WinLo
}
best3 <- score_list3[which.max(score_list3$Score), ]
print(score_list3)
```

```
##    Symbol Lambda       Sharpe  WinLoss    Score
## 1     XOM    0.1  0.233136361 1.149254 1.382390
## 2     XOM    0.2  0.233136361 1.149254 1.382390
## 3     XOM    0.3  0.060121483 1.089552 1.149674
## 4     XOM    0.4  0.135321161 1.111111 1.246432
## 5     XOM    0.5  0.143808334 1.114754 1.258562
## 6     XOM    0.6  0.217870587 1.135593 1.353464
## 7     XOM    0.7  0.177319897 1.105263 1.282583
## 8     XOM    0.8  0.187719571 1.111111 1.298831
## 9     XOM    0.9  0.030118518 1.092593 1.122711
## 10    XOM    1.0  0.007804868 1.056604 1.064409
## 11    XOM    1.1 -0.002259589 1.018868 1.016608
## 12    XOM    1.2  0.134021871 1.080000 1.214022
## 13    XOM    1.3  0.079001988 1.060000 1.139002
## 14    XOM    1.4  0.075567856 1.021277 1.096844
## 15    XOM    1.5  0.156409809 1.175000 1.331410
## 16    XOM    1.6  0.057841864 1.121951 1.179793
## 17    XOM    1.7  0.100169920 1.205128 1.305298
## 18    XOM    1.8  0.127129373 1.243243 1.370373
## 19    XOM    1.9  0.241051356 1.305556 1.546607
## 20    XOM    2.0  0.186558426 1.230769 1.417328
```

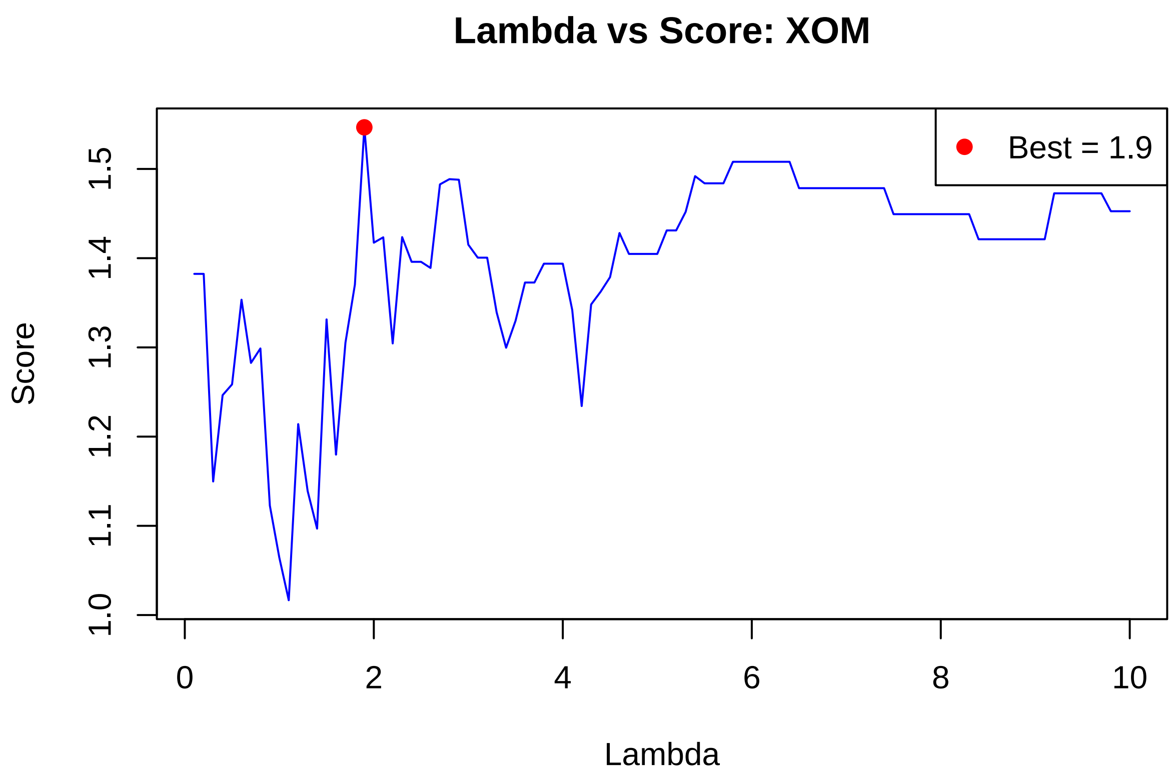
```
## 21     XOM    2.1  0.198394540 1.225000 1.423395
## 22     XOM    2.2  0.137743094 1.166667 1.304410
## 23     XOM    2.3  0.209319520 1.214286 1.423605
## 24     XOM    2.4  0.186595124 1.209302 1.395897
## 25     XOM    2.5  0.186595124 1.209302 1.395897
## 26     XOM    2.6  0.184478171 1.204545 1.389024
## 27     XOM    2.7  0.232695105 1.250000 1.482695
## 28     XOM    2.8  0.238545335 1.250000 1.488545
## 29     XOM    2.9  0.243475404 1.244444 1.487920
## 30     XOM    3.0  0.223658949 1.191489 1.415148
## 31     XOM    3.1  0.213059590 1.187500 1.400560
## 32     XOM    3.2  0.213059590 1.187500 1.400560
## 33     XOM    3.3  0.179279706 1.160000 1.339280
## 34     XOM    3.4  0.162351125 1.137255 1.299606
## 35     XOM    3.5  0.172947384 1.156863 1.329810
## 36     XOM    3.6  0.221766087 1.150943 1.372709
## 37     XOM    3.7  0.221766087 1.150943 1.372709
## 38     XOM    3.8  0.223986657 1.169811 1.393798
## 39     XOM    3.9  0.223986657 1.169811 1.393798
## 40     XOM    4.0  0.223986657 1.169811 1.393798
## 41     XOM    4.1  0.196428604 1.145455 1.341883
## 42     XOM    4.2  0.148027706 1.086207 1.234235
## 43     XOM    4.3  0.210196716 1.137931 1.348128
## 44     XOM    4.4  0.212329640 1.150000 1.362330
## 45     XOM    4.5  0.214766108 1.163934 1.378701
## 46     XOM    4.6  0.231441463 1.196721 1.428163
## 47     XOM    4.7  0.227337672 1.177419 1.404757
## 48     XOM    4.8  0.227337672 1.177419 1.404757
## 49     XOM    4.9  0.227337672 1.177419 1.404757
## 50     XOM    5.0  0.227337672 1.177419 1.404757
## 51     XOM    5.1  0.237549863 1.193548 1.431098
## 52     XOM    5.2  0.237549863 1.193548 1.431098
## 53     XOM    5.3  0.242241789 1.209677 1.451919
## 54     XOM    5.4  0.266102559 1.225806 1.491909
## 55     XOM    5.5  0.261675486 1.222222 1.483898
## 56     XOM    5.6  0.261675486 1.222222 1.483898
## 57     XOM    5.7  0.261675486 1.222222 1.483898
## 58     XOM    5.8  0.269919744 1.238095 1.508015
## 59     XOM    5.9  0.269919744 1.238095 1.508015
## 60     XOM    6.0  0.269919744 1.238095 1.508015
## 61     XOM    6.1  0.269919744 1.238095 1.508015
## 62     XOM    6.2  0.269919744 1.238095 1.508015
## 63     XOM    6.3  0.269919744 1.238095 1.508015
## 64     XOM    6.4  0.269919744 1.238095 1.508015
## 65     XOM    6.5  0.256216876 1.222222 1.478439
## 66     XOM    6.6  0.256216876 1.222222 1.478439
## 67     XOM    6.7  0.256216876 1.222222 1.478439
## 68     XOM    6.8  0.256216876 1.222222 1.478439
## 69     XOM    6.9  0.256216876 1.222222 1.478439
## 70     XOM    7.0  0.256216876 1.222222 1.478439
## 71     XOM    7.1  0.256216876 1.222222 1.478439
## 72     XOM    7.2  0.256216876 1.222222 1.478439
## 73     XOM    7.3  0.256216876 1.222222 1.478439
## 74     XOM    7.4  0.256216876 1.222222 1.478439
```

```
## 75     XOM    7.5  0.242941829 1.206349 1.449291
## 76     XOM    7.6  0.242941829 1.206349 1.449291
## 77     XOM    7.7  0.242941829 1.206349 1.449291
## 78     XOM    7.8  0.242941829 1.206349 1.449291
## 79     XOM    7.9  0.242941829 1.206349 1.449291
## 80     XOM    8.0  0.242941829 1.206349 1.449291
## 81     XOM    8.1  0.242941829 1.206349 1.449291
## 82     XOM    8.2  0.242941829 1.206349 1.449291
## 83     XOM    8.3  0.242941829 1.206349 1.449291
## 84     XOM    8.4  0.233686269 1.187500 1.421186
## 85     XOM    8.5  0.233686269 1.187500 1.421186
## 86     XOM    8.6  0.233686269 1.187500 1.421186
## 87     XOM    8.7  0.233686269 1.187500 1.421186
## 88     XOM    8.8  0.233686269 1.187500 1.421186
## 89     XOM    8.9  0.233686269 1.187500 1.421186
## 90     XOM    9.0  0.233686269 1.187500 1.421186
## 91     XOM    9.1  0.233686269 1.187500 1.421186
## 92     XOM    9.2  0.266332572 1.206349 1.472682
## 93     XOM    9.3  0.266332572 1.206349 1.472682
## 94     XOM    9.4  0.266332572 1.206349 1.472682
## 95     XOM    9.5  0.266332572 1.206349 1.472682
## 96     XOM    9.6  0.266332572 1.206349 1.472682
## 97     XOM    9.7  0.266332572 1.206349 1.472682
## 98     XOM    9.8  0.262101374 1.190476 1.452578
## 99     XOM    9.9  0.262101374 1.190476 1.452578
## 100    XOM   10.0  0.262101374 1.190476 1.452578
```

```r
print(best3)
```

```
##    Symbol Lambda    Sharpe  WinLoss    Score
## 19    XOM    1.9 0.2410514 1.305556 1.546607
```

```r
plot(score_list3$Lambda, score_list3$Score, type = "l", col = "blue",
     main = "Lambda vs Score: XOM", xlab = "Lambda", ylab = "Score")
points(best3$Lambda, best3$Score, col = "red", pch = 19)
legend("topright", legend = paste("Best =", round(best3$Lambda, 2)), col = "red", pch = 19)
```



Same logic, put CLS.TO into the loop:

```
symbol4 <- "CLS.TO"
stock_data4 <- get.hist.quote(instrument = symbol4,
                              start = "2010-01-01",
                              end = "2024-12-31",
                              quote = "AdjClose",
                              compression = "m")
```

```
## time series ends   2024-12-01
```

```
stock_data4 <- zoo(stock_data4, order.by = as.Date(time(stock_data4)))
colnames(stock_data4) <- "AdjClose"

ma_short4 <- rollmean(stock_data4$AdjClose, k = 12, fill = NA, align = "right")
ma_long4  <- rollmean(stock_data4$AdjClose, k = 36, fill = NA, align = "right")
diff_ma4  <- scale(ma_short4 - ma_long4)

score_list4 <- data.frame()
for (lambda in lambda_values) {
  p_ma <- sigma(lambda * diff_ma4)
  valid_index <- which(!is.na(p_ma))
  df <- data.frame(Date = index(stock_data4)[valid_index],
                   AdjClose = coredata(stock_data4$AdjClose)[valid_index],
                   P_MA = coredata(p_ma[valid_index]))
  returns <- diff(log(df$AdjClose))
  signals <- df$P_MA[-1]
  strategy_returns <- ifelse(signals > 0.49, returns,
                             ifelse(signals < 0.29, -returns, 0))
```

```r
  strategy_returns <- xts(strategy_returns, order.by = df$Date[-1])
  if (length(strategy_returns) < 5 || sd(strategy_returns, na.rm = TRUE) == 0) next
  sharpe <- as.numeric(SharpeRatio.annualized(strategy_returns, scale = 12, geometric = FALSE))
  wins <- sum(strategy_returns > 0)
  losses <- sum(strategy_returns < 0)
  win_loss <- ifelse(losses == 0, wins, wins / losses)
  score <- sharpe + win_loss
  score_list4 <- rbind(score_list4, data.frame(Symbol = symbol4, Lambda = lambda, Sharpe = sharpe, WinL
}
best4 <- score_list4[which.max(score_list4$Score), ]
print(score_list4)
```

```
##    Symbol Lambda    Sharpe  WinLoss    Score
## 1  CLS.TO    0.1 0.7340868 1.621622 2.355708
## 2  CLS.TO    0.2 0.5760696 1.551724 2.127794
## 3  CLS.TO    0.3 0.6574532 1.640000 2.297453
## 4  CLS.TO    0.4 0.7064382 1.809524 2.515962
## 5  CLS.TO    0.5 0.7048523 1.761905 2.466757
## 6  CLS.TO    0.6 0.7072301 2.000000 2.707230
## 7  CLS.TO    0.7 0.6788313 2.000000 2.678831
## 8  CLS.TO    0.8 0.6718558 1.941176 2.613032
## 9  CLS.TO    0.9 0.6686852 2.000000 2.668685
## 10 CLS.TO    1.0 0.7006710 2.000000 2.700671
## 11 CLS.TO    1.1 0.5026705 1.523810 2.026480
## 12 CLS.TO    1.2 0.5037274 1.545455 2.049182
## 13 CLS.TO    1.3 0.4916580 1.478261 1.969919
## 14 CLS.TO    1.4 0.4801318 1.370370 1.850502
## 15 CLS.TO    1.5 0.4762047 1.321429 1.797633
## 16 CLS.TO    1.6 0.5079815 1.407407 1.915389
## 17 CLS.TO    1.7 0.5097442 1.444444 1.954189
## 18 CLS.TO    1.8 0.5150207 1.428571 1.943592
## 19 CLS.TO    1.9 0.5170656 1.406250 1.923316
## 20 CLS.TO    2.0 0.4969788 1.314286 1.811264
## 21 CLS.TO    2.1 0.4990867 1.305556 1.804642
## 22 CLS.TO    2.2 0.4691679 1.270270 1.739438
## 23 CLS.TO    2.3 0.4576222 1.263158 1.720780
## 24 CLS.TO    2.4 0.4326417 1.200000 1.632642
## 25 CLS.TO    2.5 0.4326417 1.200000 1.632642
## 26 CLS.TO    2.6 0.4456201 1.225000 1.670620
## 27 CLS.TO    2.7 0.4429093 1.195122 1.638031
## 28 CLS.TO    2.8 0.4257479 1.195122 1.620870
## 29 CLS.TO    2.9 0.4309012 1.219512 1.650413
## 30 CLS.TO    3.0 0.4250546 1.190476 1.615531
## 31 CLS.TO    3.1 0.3763152 1.086957 1.463272
## 32 CLS.TO    3.2 0.3884633 1.108696 1.497159
## 33 CLS.TO    3.3 0.3597741 1.085106 1.444881
## 34 CLS.TO    3.4 0.3622047 1.108696 1.470900
## 35 CLS.TO    3.5 0.3820351 1.152174 1.534209
## 36 CLS.TO    3.6 0.3666326 1.127660 1.494292
## 37 CLS.TO    3.7 0.3736617 1.148936 1.522598
## 38 CLS.TO    3.8 0.3736617 1.148936 1.522598
## 39 CLS.TO    3.9 0.3457849 1.102041 1.447826
## 40 CLS.TO    4.0 0.3457849 1.102041 1.447826
## 41 CLS.TO    4.1 0.3581523 1.122449 1.480601
```

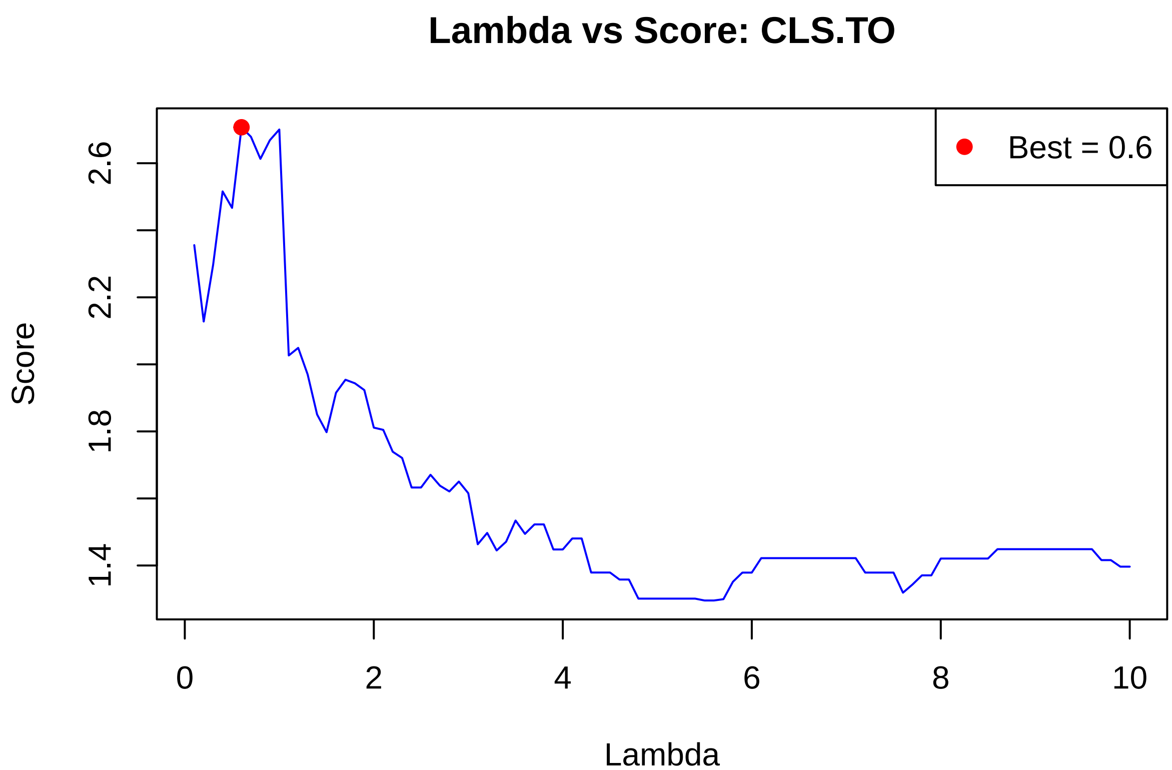
```
## 42  CLS.TO    4.2 0.3581523 1.122449 1.480601
## 43  CLS.TO    4.3 0.3006313 1.078431 1.379063
## 44  CLS.TO    4.4 0.3006313 1.078431 1.379063
## 45  CLS.TO    4.5 0.3006313 1.078431 1.379063
## 46  CLS.TO    4.6 0.3004435 1.057692 1.358136
## 47  CLS.TO    4.7 0.3004435 1.057692 1.358136
## 48  CLS.TO    4.8 0.2826702 1.018519 1.301189
## 49  CLS.TO    4.9 0.2826702 1.018519 1.301189
## 50  CLS.TO    5.0 0.2826702 1.018519 1.301189
## 51  CLS.TO    5.1 0.2826702 1.018519 1.301189
## 52  CLS.TO    5.2 0.2826702 1.018519 1.301189
## 53  CLS.TO    5.3 0.2826702 1.018519 1.301189
## 54  CLS.TO    5.4 0.2826702 1.018519 1.301189
## 55  CLS.TO    5.5 0.2771398 1.018519 1.295658
## 56  CLS.TO    5.6 0.2771398 1.018519 1.295658
## 57  CLS.TO    5.7 0.2813271 1.018182 1.299509
## 58  CLS.TO    5.8 0.3149954 1.036364 1.351359
## 59  CLS.TO    5.9 0.3242712 1.054545 1.378817
## 60  CLS.TO    6.0 0.3242712 1.054545 1.378817
## 61  CLS.TO    6.1 0.3478445 1.074074 1.421919
## 62  CLS.TO    6.2 0.3478445 1.074074 1.421919
## 63  CLS.TO    6.3 0.3478445 1.074074 1.421919
## 64  CLS.TO    6.4 0.3478445 1.074074 1.421919
## 65  CLS.TO    6.5 0.3478445 1.074074 1.421919
## 66  CLS.TO    6.6 0.3478445 1.074074 1.421919
## 67  CLS.TO    6.7 0.3478445 1.074074 1.421919
## 68  CLS.TO    6.8 0.3478445 1.074074 1.421919
## 69  CLS.TO    6.9 0.3478445 1.074074 1.421919
## 70  CLS.TO    7.0 0.3478445 1.074074 1.421919
## 71  CLS.TO    7.1 0.3478445 1.074074 1.421919
## 72  CLS.TO    7.2 0.3243460 1.054545 1.378891
## 73  CLS.TO    7.3 0.3243460 1.054545 1.378891
## 74  CLS.TO    7.4 0.3243460 1.054545 1.378891
## 75  CLS.TO    7.5 0.3243460 1.054545 1.378891
## 76  CLS.TO    7.6 0.3013348 1.017544 1.318879
## 77  CLS.TO    7.7 0.3080870 1.035088 1.343175
## 78  CLS.TO    7.8 0.3179075 1.052632 1.370539
## 79  CLS.TO    7.9 0.3179075 1.052632 1.370539
## 80  CLS.TO    8.0 0.3505827 1.070175 1.420758
## 81  CLS.TO    8.1 0.3505827 1.070175 1.420758
## 82  CLS.TO    8.2 0.3505827 1.070175 1.420758
## 83  CLS.TO    8.3 0.3505827 1.070175 1.420758
## 84  CLS.TO    8.4 0.3505827 1.070175 1.420758
## 85  CLS.TO    8.5 0.3505827 1.070175 1.420758
## 86  CLS.TO    8.6 0.3609792 1.087719 1.448698
## 87  CLS.TO    8.7 0.3609792 1.087719 1.448698
## 88  CLS.TO    8.8 0.3609792 1.087719 1.448698
## 89  CLS.TO    8.9 0.3609792 1.087719 1.448698
## 90  CLS.TO    9.0 0.3609792 1.087719 1.448698
## 91  CLS.TO    9.1 0.3609792 1.087719 1.448698
## 92  CLS.TO    9.2 0.3609792 1.087719 1.448698
## 93  CLS.TO    9.3 0.3609792 1.087719 1.448698
## 94  CLS.TO    9.4 0.3609792 1.087719 1.448698
## 95  CLS.TO    9.5 0.3609792 1.087719 1.448698
```

```
## 96   CLS.TO     9.6 0.3609792 1.087719 1.448698
## 97   CLS.TO     9.7 0.3458800 1.070175 1.416055
## 98   CLS.TO     9.8 0.3458800 1.070175 1.416055
## 99   CLS.TO     9.9 0.3448761 1.051724 1.396600
## 100 CLS.TO    10.0 0.3448761 1.051724 1.396600
```

```r
print(best4)
```

```
##   Symbol Lambda    Sharpe WinLoss   Score
## 6 CLS.TO    0.6 0.7072301       2 2.70723
```

```r
plot(score_list4$Lambda, score_list4$Score, type = "l", col = "blue",
     main = "Lambda vs Score: CLS.TO", xlab = "Lambda", ylab = "Score")
points(best4$Lambda, best4$Score, col = "red", pch = 19)
legend("topright", legend = paste("Best =", round(best4$Lambda, 2)), col = "red", pch = 19)
```



Put a list of all stocks' P_MA:

```r
# Function of P_MA with best lambda:
get_pma <- function(stock_data, lambda, short_k = 12, long_k = 36) {
  ma_short <- rollmean(stock_data$AdjClose, k = short_k, fill = NA, align = "right")
  ma_long  <- rollmean(stock_data$AdjClose, k = long_k, fill = NA, align = "right")
  diff_ma  <- scale(ma_short - ma_long)
  p_ma <- 1 / (1 + exp(-lambda * diff_ma))
  return(p_ma)
}

# take best lambda:
p_ma_nvda <- get_pma(stock_data,     lambda = 0.1)  # NVDA
p_ma_soxl <- get_pma(stock_data2,    lambda = 0.1)  # SOXL
p_ma_xom  <- get_pma(stock_data3,    lambda = 1.9)  # XOM
p_ma_cls  <- get_pma(stock_data4,    lambda = 0.6)  # CLS.TO
```

```r
pma_list <- list(
  "CLS.TO" = p_ma_cls,
  "NVDA"   = p_ma_nvda,
  "XOM"    = p_ma_xom,
  "SOXL"   = p_ma_soxl
)

library(knitr)

# Check the tail of each stocks:
tail_df <- data.frame(
  Date    = tail(index(stock_data)),
  NVDA    = tail(na.omit(p_ma_nvda)),
  SOXL    = tail(na.omit(p_ma_soxl)),
  XOM     = tail(na.omit(p_ma_xom)),
  CLS.TO  = tail(na.omit(p_ma_cls))
)

kable(tail_df, caption = "Tail of P_MA for Each Stock (Last 6 Months)")
```

Table 1: Tail of P_MA for Each Stock (Last 6 Months)

	Date	NVDA	SOXL	XOM	CLS.TO
2024-07-01	2024-07-01	0.5750065	0.5108169	0.8676680	0.8687553
2024-08-01	2024-08-01	0.5829288	0.5162514	0.8456944	0.8829694
2024-09-01	2024-09-01	0.5922005	0.5221241	0.8092400	0.8947299
2024-10-01	2024-10-01	0.6038610	0.5286282	0.7975335	0.9162683
2024-11-01	2024-11-01	0.6154123	0.5339862	0.7910554	0.9392021
2024-12-01	2024-12-01	0.6255272	0.5368668	0.7748272	0.9580058