Strategy Design (ML Fin Data - Project 1)

Hair Albeiro Parra Barrera

Libraries

0. Scraping the SP500

In order to test the logic within the strategy, I have fetched functions that retrieve a number of sample stocks by sector from the SP500.

```r
# to obtain relative paths
library(here)

# Load code into environment
source(here("functions", "fetch_sp500_sectors.R"))
```

```
## Getting holdings for SP500
```

0.0.1 SP500 Economic Sectors

The following function fetches and extract the economic sectors from the SP500, taken from Wikipedia.

```r
# fetch the sectors as a dataframe
sp500_sectors <- f_get_sp500_sectors()
head(sp500_sectors)
```

```
##   tickers                sectors
## 1     MMM            Industrials
## 2     AOS            Industrials
## 3     ABT            Health Care
## 4    ABBV            Health Care
## 5     ACN Information Technology
## 6    ATVI Communication Services
```

0.0.2 SP500 Sector Weight

```r
# wrap into a single argument funciton
fetch_sp500_sector_data <- function(x){f_fetch_sector_data(x, sp500, sp500_sectors)}

# call the function
head(fetch_sp500_sector_data("Information Technology"))
```

```
##   ticker                 sector      weight shares_held
## 1   AAPL Information Technology 0.070819853   164712193
## 2    ACN Information Technology 0.005393819     7070903
## 3   ADBE Information Technology 0.006588889     5109299
## 4    ADI Information Technology 0.002440011     5620612
## 5   ADSK Information Technology 0.001238980     2395718
## 6   AKAM Information Technology 0.000452252     1710719
```

0.0.3 Retrieving top sectors and stocks

Pack everything into one function to retrieve all the data

```r
# Retrieve top 10 stocks by weight for each sector in the top 5 sectors from the SP500 (by weight)
sector_list <- f_retrieve_top_sp500(top_n_sectors = 6, top_n_stocks = 15, only_tickers=TRUE)
sector_list
```

```
## $Industrials
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
##
## $`Health Care`
##  [1] "ABBV" "ABT"  "AMGN" "BMY"  "DHR"  "ELV"  "ISRG" "JNJ"  "LLY"  "MDT"
## [11] "MRK"  "PFE"  "SYK"  "TMO"  "UNH"
##
## $`Information Technology`
##  [1] "AAPL" "ACN"  "ADBE" "AMD"  "AVGO" "CRM"  "CSCO" "IBM"  "INTC" "INTU"
## [11] "MSFT" "NVDA" "ORCL" "QCOM" "TXN"
##
## $`Communication Services`
##  [1] "ATVI"  "CHTR"  "CMCSA" "DIS"   "EA"    "GOOG"  "GOOGL" "META"  "NFLX"
## [10] "OMC"   "T"     "TMUS"  "TTWO"  "VZ"    "WBD"
##
## $Financials
##  [1] "AXP"  "BAC"  "BLK"  "C"    "CB"   "GS"   "JPM"  "MA"   "MMC"  "MS"
## [11] "PGR"  "SCHW" "SPGI" "V"    "WFC"
##
## $`Consumer Discretionary`
##  [1] "ABNB" "AMZN" "AZO"  "BKNG" "CMG"  "F"    "GM"   "HD"   "MAR"  "MCD"
## [11] "NKE"  "ORLY" "SBUX" "TJX"  "TSLA"
```

This logic is implemented under **functions/fetch_sp500_sectors.R**

0.0.4 Retrieving top sectors and stocks

```r
# function to fetch all the information for one ticker into a nice xts dataframe
sp500_stocks <- lapply(sector_list,
                       f_fetch_all_tickers,
                       start_date="2016-01-01",
                       end_date="2022-12-01")
```

```r
# Show the available sectors
names(sp500_stocks)
```

```
## [1] "Industrials"            "Health Care"            "Information Technology"
## [4] "Communication Services" "Financials"             "Consumer Discretionary"
```

```r
# Show available stocks for Industrials
names(sp500_stocks$Industrials)
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```r
# access the xts of the stocks in industrials
tail(sp500_stocks$Industrials$ADP)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2022-10-26              1      0.009733913    0.008113008   0.039930970
## 2022-11-02              1      0.012306040    0.009733913   0.008113008
## 2022-11-09              1      0.053616090    0.012306040   0.009733913
## 2022-11-16              1      0.034718700    0.053616090   0.012306040
## 2022-11-23              1      0.005923517    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923517   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535730   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535730  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113008   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733913   0.008113008 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012306040   0.009733913 10.243009 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012306040 10.247795 16.53998   100 0.8907336
##            chaikin_vol        clv         emv     macd      mfi      sar
## 2022-10-26 -1.49750300 -0.1320576 -0.01707202 2.049576 51.52422 260.0428
## 2022-11-02  2.90314600 -0.2863719  0.02711271 1.939312 49.23300 258.6055
## 2022-11-09 -0.09676625 -0.3920529  0.04765004 1.866926 49.20839 257.2257
## 2022-11-16 -0.38397100 -0.4461119  0.09074850 1.906715 48.83463 256.7200
## 2022-11-23 -0.20180520 -0.3205142  0.11758529 2.068291 49.31528 224.1100
## 2022-11-30  0.48394890 -0.1089895  0.12144667 2.300754 42.97382 224.1100
##                  smi     volat month_index
## 2022-10-26  8.131402 0.2269538          82
## 2022-11-02  5.546375 0.2606250          83
## 2022-11-09  3.943960 0.2653165          83
## 2022-11-16  6.291102 0.2641173          83
## 2022-11-23 11.099826 0.2624611          83
## 2022-11-30 16.713518 0.2759187          83
```

BACKTESTING LOGIC

Adding a numeric index

First, we need to create a corresponding index for each week:

```r
# count number of weeks in data from one of the dataframes
sample_xts <- sp500_stocks$Industrials$CSX
tail(sample_xts, 10)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2022-09-28              1      0.006853095   -0.053209662  -0.069267283
## 2022-10-05             -1     -0.042966082    0.006853095  -0.053209662
## 2022-10-12              1      0.046554111   -0.042966082   0.006853095
## 2022-10-19              1      0.029989991    0.046554111  -0.042966082
## 2022-10-26             -1     -0.008377096    0.029989991   0.046554111
## 2022-11-02              1      0.031058456   -0.008377096   0.029989991
## 2022-11-09              1      0.059684655    0.031058456  -0.008377096
## 2022-11-16              1      0.026221770    0.059684655   0.031058456
## 2022-11-23              1      0.022307721    0.026221770   0.059684655
## 2022-11-30             NA               NA    0.022307721   0.026221770
##            adjclose_lag2 adjclose_lag3      atr      adx aaron         bb
## 2022-09-28  -0.020913351   0.007554347 1.441481 16.24190  -100 0.04467755
## 2022-10-05  -0.069267283  -0.020913351 1.384232 17.10559   -50 0.13495813
## 2022-10-12  -0.053209662  -0.069267283 1.379644 18.24157   -50 0.07457368
```

```
## 2022-10-19   0.006853095  -0.053209662 1.394670 18.58490    50 0.23730603
## 2022-10-26  -0.042966082   0.006853095 1.398622 18.20787   100 0.36428555
## 2022-11-02   0.046554111  -0.042966082 1.385863 17.63796   100 0.36718737
## 2022-11-09   0.029989991   0.046554111 1.385444 17.00435    50 0.43456871
## 2022-11-16  -0.008377096   0.029989991 1.429341 16.04316   100 0.61239403
## 2022-11-23   0.031058456  -0.008377096 1.395102 15.54651   100 0.68335600
## 2022-11-30   0.059684655   0.031058456 1.369024 15.36369   100 0.70213009
##              chaikin_vol         clv           emv      macd      mfi      sar
## 2022-09-28  2.43234200  0.21475805 -1.787304e-04 -2.031918 46.90353 34.67000
## 2022-10-05 -0.44268680  0.22116568 -2.096124e-04 -2.290153 46.43088 34.38840
## 2022-10-12  0.43839330  0.07934922 -3.472192e-04 -2.649750 46.62430 34.11806
## 2022-10-19 -1.12835800  0.03125187 -3.458817e-04 -2.983549 54.92321 33.66998
## 2022-10-26  0.36773750 -0.10430028 -2.858648e-04 -3.232381 56.20916 33.24878
## 2022-11-02 -8.91414900 -0.26417408 -1.913069e-04 -3.420978 48.82911 32.85285
## 2022-11-09 -0.08886197 -0.35167976 -1.696224e-04 -3.505779 48.94612 32.48068
## 2022-11-16 -0.69757770 -0.28307675 -6.177828e-05 -3.415472 46.83053 32.13084
## 2022-11-23 -2.77541900 -0.16462184  6.920197e-05 -3.168499 45.87661 26.65000
## 2022-11-30 -0.65517410  0.02947430  2.043992e-04 -2.797269 55.72098 26.65000
##                    smi     volat month_index
## 2022-09-28 -18.01681 0.2279791          81
## 2022-10-05 -22.89976 0.2353109          82
## 2022-10-12 -28.89441 0.2481376          82
## 2022-10-19 -32.89471 0.2465206          82
## 2022-10-26 -34.78229 0.2484444          82
## 2022-11-02 -36.26677 0.2806964          83
## 2022-11-09 -36.24474 0.2819226          83
## 2022-11-16 -32.84559 0.2767814          83
## 2022-11-23 -26.53377 0.2587499          83
## 2022-11-30 -18.89848 0.2672197          83
```

```r
sample_xts[, c( "month_index")]
```

```
##            month_index
## 2016-01-06           1
## 2016-01-13           1
## 2016-01-20           1
## 2016-01-27           1
## 2016-02-03           2
## 2016-02-10           2
## 2016-02-17           2
## 2016-02-24           2
## 2016-03-02           3
## 2016-03-09           3
##        ...
## 2022-09-28          81
## 2022-10-05          82
## 2022-10-12          82
## 2022-10-19          82
## 2022-10-26          82
## 2022-11-02          83
## 2022-11-09          83
## 2022-11-16          83
## 2022-11-23          83
## 2022-11-30          83
```

BACKTESTING_PROCEDURE

1. Assume we have N_{years} years of weekly data, giving a total of N_{months} many months. 2. We want to fix a window of $N_W = 12$ months at the time (i.e. a year of data).

2. The total number of runs is given by

$$N^{runs} = \left\lfloor \frac{N_{months} - N_W}{s} \right\rfloor + 1$$

, where $s = 1$ is the number of months to move at the time (because of monthly rebalance).

i.e., we can move N^{runs} times when predicting one month at the time, starting with having all the data until month 12.

That is, $\tau = 1, \ldots, 48$

```
# Set up backtesting simulation parameters
sample_xts <- sp500_stocks$Industrials$ADP
sectors <- names(sp500_stocks)
N_sector_best_stocks <- 3 # new strategy: 3x2 = 6

# Formula parameters
slide <- 1
N_months <- length(names(split.xts(sample_xts, f= "months")))
N_window <- 24 # number of months in size for each window
N_runs <- floor((N_months - N_window)/slide)

# display parameters
print(paste0("N_months: ", N_months))
```

```
## [1] "N_months: 83"
```

```
print(paste0("N_runs: ", N_runs))
```

```
## [1] "N_runs: 59"
```

```
print(paste0("slide: ", slide))
```

```
## [1] "slide: 1"
```

```
# setup initial portfolio tracking variables
initial_capital <- 500000
num_tickers <- length(sectors)*N_sector_best_stocks*2 # two sub-strategies for picking
initial_tickers <- rep(NA, num_tickers)
weights <- rep(1/num_tickers, num_tickers) # initialize to 1/n
returns <- rep(NA, N_runs)

# repack the portfolio
portfolio <- list(tickers = initial_tickers,
                  weights = weights,
                  capital = initial_capital,
                  returns = returns,
                  data = NA
                  )
portfolio
```

```
## $tickers
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA
##
## $weights
##  [1] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##  [7] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
```

```
## [13] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [19] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [25] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [31] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##
## $capital
## [1] 5e+05
##
## $returns
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA NA NA NA NA
##
## $data
## [1] NA
```

```r
# Initiate backtesting
print(paste(rep("-", 100), collapse = ""))
```

```
## [1] "----------------------------------------------------------------------------------------------------"
```

```r
print("BACKTESTING")
```

```
## [1] "BACKTESTING"
```

```r
print(paste(rep("-", 100), collapse = ""))
```

```
## [1] "----------------------------------------------------------------------------------------------------"
```

```r
print("")
```

```
## [1] ""
```

```r
# for every run (sliding window of time to consider)
for(tau in seq(N_runs)){
  # close any positions
  print("###############")
  print(paste0("### (tau=", tau, ") ###"))
  print("###############")
  print("CLOSE all positions")

  # Calculate and record profit-loss
  print("(1) COMPUTE_P/L(portfolio)")
  portfolio$capital <- portfolio$capital * (1 + runif(1, -0.05, 0.10))
  print(paste0("--> Capital:", portfolio$capital, "$"))

  # variables
  i_sector <- 1 # keep index counter for sectors
  num_top_pick <- N_sector_best_stocks*2 # number of stocks picked per sector

  # current portf
  cur_tickers <- rep(NA, num_tickers)

  print("")
  print("(2) PORTFOLIO_LOOP:")
  # loop through all the sectors
```

```r
  for(G in sectors){
    # execute sector procedure
    print(paste0("    SECTOR_PROCEDURE(G=", G, ", tau=",tau, ")"))

    # return top 3 best stocks according to procedure
    top_sector_stocks <- sample(names(sp500_stocks[[G]]), num_top_pick)

    # assign best stocks to portfolio (NEED TO UPDATE LOGIC!)
    i_replace <- rep(i_sector, num_top_pick) + seq(0, num_top_pick-1) # indexes to choose from
    cur_tickers[i_replace] <- top_sector_stocks
    i_sector <- i_sector + num_top_pick
  }

  # Assign tickers for this simulation
  portfolio$tickers <- as.vector(cur_tickers)

  # Display selected portfolio tickers
  print("Cur Portfolio:")
  print(portfolio$tickers)

  # Optimize portfolio weights using modified min_variance
  print("")
  print("(3) OPTIMIZE_PORTFOLIO(portfolio)")
  # simulate the optimization
  portfolio$weights <- runif(length(portfolio$weights)) / sum(runif(length(portfolio$weights)))
  print("weights: ")
  print(paste(" ", portfolio$weights))

  print("")
  print("(4) LONG PORTFOLIO()")

  # Separate similuation (over)
  print(paste(rep("-", 100), collapse = ""))

  # TEST: Just for this small printing simulation !!
  if(tau > 4){
    break
  }
}
```

```
## [1] "###############"
## [1] "### (tau=1) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:525137.338205241$"
## [1] ""
## [1] "(2) PORTFOLIO_LOOP:"
## [1] "    SECTOR_PROCEDURE(G=Industrials, tau=1)"
## [1] "    SECTOR_PROCEDURE(G=Health Care, tau=1)"
## [1] "    SECTOR_PROCEDURE(G=Information Technology, tau=1)"
## [1] "    SECTOR_PROCEDURE(G=Communication Services, tau=1)"
## [1] "    SECTOR_PROCEDURE(G=Financials, tau=1)"
## [1] "    SECTOR_PROCEDURE(G=Consumer Discretionary, tau=1)"
## [1] "Cur Portfolio:"
##  [1] "BA"   "ADP"  "CSX"  "CAT"  "DE"   "GE"   "BMY"  "AMGN" "SYK"  "MRK"
## [11] "ABT"  "TMO"  "IBM"  "CSCO" "TXN"  "AVGO" "MSFT" "ADBE" "T"    "GOOG"
## [21] "META" "TMUS" "DIS"  "WBD"  "V"    "C"    "PGR"  "SCHW" "MMC"  "WFC"
## [31] "AMZN" "NKE"  "ORLY" "AZO"  "GM"   "SBUX"
```

```
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0482059920169423"  "  0.0461733191894479"  "  0.0415262636038605"
##  [4] "  0.0469574536282151"  "  0.012358626075758"   "  0.0216578434150279"
##  [7] "  0.043467382720153"   "  0.0253376897137651"  "  0.00733708566052511"
## [10] "  0.0389343842804391"  "  0.0100751129346321"  "  0.0459712663724311"
## [13] "  0.0477910211393527"  "  0.0388396815948261"  "  0.0469993794062171"
## [16] "  0.00209130988887528" "  0.00702216778882097" "  0.00443268358548388"
## [19] "  0.0360357465347816"  "  0.0228416105607116"  "  0.0139297546037314"
## [22] "  0.0279660517034762"  "  0.0105887297817777"  "  0.0286996949895388"
## [25] "  0.0115009486236665"  "  0.0164309213570394"  "  0.0283939817516448"
## [28] "  0.0171187079398162"  "  0.00392448453569571" "  0.0110801931413277"
## [31] "  0.0293412459895565"  "  0.00176509279959754" "  0.00335882395391552"
## [34] "  0.0245502925292782"  "  0.00857850123456129" "  0.0205423198657427"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "-----------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=2) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:512720.931594704$"
## [1] ""
## [1] "(2) PORTFOLIO_LOOP:"
## [1] "    SECTOR_PROCEDURE(G=Industrials, tau=2)"
## [1] "    SECTOR_PROCEDURE(G=Health Care, tau=2)"
## [1] "    SECTOR_PROCEDURE(G=Information Technology, tau=2)"
## [1] "    SECTOR_PROCEDURE(G=Communication Services, tau=2)"
## [1] "    SECTOR_PROCEDURE(G=Financials, tau=2)"
## [1] "    SECTOR_PROCEDURE(G=Consumer Discretionary, tau=2)"
## [1] "Cur Portfolio:"
##  [1] "ADP"  "UNP"  "UPS"  "CSX"  "GE"   "BA"   "MDT"  "JNJ"  "ABT"  "LLY"
## [11] "SYK"  "ISRG" "AVGO" "AMD"  "ADBE" "TXN"  "IBM"  "CSCO" "ATVI" "TMUS"
## [21] "OMC"  "GOOG" "EA"   "DIS"  "MMC"  "SCHW" "C"    "MS"   "V"    "SPGI"
## [31] "SBUX" "CMG"  "GM"   "ORLY" "NKE"  "TSLA"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0016260471734588"   "  0.00842404418338589"  "  0.0164623256137902"
##  [4] "  0.00878220877427243"  "  0.00554774540474547"  "  0.0292967784651774"
##  [7] "  4.73613234525337e-05" "  0.0523978130284172"   "  0.0278459367163855"
## [10] "  0.0378759314520065"   "  0.0496941370797583"   "  0.0204544144773149"
## [13] "  0.0112983730965131"   "  0.0166600879590932"   "  0.0397079858259344"
## [16] "  0.0509211350837449"   "  0.0105207190193659"   "  0.013691101358429"
## [19] "  0.0390098342318434"   "  0.0167660590122796"   "  0.0193323088997988"
## [22] "  0.0331063853352877"   "  0.0281144269122132"   "  0.00846509791755194"
## [25] "  0.0307107083258924"   "  0.0239644020866693"   "  0.0404591303635017"
## [28] "  0.0116585525739931"   "  0.0101473210968308"   "  0.0361545928306766"
## [31] "  0.00798468120879843"  "  0.0336364634808896"   "  0.010772433131874"
## [34] "  0.0432271237765983"   "  0.0203956713236513"   "  0.0511798254245493"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "-----------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=3) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
```

```
## [1] "--> Capital:553145.558914472$"
## [1] ""
## [1] "(2) PORTFOLIO_LOOP:"
## [1] "    SECTOR_PROCEDURE(G=Industrials, tau=3)"
## [1] "    SECTOR_PROCEDURE(G=Health Care, tau=3)"
## [1] "    SECTOR_PROCEDURE(G=Information Technology, tau=3)"
## [1] "    SECTOR_PROCEDURE(G=Communication Services, tau=3)"
## [1] "    SECTOR_PROCEDURE(G=Financials, tau=3)"
## [1] "    SECTOR_PROCEDURE(G=Consumer Discretionary, tau=3)"
## [1] "Cur Portfolio:"
##  [1] "BA"    "HON"   "LMT"   "ETN"   "RTX"   "CAT"   "LLY"   "ELV"   "SYK"
## [10] "JNJ"   "TMO"   "UNH"   "INTC"  "AAPL"  "AMD"   "ACN"   "TXN"   "INTU"
## [19] "CHTR"  "TTWO"  "META"  "GOOGL" "GOOG"  "VZ"    "SCHW"  "GS"    "MS"
## [28] "PGR"   "BLK"   "SPGI"  "AMZN"  "TJX"   "ORLY"  "SBUX"  "F"     "ABNB"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0499788824675893"  "  0.0306702542874202"  "  0.0418700696890506"
##  [4] "  0.0532634120854659"  "  0.0413314392782679"  "  0.0319792904409967"
##  [7] "  0.0555177616764978"  "  0.012570180259613"   "  0.00419138190186126"
## [10] "  0.0290250986073827"  "  0.0372498828221388"  "  0.00334843629967522"
## [13] "  0.0315784187084417"  "  0.00886582960599666" "  0.0240426815354837"
## [16] "  0.0370553981807697"  "  0.0022831807564084"  "  0.0401357916117935"
## [19] "  0.0236158765552396"  "  0.0274155961314213"  "  0.0141243217820827"
## [22] "  0.0539952415827246"  "  0.0565660871518963"  "  0.0309263433450456"
## [25] "  0.00411320802009358" "  0.0209354007898268"  "  0.0288639334302285"
## [28] "  0.0180820219415754"  "  0.0524197645953656"  "  0.0427049206241226"
## [31] "  0.0463108268740697"  "  0.0485636198518906"  "  0.0401217293130865"
## [34] "  0.0524535454884725"  "  0.0450348451454891"  "  0.0234029376881964"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=4) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:586359.145113085$"
## [1] ""
## [1] "(2) PORTFOLIO_LOOP:"
## [1] "    SECTOR_PROCEDURE(G=Industrials, tau=4)"
## [1] "    SECTOR_PROCEDURE(G=Health Care, tau=4)"
## [1] "    SECTOR_PROCEDURE(G=Information Technology, tau=4)"
## [1] "    SECTOR_PROCEDURE(G=Communication Services, tau=4)"
## [1] "    SECTOR_PROCEDURE(G=Financials, tau=4)"
## [1] "    SECTOR_PROCEDURE(G=Consumer Discretionary, tau=4)"
## [1] "Cur Portfolio:"
##  [1] "FDX"  "GE"   "ITW"  "CAT"  "BA"   "ADP"  "JNJ"  "MRK"  "ELV"  "ABT"
## [11] "UNH"  "SYK"  "INTC" "TXN"  "QCOM" "ACN"  "INTU" "AMD"  "T"    "EA"
## [21] "TTWO" "DIS"  "GOOG" "ATVI" "SPGI" "BLK"  "GS"   "SCHW" "V"    "PGR"
## [31] "MAR"  "TJX"  "F"    "CMG"  "AMZN" "MCD"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0275875062742281"   "  0.00897149709102982"  "  0.0359926066418334"
##  [4] "  0.0335015123671435"   "  0.0484515068265136"   "  0.0448191027771505"
##  [7] "  0.0111762899535448"   "  0.0391334116029868"   "  0.00851912003986717"
## [10] "  0.00416272585877248"  "  0.000471743424228259" "  0.0313168325246908"
## [13] "  0.0389149749670736"   "  0.0459987737814194"   "  0.0111831066594336"
## [16] "  0.0140302573914645"   "  0.000569752167008196" "  0.0127050881825041"
```

```
## [19] "  0.0235508110200262"   "  0.0399630920255638"   "  0.00617763211253737"
## [22] "  0.0406899145346077"   "  0.00723830721706692"  "  0.0285833027944361"
## [25] "  0.0448992956448501"   "  0.0380317409520168"   "  0.0322539383681085"
## [28] "  0.0194853820472622"   "  3.89210910787683e-05" "  0.0199445289841706"
## [31] "  0.00609050825472995"  "  0.0084725605574342"   "  0.0254386153010785"
## [34] "  0.0179616494780707"   "  0.0337615132106422"   "  0.0371624790429045"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "---------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=5) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:619651.765771017$"
## [1] ""
## [1] "(2) PORTFOLIO_LOOP:"
## [1] "    SECTOR_PROCEDURE(G=Industrials, tau=5)"
## [1] "    SECTOR_PROCEDURE(G=Health Care, tau=5)"
## [1] "    SECTOR_PROCEDURE(G=Information Technology, tau=5)"
## [1] "    SECTOR_PROCEDURE(G=Communication Services, tau=5)"
## [1] "    SECTOR_PROCEDURE(G=Financials, tau=5)"
## [1] "    SECTOR_PROCEDURE(G=Consumer Discretionary, tau=5)"
## [1] "Cur Portfolio:"
##  [1] "ADP"  "DE"   "HON"  "UPS"  "UNP"  "CSX"  "PFE"  "SYK"  "MRK"  "DHR"
## [11] "ELV"  "BMY"  "INTU" "AMD"  "ADBE" "ORCL" "IBM"  "ACN"  "TTWO" "VZ"
## [21] "OMC"  "DIS"  "NFLX" "T"    "BAC"  "V"    "WFC"  "MMC"  "BLK"  "JPM"
## [31] "ABNB" "CMG"  "MAR"  "TSLA" "MCD"  "HD"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0492659214707945"  "  0.0481550826452409"  "  0.0502228249191179"
##  [4] "  0.00582265541047368" "  0.0292294995511896"  "  0.0188025413238795"
##  [7] "  0.0127900192302594"  "  0.0490384336398807"  "  0.0323007144313486"
## [10] "  0.00235507205380725" "  0.0100913138287059"  "  0.0433362433326903"
## [13] "  0.0474742827731393"  "  0.0415115643764983"  "  0.0353246788380345"
## [16] "  0.0158196248970174"  "  0.0287123895711805"  "  0.023714306915549"
## [19] "  0.0100360114751835"  "  0.0495898948393437"  "  0.031932864368585"
## [22] "  0.04721157304153"    "  0.0290845824395578"  "  0.0348111565301207"
## [25] "  0.02757399698596"    "  0.0294272914802595"  "  0.0488106066024074"
## [28] "  0.00461026789952682" "  0.0181147017603152"  "  0.0386119480047509"
## [31] "  0.0529118169625229"  "  0.0474799443624355"  "  0.032240606171986"
## [34] "  0.0374552207512913"  "  0.0359835990462247"  "  0.0283173372624626"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "---------------------------------------------------------------------------------------------"
```

SECTOR_PROCEDURE

1. Sector G contains tickers $\{S_1, S_1, \ldots, S_{|G|}\}$, where $|G|$= number of stocks per sector (before selection).
2. For each ticker, want to calculate **current window:**

$$[t_1 = \text{week } W_{s\times\tau}\ ,\ t_{12} = \text{week } W_{s\times\tau+11}]$$

e.g. with $s = 1$ (slide one month at the time)

$$
\begin{cases}
\tau = 1 \implies [t_1 = W_1 \ , \ t_{12} = W_{12}] \\
\tau = 2 \implies [t_1 = W_2 \ , \ t_{12} = W_{13}] \\
\vdots \\
\tau = i \implies [t_1 = W_i \ , \ t_{12} = W_{i+11}] \\
\vdots \\
\tau = T \implies [t_1 = W_{T-12} \ , \ t_{12} = W_T]
\end{cases}
$$

EXTRACT_STATIC_FEATURES()

We had a set of features for some stock:

```r
# sample stock dataframe
sample_xts <- sp500_stocks$Industrials$ADP
head(sample_xts, 5)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-01-06             -1      -0.04944265             NA            NA
## 2016-01-13              1       0.01131413    -0.04944265            NA
## 2016-01-20              1       0.02848332     0.01131413   -0.04944265
## 2016-01-27              1       0.02053834     0.02848332    0.01131413
## 2016-02-03             -1      -0.01619911     0.02053834    0.02848332
##            adjclose_lag2 adjclose_lag3 atr adx aaron bb chaikin_vol clv emv
## 2016-01-06            NA            NA  NA  NA    NA NA          NA  NA  NA
## 2016-01-13            NA            NA  NA  NA   -50 NA          NA  NA  NA
## 2016-01-20            NA            NA  NA  NA  -100 NA          NA  NA  NA
## 2016-01-27   -0.04944265            NA  NA  NA    50 NA          NA  NA  NA
## 2016-02-03    0.01131413   -0.04944265  NA  NA   100 NA          NA  NA  NA
##            macd mfi      sar smi volat month_index
## 2016-01-06   NA  NA 79.55761  NA    NA           1
## 2016-01-13   NA  NA 81.71000  NA    NA           1
## 2016-01-20   NA  NA 81.71000  NA    NA           1
## 2016-01-27   NA  NA 77.34000  NA    NA           1
## 2016-02-03   NA  NA 77.34000  NA    NA           2
```

```r
# source the feature engineering file
library("here")
source(here("functions", "feature_engineering.R"))

# test out for a sample run
tau = 3 # suppose we're at run number 3
sample_xts_train_val <- f_extract_train_val_features(sample_xts, # stock xts
                                                     tau=tau, # current run
                                                     n_months = N_window, # size of window
                                                     val_lag = 1 # validation month
                                                     )

# display some columns for the extracted data
head(sample_xts_train_val$train[,c("direction_lead", "clv", "volat", "month_index")])
```

```
##            direction_lead          clv     volat month_index
## 2016-03-02              1           NA        NA           3
## 2016-03-09              1  0.075378023 0.2380100           3
## 2016-03-16              1  0.175116926 0.2389290           3
## 2016-03-23              1  0.162085438 0.2214060           3
## 2016-03-30              1 -0.003746352 0.1992566           3
## 2016-04-06             -1  0.156024412 0.1872713           4
```

```r
print("")
```

```
## [1] ""
```

```r
head(sample_xts_train_val$val[,c("direction_lead", "clv", "volat", "month_index")])
```

```
##            direction_lead         clv     volat month_index
## 2018-02-07             -1 -0.02045124 0.2037605          26
## 2018-02-14              1  0.14581944 0.2180265          26
## 2018-02-21             -1  0.02476083 0.2316219          26
## 2018-02-28             -1 -0.15801223 0.2332037          26
```

EXTRACT_DYNAMIC_FEATURES

```r
# add GARCH features only
sample_xts_with_garch <- f_add_garch_forecast(sample_xts, volat_col="volat")

# display
tail(sample_xts_with_garch, 3)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2022-11-16              1      0.034718700    0.053616090    0.01230604
## 2022-11-23              1      0.005923517    0.034718700    0.05361609
## 2022-11-30             NA               NA    0.005923517    0.03471870
##            adjclose_lag2 adjclose_lag3      atr      adx aaron        bb
## 2022-11-16   0.009733913   0.008113008 10.23247 14.68326   100 0.8325740
## 2022-11-23   0.012306040   0.009733913 10.24301 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012306040 10.24779 16.53998   100 0.8907336
##            chaikin_vol        clv       emv     macd      mfi    sar       smi
## 2022-11-16  -0.3839710 -0.4461119 0.0907485 1.906715 48.83463 256.72  6.291102
## 2022-11-23  -0.2018052 -0.3205142 0.1175853 2.068291 49.31528 224.11 11.099826
## 2022-11-30   0.4839489 -0.1089895 0.1214467 2.300754 42.97382 224.11 16.713518
##                volat month_index vol_forecast
## 2022-11-16 0.2641173          83    0.2642679
## 2022-11-23 0.2624611          83    0.2651389
## 2022-11-30 0.2759187          83    0.2659892
```

```r
# Example usage
sample_xts_with_arima <- f_add_arima_forecast(sample_xts_with_garch,
                                               return_col="realized_returns")
tail(sample_xts_with_arima)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2022-10-26              1      0.009733913    0.008113008   0.039930970
## 2022-11-02              1      0.012306040    0.009733913   0.008113008
## 2022-11-09              1      0.053616090    0.012306040   0.009733913
## 2022-11-16              1      0.034718700    0.053616090   0.012306040
## 2022-11-23              1      0.005923517    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923517   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535730   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535730  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113008   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733913   0.008113008 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012306040   0.009733913 10.243009 15.95273   100 0.9310325
```

```
## 2022-11-30   0.053616090   0.012306040 10.247795 16.53998   100 0.8907336
##             chaikin_vol        clv          emv      macd      mfi      sar
## 2022-10-26 -1.49750300 -0.1320576 -0.01707202 2.049576 51.52422 260.0428
## 2022-11-02  2.90314600 -0.2863719  0.02711271 1.939312 49.23300 258.6055
## 2022-11-09 -0.09676625 -0.3920529  0.04765004 1.866926 49.20839 257.2257
## 2022-11-16 -0.38397100 -0.4461119  0.09074850 1.906715 48.83463 256.7200
## 2022-11-23 -0.20180520 -0.3205142  0.11758529 2.068291 49.31528 224.1100
## 2022-11-30  0.48394890 -0.1089895  0.12144667 2.300754 42.97382 224.1100
##                   smi     volat month_index vol_forecast arima_100_001
## 2022-10-26  8.131402 0.2269538          82    0.2624611   0.005473012
## 2022-11-02  5.546375 0.2606250          83    0.2759187   0.003833981
## 2022-11-09  3.943960 0.2653165          83    0.2633755   0.003715044
## 2022-11-16  6.291102 0.2641173          83    0.2642679   0.003708274
## 2022-11-23 11.099826 0.2624611          83    0.2651389   0.003707888
## 2022-11-30 16.713518 0.2759187          83    0.2659892   0.003707866
##            arima_010_001 arima_110_001 arima_020_001 arima_120_001
## 2022-10-26   0.034718700    0.04342609    0.01582131    0.05513172
## 2022-11-02   0.005923517    0.01919154   -0.02287167   -0.01640924
## 2022-11-09   0.005923517    0.01307800   -0.05166685   -0.04296142
## 2022-11-16   0.005923517    0.01589495   -0.08046203   -0.06675866
## 2022-11-23   0.005923517    0.01459698   -0.10925721   -0.09235465
## 2022-11-30   0.005923517    0.01519505   -0.13805240   -0.11677621
##            arima_100_011 arima_010_011 arima_110_011 arima_020_011
## 2022-10-26   0.005473012   0.034718700    0.04342609    0.01582131
## 2022-11-02   0.003833981   0.005923517    0.01919154   -0.02287167
## 2022-11-09   0.003715044   0.005923517    0.01307800   -0.05166685
## 2022-11-16   0.003708274   0.005923517    0.01589495   -0.08046203
## 2022-11-23   0.003707888   0.005923517    0.01459698   -0.10925721
## 2022-11-30   0.003707866   0.005923517    0.01519505   -0.13805240
##            arima_120_011
## 2022-10-26    0.05513172
## 2022-11-02   -0.01640924
## 2022-11-09   -0.04296142
## 2022-11-16   -0.06675866
## 2022-11-23   -0.09235465
## 2022-11-30   -0.11677621
```

```r
sample_xts_with_arima[, c("actual_returns", "vol_forecast")]
```

```
##            actual_returns vol_forecast
## 2016-01-06             NA           NA
## 2016-01-13  -0.0494426500           NA
## 2016-01-20   0.0113141300           NA
## 2016-01-27   0.0284833200           NA
## 2016-02-03   0.0205383400           NA
## 2016-02-10  -0.0161991100    0.2380100
## 2016-02-17   0.0541783600    0.2389290
## 2016-02-24  -0.0008205272    0.2214060
## 2016-03-02   0.0045634540    0.1992566
## 2016-03-09   0.0070357570    0.1872713
##        ...
## 2022-09-28   0.0066180690    0.2269538
## 2022-10-05   0.0301509800    0.2606250
## 2022-10-12  -0.0645357300    0.2653165
## 2022-10-19   0.0399309700    0.2641173
## 2022-10-26   0.0081130080    0.2624611
## 2022-11-02   0.0097339130    0.2759187
## 2022-11-09   0.0123060400    0.2633755
## 2022-11-16   0.0536160900    0.2642679
```

```
## 2022-11-23   0.0347187000    0.2651389
## 2022-11-30   0.0059235170    0.2659892
```

```r
# Example usage
sample_xts_full <- f_extract_dynamic_features(sample_xts_with_garch,
                                              return_col="realized_returns")
tail(sample_xts_full)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2022-10-26              1      0.009733913    0.008113008   0.039930970
## 2022-11-02              1      0.012306040    0.009733913   0.008113008
## 2022-11-09              1      0.053616090    0.012306040   0.009733913
## 2022-11-16              1      0.034718700    0.053616090   0.012306040
## 2022-11-23              1      0.005923517    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923517   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535730   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535730  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113008   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733913   0.008113008 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012306040   0.009733913 10.243009 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012306040 10.247795 16.53998   100 0.8907336
##            chaikin_vol        clv         emv     macd      mfi      sar
## 2022-10-26 -1.49750300 -0.1320576 -0.01707202 2.049576 51.52422 260.0428
## 2022-11-02  2.90314600 -0.2863719  0.02711271 1.939312 49.23300 258.6055
## 2022-11-09 -0.09676625 -0.3920529  0.04765004 1.866926 49.20839 257.2257
## 2022-11-16 -0.38397100 -0.4461119  0.09074850 1.906715 48.83463 256.7200
## 2022-11-23 -0.20180520 -0.3205142  0.11758529 2.068291 49.31528 224.1100
## 2022-11-30  0.48394890 -0.1089895  0.12144667 2.300754 42.97382 224.1100
##                  smi     volat month_index vol_forecast arima_100_001
## 2022-10-26  8.131402 0.2269538          82    0.2624611   0.005473012
## 2022-11-02  5.546375 0.2606250          83    0.2759187   0.003833981
## 2022-11-09  3.943960 0.2653165          83    0.2633755   0.003715044
## 2022-11-16  6.291102 0.2641173          83    0.2642679   0.003708274
## 2022-11-23 11.099826 0.2624611          83    0.2651389   0.003707888
## 2022-11-30 16.713518 0.2759187          83    0.2659892   0.003707866
##            arima_010_001 arima_110_001 arima_020_001 arima_120_001
## 2022-10-26   0.034718700    0.04342609    0.01582131    0.05513172
## 2022-11-02   0.005923517    0.01919154   -0.02287167   -0.01640924
## 2022-11-09   0.005923517    0.01307800   -0.05166685   -0.04296142
## 2022-11-16   0.005923517    0.01589495   -0.08046203   -0.06675866
## 2022-11-23   0.005923517    0.01459698   -0.10925721   -0.09235465
## 2022-11-30   0.005923517    0.01519505   -0.13805240   -0.11677621
##            arima_100_011 arima_010_011 arima_110_011 arima_020_011
## 2022-10-26   0.005473012   0.034718700    0.04342609    0.01582131
## 2022-11-02   0.003833981   0.005923517    0.01919154   -0.02287167
## 2022-11-09   0.003715044   0.005923517    0.01307800   -0.05166685
## 2022-11-16   0.003708274   0.005923517    0.01589495   -0.08046203
## 2022-11-23   0.003707888   0.005923517    0.01459698   -0.10925721
## 2022-11-30   0.003707866   0.005923517    0.01519505   -0.13805240
##            arima_120_011
## 2022-10-26    0.05513172
## 2022-11-02   -0.01640924
## 2022-11-09   -0.04296142
## 2022-11-16   -0.06675866
## 2022-11-23   -0.09235465
## 2022-11-30   -0.11677621
```

SECTOR PROCEDURE

```
SECTOR_PROCEDURE <- function(G, tau){
  ##
  ## Params:
  ##  - G (str): Economic sector name; will be used to fetch the  List of lists
  ## which are the pre-selected stocks for that sector.
  ##  - tau (numeric): Integer that corresponds to the actual run of the backtest.
  ##


  ### TEST ###
  # NOTE: For testing only, will be removed later!
  num_top_pick <- N_sector_best_stocks*2 # number of stocks picked per sector
  ### TEST ###

  print(paste0("SECTOR_PROCEDURE(G=", G, ", tau=",tau, ")"))

  # retrieve sector data
  sector_data <- sp500_stocks[[G]]

  # stocks for sector provided
  sector_tickers <- names(sector_data)

  # to store subset features for window
  sector_stocks_window <- rep(NA, length(sector_tickers))
  names(sector_stocks_window) <- sector_tickers

  # extract static train-val for all stocks
  list_train_val_sector <- lapply(sector_data,
                                  f_extract_train_val_features,
                                  tau=tau, # current run
                                  n_months = 12, # size of window
                                  val_lag = 1 # months to use in val set
                                  )

  # return top 3 best stocks according to modelling procedure
  print("  MODELLING_PROCEDURE(list_train_val_sector)")
  top_sector_stocks <- sample(names(sp500_stocks[[G]]), num_top_pick)

  ########## Inside MODELLING_PROCEDURE ########################
  ### NOTE: The MODELLING_PROCEDURE internally will use the train and

  # Stack the train and val splitted data for all stocks in sector
  sector_stocks <- lapply(list_train_val_sector, function(stock) {
    # Concatenate 'train' and 'val' xts objects within each stock
    concatenated_xts <- rbind(stock$train, stock$val)
    return(concatenated_xts)
  })

  # NOTE: MODELLLING_PROCEDURE should also compute dynamic features for concatenated data
  sector_stocks <- lapply(sector_stocks, f_extract_dynamic_features)

  # should return the train-val list for the chosen stocks
  chosen_stocks <- sector_stocks[names(sector_stocks) %in% top_sector_stocks]

  ########## Inside MODELLING_PROCEDURE ########################

  return(chosen_stocks) # not actual return value!
```

```
}

# peform the sector procedure
G = names(sp500_stocks)[[1]]
tau = 5
sector_stocks_window <- SECTOR_PROCEDURE(G, tau)
```

```
## [1] "SECTOR_PROCEDURE(G=Industrials, tau=5)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
```

```
names(sector_stocks_window) # names are tickers, values are list of train-val xts
```

```
## [1] "ADP" "DE"  "ETN" "ITW" "LMT" "UPS"
```

```
head(sector_stocks_window[[2]]) # show ticker xts
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-05-04              1       0.02071379    -0.03032161   0.005325853
## 2016-05-11             -1      -0.01561578     0.02071379  -0.030321608
## 2016-05-18             -1      -0.02587708    -0.01561578   0.020713791
## 2016-05-25              1       0.02865776    -0.02587708  -0.015615779
## 2016-06-01              1       0.05118445     0.02865776  -0.025877079
## 2016-06-08             -1      -0.02732549     0.05118445   0.028657761
##            adjclose_lag2 adjclose_lag3      atr adx aaron        bb chaikin_vol
## 2016-05-04   0.071318624   0.027391959 3.216998  NA    50        NA          NA
## 2016-05-11   0.005325853   0.071318624 3.185069  NA   -50        NA          NA
## 2016-05-18  -0.030321608   0.005325853 3.098993  NA    50 0.7162007          NA
## 2016-05-25   0.020713791  -0.030321608 3.066922  NA  -100 0.5630780   0.0516258
## 2016-06-01  -0.015615779   0.020713791 3.027857  NA   -50 0.6751524  -0.2165642
## 2016-06-08  -0.025877079  -0.015615779 3.195152  NA   100 0.9914412  -0.5258239
##                   clv           emv macd      mfi      sar smi     volat
## 2016-05-04 0.13622365  1.360177e-03   NA 71.10988 78.19043  NA 0.2203978
## 2016-05-11 0.09544153  3.719926e-05   NA 71.69679 78.89539  NA 0.2245193
## 2016-05-18 0.10406222  8.688395e-04   NA 65.72483 79.52985  NA 0.2164100
## 2016-05-25 0.11446762 -1.852668e-04   NA 56.91212 80.10086  NA 0.2142833
## 2016-06-01 0.23604308  1.481416e-03   NA 63.47143 80.49000  NA 0.2181418
## 2016-06-08 0.16656899  6.728191e-03   NA 64.56882 80.96500  NA 0.2259061
##            month_index arima_100_001 arima_010_001 arima_110_001 arima_020_001
## 2016-05-04           5   0.001564469    0.02865776  -0.003290775   0.083192600
## 2016-05-11           5  -0.003171989    0.05118445   0.037987478   0.073711137
## 2016-05-18           5   0.013335500   -0.02732549   0.018668549  -0.105835435
## 2016-05-25           5   0.009857793   -0.01078545  -0.020475222   0.005754593
## 2016-06-01           6   0.012145133   -0.02166408  -0.015290976  -0.032542717
## 2016-06-08           6   0.012919387   -0.02534645  -0.023189185  -0.029028825
##            arima_120_001 arima_100_011 arima_010_011 arima_110_011
## 2016-05-04    0.03603088   0.001564469    0.02865776  -0.003290775
## 2016-05-11    0.09700819  -0.003171989    0.05118445   0.037987478
## 2016-05-18   -0.03229616   0.013335500   -0.02732549   0.018668549
## 2016-05-25   -0.06342732   0.009857793   -0.01078545  -0.020475222
## 2016-06-01   -0.01258610   0.012145133   -0.02166408  -0.015290976
## 2016-06-08   -0.03426661   0.012919387   -0.02534645  -0.023189185
##            arima_020_011 arima_120_011 vol_forecast
## 2016-05-04   0.083192600    0.03603088    0.2181418
## 2016-05-11   0.073711137    0.09700819    0.2259061
## 2016-05-18  -0.105835435   -0.03229616    0.2217548
## 2016-05-25   0.005754593   -0.06342732    0.2113276
## 2016-06-01  -0.032542717   -0.01258610    0.1902579
## 2016-06-08  -0.029028825   -0.03426661    0.1858925
```

MODELLING_PROCEDURE

```r
# parameters
G <- names(sp500_stocks)[1] # sample sector
tau <- 1 # suppose we are in run 5 of the backtest

####### Inside SECTOR_PROCEDURE ########

# retrieve sector data
sector_data <- sp500_stocks[[G]]

# stocks for sector provided
sector_tickers <- names(sector_data)

# to store subset features for window
sector_stocks_window <- rep(NA, length(sector_tickers))
names(sector_stocks_window) <- sector_tickers

# extract static train-val for all stocks
list_train_val_sector <- lapply(sector_data,
                                f_extract_train_val_features,
                                tau=tau, # current run
                                n_months = N_window, # size of window
                                val_lag = 1 # months to use in val set
                                )

####### Inside SECTOR_PROCEDURE ########

# keys are stock tickers for that sector
names(list_train_val_sector)
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```r
# each stock has train and test
names(list_train_val_sector[[1]])
```

```
## [1] "train" "val"
```

```r
# Check some of train and val data for one stock
head(list_train_val_sector[[1]]$train, 3)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-01-06             -1      -0.04944265             NA            NA
## 2016-01-13              1       0.01131413    -0.04944265            NA
## 2016-01-20              1       0.02848332     0.01131413   -0.04944265
##            adjclose_lag2 adjclose_lag3 atr adx aaron bb chaikin_vol clv emv
## 2016-01-06            NA            NA  NA  NA    NA NA          NA  NA  NA
## 2016-01-13            NA            NA  NA  NA   -50 NA          NA  NA  NA
## 2016-01-20            NA            NA  NA  NA  -100 NA          NA  NA  NA
##            macd mfi      sar smi volat month_index
## 2016-01-06   NA  NA 79.55761  NA    NA           1
## 2016-01-13   NA  NA 81.71000  NA    NA           1
## 2016-01-20   NA  NA 81.71000  NA    NA           1
```

```r
print("")
```

```
## [1] ""
```

```r
tail(list_train_val_sector[[1]]$val, 3)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2017-12-13              1     7.152516e-03    0.014894270   0.022596070
## 2017-12-20             -1    -5.103599e-03    0.007152516   0.014894270
## 2017-12-27             -1    -8.541914e-05   -0.005103599   0.007152516
##            adjclose_lag2 adjclose_lag3      atr      adx aaron        bb
## 2017-12-13    0.02791647  -0.004074823 2.772381 18.58305   100 0.8875978
## 2017-12-20    0.02259607   0.027916470 2.709354 19.53785   100 0.9247934
## 2017-12-27    0.01489427   0.022596070 2.583686 20.14515    50 0.8325108
##            chaikin_vol        clv         emv     macd      mfi      sar
## 2017-12-13 -1.58453000 -0.1773531 0.003475763 2.612841 59.76057 110.2100
## 2017-12-20  1.99874900 -0.2788403 0.004734173 2.662641 64.80303 110.5068
## 2017-12-27 -0.07351845 -0.2669577 0.001732219 2.709642 68.29887 111.0110
##                 smi     volat month_index
## 2017-12-13 35.44917 0.1594832          24
## 2017-12-20 38.35108 0.1537416          24
## 2017-12-27 40.23123 0.1529238          24
```

```r
print("")
```

```
## [1] ""
```

```r
nrow(list_train_val_sector[[1]]$train)
```

```
## [1] 100
```

```r
nrow(list_train_val_sector[[1]]$val)
```

```
## [1] 4
```

We have 46 observations (weeks) for train, and 4 (weeks) for val.

```r
print(head(list_train_val_sector[[1]]$train$month_index, 1)) # beginning month of window
```

```
##            month_index
## 2016-01-06           1
```

```r
print(tail(list_train_val_sector[[1]]$val$month_index, 1)) # end month of window
```

```
##            month_index
## 2017-12-27          24
```

```r
length(seq(5, 16)) # 12 months
```

```
## [1] 12
```

Feature Selection

Only on the `train_set`.

```r
# Load the package
source(here("functions", "feature_engineering.R"))

# Define the formula for regression
fmla <- realized_returns ~ . -realized_returns -month_index

# try obtaining best features for a sample train set for a stock in the sample sector
best_feat_list <- f_select_features(
                  fmla = fmla, # formula for regression
                  data = list_train_val_sector[[1]]$train, # train data for one stock of current sector
                  target_var = "realized_returns", # y
                  nvmax = 50,
                  method="forward")
```

```
## Loading required package: leaps
```

```r
best_feat_list
```

```
## $featnames
##  [1] "direction_lead" "actual_returns" "adjclose_lag2"  "adjclose_lag3"
##  [5] "atr"            "adx"            "aaron"          "clv"
##  [9] "macd"           "mfi"            "smi"            "volat"
##
## $fmla
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     adjclose_lag3 + atr + adx + aaron + clv + macd + mfi + smi +
##     volat
## <environment: 0x000001701aaa5aa8>
```

Regularized MLR (Elasticnet)

$$\mathcal{L}(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\left[\alpha||\beta||_1 + (1-\alpha)||\beta||_2^2\right]$$

```r
### Perform feature selection on the train set for every stock

# load required libraries
library("caret")
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library("Metrics")
```

```
##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
# Define the formula for regression
fmla <- realized_returns ~ . -realized_returns -month_index

# Create a grid for elastic net regression hyperparameters
grid_enet <- expand.grid(alpha = seq(from = 0, to = 1, by = 0.1),  # Elastic net mixing parameter
                         lambda = seq(from = 0, to = 0.05, by = 0.001))  # Regularization strength

# Initialize variable to save forecasted returns, MSEs and Sharpe Ratios
sector_tracker <- as.list(rep(NA, length(sector_tickers)))
names(sector_tracker) <- sector_tickers

# transform into a list of lists
sector_tracker <- lapply(sector_tracker, function(x) list(
  forecasted_ret = NA,
  sharpe = NA,
  rmse = NA,
  data = NA
))

# display values
fmla # all initial variables
```

```
## realized_returns ~ . - realized_returns - month_index
```

```
names(sector_tracker) # list of lists
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```
names(sector_tracker[[1]]) # to store the values as the loop happens
```

```
## [1] "forecasted_ret" "sharpe"         "rmse"           "data"
```

```
# Loop for every stock ticker in sector G
for(ticker in sector_tickers){
  print(paste0("ticker: ", ticker))

  # fetch data for that ticker
  ticker_data_train <- list_train_val_sector[[ticker]]$train
  ticker_data_val <- list_train_val_sector[[ticker]]$val

  # remove nas
  ticker_data_train <- na.omit(ticker_data_train) # data cannot contain nas
  ticker_data_val <- na.omit(ticker_data_val) # data cannot contain nas

  ### Step 1: Feature Selection

  # Perform feature selection for that stock
  best_feat_list <-f_select_features(
                      fmla = fmla, # formula for regression
                      data = ticker_data_train, # train data for one stock of current sector
                      target_var = "realized_returns", # y
                      nvmax = 50,
                      method="forward")
```

```
  print(c(best_feat_list$fmla))

  ### Step 2: Elasticnet

  # Set up time-slice cross-validation parameters
  ctr_train <- trainControl(method = "timeslice",
                            initialWindow = 52,  # Consecutive number of weeks ~= 6 months
                            horizon = 4,         # Horizon is one month prediction (4 weeks)
                            skip = 1,            # No skip, our data will overlap in practice
                            fixedWindow = TRUE,   # Use a fixed window
                            allowParallel = TRUE) # Enable parallel processing

  # Stack together train and val, since enet will cross-validate inside
  full_train <- rbind.xts(ticker_data_train, ticker_data_val)

  # Train the elastic net regression model using time-slice cross-validation
  model_enet_best <- train(form = best_feat_list$fmla,           # Formula from feature selection
                           data = ticker_data_train,             # Training data
                           method = "glmnet",                    # Model method
                           tuneGrid = grid_enet,                 # Hyperparameter grid
                           trControl = ctr_train,                # Cross-validation control
                           preProc = c("center", "scale"),       # Preprocessing steps
                           metric = "Rsquared",                  # Metric for selecting the best model
                           threshold = 0.2)

  # Extract the best alpha and beta fitted
  best_alpha <- model_enet_best$bestTune$alpha
  best_lambda <- model_enet_best$bestTune$lambda

  # Use the best-fitted elastic net regression model to make predictions on the val_data
  pred_enet_best <- predict(model_enet_best, ticker_data_val) # predict on val
  pred_enet_best <- mean(pred_enet_best) # take the average
  sector_tracker[[ticker]]$forecasted_ret <- pred_enet_best # save in tracker

  # Compute the RMSE on the validation set
  enet_rmse <- sqrt(mse(actual =ticker_data_val[, "realized_returns"], predicted = pred_enet_best))

  print("")
  print(paste("predicted return: ", pred_enet_best))
  print(paste("rmse: ", enet_rmse))


  print("############################################")
}
```

```
## [1] "ticker: ADP"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     adjclose_lag3 + atr + adx + aaron + clv + macd + mfi + smi +
##     volat
## <environment: 0x0000017023a6f230>
##
## [1] ""
## [1] "predicted return:  0.00425287070597015"
## [1] "rmse:  0.00755002658790937"
## [1] "############################################"
## [1] "ticker: BA"
## [[1]]
## realized_returns ~ direction_lead + clv + mfi + volat
```

```
## <environment: 0x000001702350fb30>
##
## [1] ""
## [1] "predicted return:  0.0115106564731343"
## [1] "rmse:  0.0210409061533028"
## [1] "###########################################"
## [1] "ticker: CAT"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag2 + adx + bb +
##     chaikin_vol + clv + mfi + sar
## <environment: 0x0000017020162bd0>
##
## [1] ""
## [1] "predicted return:  0.00838633425820895"
## [1] "rmse:  0.0289348992716546"
## [1] "###########################################"
## [1] "ticker: CSX"
## [[1]]
## realized_returns ~ direction_lead + adx + emv + macd + mfi +
##     sar + smi + volat
## <environment: 0x000001701e910d70>
##
## [1] ""
## [1] "predicted return:  0.0103945755477612"
## [1] "rmse:  0.0317795409205346"
## [1] "###########################################"
## [1] "ticker: DE"
## [[1]]
## realized_returns ~ direction_lead + atr + adx + clv + emv + sar +
##     volat
## <environment: 0x0000017017a3dfd0>
##
## [1] ""
## [1] "predicted return:  0.0339572252678987"
## [1] "rmse:  0.0195886327607426"
## [1] "###########################################"
## [1] "ticker: ETN"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag2 + adx + aaron +
##     bb + emv + macd + sar + volat
## <environment: 0x000001701c949050>
##
## [1] ""
## [1] "predicted return:  -0.000654111942170204"
## [1] "rmse:  0.0295652977845325"
## [1] "###########################################"
## [1] "ticker: FDX"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     adjclose_lag3 + adx + aaron + chaikin_vol + clv + sar
## <environment: 0x0000017020a8a590>
##
## [1] ""
## [1] "predicted return:  0.00537071614077343"
## [1] "rmse:  0.0266614966303764"
## [1] "###########################################"
## [1] "ticker: GE"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adx + bb + clv + macd + mfi + sar + smi + volat
```

```
## <environment: 0x0000017024222898>
##
## [1] ""
## [1] "predicted return:  -0.0178832912566443"
## [1] "rmse:  0.0344007333458254"
## [1] "##########################################"
## [1] "ticker: HON"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     aaron + bb + clv + emv + macd + sar + smi
## <environment: 0x000001701f16b660>
##
## [1] ""
## [1] "predicted return:  0.00450564574119403"
## [1] "rmse:  0.0123824899060627"
## [1] "##########################################"
## [1] "ticker: ITW"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag1 + adjclose_lag2 +
##     atr + aaron + bb + macd + mfi + volat
## <environment: 0x000001701cbc1b08>
##
## [1] ""
## [1] "predicted return:  -0.0156324603227544"
## [1] "rmse:  0.0233067096088579"
## [1] "##########################################"
## [1] "ticker: LMT"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag2 + chaikin_vol +
##     emv + macd + smi
## <environment: 0x0000017020f3e450>
##
## [1] ""
## [1] "predicted return:  0.012815531059591"
## [1] "rmse:  0.00880642120449787"
## [1] "##########################################"
## [1] "ticker: NOC"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adx + aaron + chaikin_vol + clv + emv + macd + smi + volat
## <environment: 0x0000017023ee5b48>
##
## [1] ""
## [1] "predicted return:  -0.00108130361658448"
## [1] "rmse:  0.0143875548124027"
## [1] "##########################################"
## [1] "ticker: RTX"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag3 + atr + adx +
##     chaikin_vol + clv + emv + mfi + sar + smi + volat
## <environment: 0x000001701dc504b0>
##
## [1] ""
## [1] "predicted return:  0.00293808272643091"
## [1] "rmse:  0.0166283725417561"
## [1] "##########################################"
## [1] "ticker: UNP"
## [[1]]
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + bb + clv + emv + macd + mfi +
```

```
##     volat
## <environment: 0x0000017022f26f98>
##
## [1] ""
## [1] "predicted return:  0.00503150334577797"
## [1] "rmse:  0.0159573164698068"
## [1] "###########################################"
## [1] "ticker: UPS"
## [[1]]
## realized_returns ~ direction_lead + adjclose_lag3 + atr + adx +
##     bb + clv + emv + macd
## <environment: 0x000001701c8730c0>
##
## [1] ""
## [1] "predicted return:  0.00864228673982985"
## [1] "rmse:  0.0262707919714151"
## [1] "###########################################"
```

Aside: Format for Portfolio Optimization

```
## This chunk of code simply obtains some portfolio stock tickers
## in a way that will be similar to the final result

# repack the portfolio (repeated from before)
portfolio <- list(tickers = initial_tickers,
                  weights = weights,
                  capital = initial_capital,
                  returns = returns,
                  data = NA
                  )
portfolio
```

```
## $tickers
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA
##
## $weights
##  [1] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##  [7] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [13] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [19] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [25] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [31] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##
## $capital
## [1] 5e+05
##
## $returns
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA NA NA NA NA
##
## $data
## [1] NA
```

The following simulates best tickers that would be obtained after modelling procedure for all sectors

```r
# Set up backtesting simulation parameters
sample_xts <- sp500_stocks$Industrials$ADP
sectors <- names(sp500_stocks)
N_sector_best_stocks <- 3
tau <- 3

# store ticker for current portfolio
cur_tickers <- rep(NA, num_tickers)

# store actual data for each run
portf_stocks_data <- as.list(rep(NA, length(sectors)))
names(portf_stocks_data) <- sectors

# keep index counter for sectors
i_sector <- 1

print("")
```

```
## [1] ""
```

```r
print("(2) PORTFOLIO_LOOP:")
```

```
## [1] "(2) PORTFOLIO_LOOP:"
```

```r
# loop through all the sectors
for(G in sectors){

  # return top 3 best stocks (xts data) according to procedure
  top_sector_stocks <- SECTOR_PROCEDURE(G, tau)

  # assign best stocks to portfolio (NEED TO UPDATE LOGIC!)
  i_replace <- rep(i_sector, num_top_pick) + seq(0, num_top_pick-1) # indexes to choose from
  cur_tickers[i_replace] <- names(top_sector_stocks)
  i_sector <- i_sector + num_top_pick

  # assign the data to the portfolio
  portf_stocks_data[[G]] <- top_sector_stocks
}
```

```
## [1] "SECTOR_PROCEDURE(G=Industrials, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
## [1] "SECTOR_PROCEDURE(G=Health Care, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
## [1] "SECTOR_PROCEDURE(G=Information Technology, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
## [1] "SECTOR_PROCEDURE(G=Communication Services, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
## [1] "SECTOR_PROCEDURE(G=Financials, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
## [1] "SECTOR_PROCEDURE(G=Consumer Discretionary, tau=3)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
```

```r
# Portfolio tickers get updated
portfolio$tickers <- cur_tickers
```

```r
# unlist data best stocks data format into a singles list
portf_data <- f_unlist_portf_data(portf_stocks_data)

# assign list to portfolio
portfolio$data <- portf_data
```

Data format for portfoli optimization

Note that at this point, the portfolio will have the tickers and the weights attributes.

```r
# Checko out the resulting portfolio
portfolio$tickers
```

```
##  [1] "ADP"  "BA"   "ETN"  "FDX"  "GE"   "UNP"  "ELV"  "ISRG" "JNJ"  "LLY"
## [11] "PFE"  "UNH"  "AVGO" "CSCO" "INTU" "MSFT" "QCOM" "TXN"  "EA"   "GOOG"
## [21] "NFLX" "TTWO" "VZ"   "WBD"  "GS"   "JPM"  "MMC"  "MS"   "PGR"  "V"
## [31] "AMZN" "GM"   "HD"   "MAR"  "ORLY" "TJX"
```

```r
portfolio$capital
```

```
## [1] 5e+05
```

```r
portfolio$returns
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA NA NA NA NA
```

```r
print("")
```

```
## [1] ""
```

```r
# inspect the names and data for one stock
names(portfolio$data)
```

```
##  [1] "ADP"  "BA"   "ETN"  "FDX"  "GE"   "UNP"  "ELV"  "ISRG" "JNJ"  "LLY"
## [11] "PFE"  "UNH"  "AVGO" "CSCO" "INTU" "MSFT" "QCOM" "TXN"  "EA"   "GOOG"
## [21] "NFLX" "TTWO" "VZ"   "WBD"  "GS"   "JPM"  "MMC"  "MS"   "PGR"  "V"
## [31] "AMZN" "GM"   "HD"   "MAR"  "ORLY" "TJX"
```

```r
head(portfolio$data[[1]])
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-03-02              1      0.007035757    0.004563454 -0.0008205272
## 2016-03-09              1      0.022379780    0.007035757  0.0045634540
## 2016-03-16              1      0.009875713    0.022379780  0.0070357570
## 2016-03-23              1      0.006978770    0.009875713  0.0223797800
## 2016-03-30              1      0.018779390    0.006978770  0.0098757130
## 2016-04-06             -1     -0.006627075    0.018779390  0.0069787700
##            adjclose_lag2 adjclose_lag3 atr adx aaron bb chaikin_vol
## 2016-03-02  0.0541783600 -0.0161991100  NA  NA    50 NA          NA
## 2016-03-09 -0.0008205272  0.0541783600  NA  NA    50 NA          NA
## 2016-03-16  0.0045634540 -0.0008205272  NA  NA   100 NA          NA
## 2016-03-23  0.0070357570  0.0045634540  NA  NA   100 NA          NA
```

```
## 2016-03-30  0.0223797800  0.0070357570  NA  NA   100 NA          NA
## 2016-04-06  0.0098757130  0.0223797800  NA  NA   100 NA          NA
##                      clv          emv macd mfi      sar smi     volat month_index
## 2016-03-02            NA           NA   NA  NA 78.83754  NA        NA           3
## 2016-03-09  0.075378023 0.002275049   NA  NA 79.59079  NA 0.2380100           3
## 2016-03-16  0.175116926 0.009077995   NA  NA 80.26871  NA 0.2389290           3
## 2016-03-23  0.162085438 0.010112252   NA  NA 81.18206  NA 0.2214060           3
## 2016-03-30 -0.003746352 0.006978234   NA  NA 82.24717  NA 0.1992566           3
## 2016-04-06  0.156024412 0.006624761   NA  NA 83.45243  NA 0.1872713           4
##            arima_100_001 arima_010_001 arima_110_001 arima_020_001
## 2016-03-02   0.004316747   0.006978770  0.0086298106   0.004081827
## 2016-03-09   0.004203734   0.018779390  0.0120539202   0.030580010
## 2016-03-16   0.004447047  -0.006627075  0.0078527076  -0.032033540
## 2016-03-23   0.004396324  -0.001330622 -0.0043492036   0.003965831
## 2016-03-30   0.004383581   0.000000000 -0.0007583549   0.001330622
## 2016-04-06   0.004569211  -0.019383310 -0.0083362747  -0.038766620
##            arima_120_001 arima_100_011 arima_010_011 arima_110_011
## 2016-03-02  -0.002876931   0.004316747   0.006978770  0.0086298106
## 2016-03-09   0.019934078   0.004203734   0.018779390  0.0120539202
## 2016-03-16  -0.005083216   0.004447047  -0.006627075  0.0078527076
## 2016-03-23  -0.018273310   0.004396324  -0.001330622 -0.0043492036
## 2016-03-30   0.004203205   0.004383581   0.000000000 -0.0007583549
## 2016-04-06  -0.023762833   0.004569211  -0.019383310 -0.0083362747
##            arima_020_011 arima_120_011 vol_forecast
## 2016-03-02   0.004081827  -0.002876931    0.1992566
## 2016-03-09   0.030580010   0.019934078    0.1872713
## 2016-03-16  -0.032033540  -0.005083216    0.1614380
## 2016-03-23   0.003965831  -0.018273310    0.1423489
## 2016-03-30   0.001330622   0.004203205    0.1369465
## 2016-04-06  -0.038766620  -0.023762833    0.1102818
```