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 0.0711858302   162886578
## 2    ACN Information Technology 0.0054443906     6992528
## 3   ADBE Information Technology 0.0065918992     5052660
## 4    ADI Information Technology 0.0024309621     5558330
## 5   ADSK Information Technology 0.0012345301     2369175
## 6   AKAM Information Technology 0.0004536286     1691909
```

0.0.3 Retrieving top sectors and stocks

Pack everything into one function to retrieve all the data

```
# 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"  "GILD" "ISRG" "JNJ"  "LLY"
## [11] "MDT"  "MRK"  "PFE"  "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

```
# 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")
```

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

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

```
# 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"
```

```
# 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      0.009733979    0.008113075   0.039930970
## 2022-11-02              1      0.012305970    0.009733979   0.008113075
## 2022-11-09              1      0.053616090    0.012305970   0.009733979
## 2022-11-16              1      0.034718700    0.053616090   0.012305970
## 2022-11-23              1      0.005923635    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923635   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535800   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535800  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113075   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733979   0.008113075 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012305970   0.009733979 10.243009 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012305970 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

The data-fetching logic includes addition of a numerical index indicating to which month in the simulation the observations belong.

```
# 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      0.059684716    0.031058456  -0.008377096
## 2022-11-16              1      0.026221708    0.059684716   0.031058456
## 2022-11-23              1      0.022307721    0.026221708   0.059684716
## 2022-11-30             NA               NA    0.022307721   0.026221708
##            adjclose_lag2 adjclose_lag3      atr      adx aaron         bb
## 2022-09-28  -0.020913291   0.007554287 1.441481 16.24190  -100 0.04467755
## 2022-10-05  -0.069267283  -0.020913291 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   0.059684716   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
```

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

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

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

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

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

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

```
print("")
```

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

```
# 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
  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 simiulation (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:543187.393568223$"
## [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] "HON"   "CSX"   "LMT"   "ADP"   "NOC"   "BA"    "ISRG"  "MDT"   "ABT"
## [10] "DHR"   "BMY"   "JNJ"   "INTC"  "AMD"   "QCOM"  "ORCL"  "AAPL"  "ADBE"
## [19] "NFLX"  "CMCSA" "VZ"    "TMUS"  "EA"    "WBD"   "MA"    "MS"    "PGR"
## [28] "BLK"   "SPGI"  "AXP"   "NKE"   "F"     "MCD"   "GM"    "ABNB"  "AZO"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0434214875330438"   "  0.0519929994188102"   "  0.0477183403071731"
##  [4] "  0.0484325208910695"   "  0.0277702968861579"   "  0.00518665370711774"
##  [7] "  0.0441452980154356"   "  0.0313633544335068"   "  0.00452720044075704"
## [10] "  0.0219734067727598"   "  0.0113792920511932"   "  0.0242660778524668"
## [13] "  0.00212498155565191"  "  0.000743699433634136" "  0.00605274857602658"
## [16] "  0.0407640216991498"   "  0.0123486749553534"   "  0.0118684732764272"
## [19] "  0.00732169573207395"  "  0.00761962578011251"  "  0.00246498348351636"
## [22] "  0.0332887436326676"   "  0.0349407725778927"   "  0.0257480286048208"
## [25] "  0.0175432308801082"   "  0.0264750045107239"   "  0.0157226237781367"
## [28] "  0.0201888850473793"   "  0.0400763909781408"   "  0.0502233921711418"
## [31] "  0.0342520854194661"   "  0.0130889018530172"   "  0.0375536706589995"
## [34] "  0.00601022363938667"  "  0.0275142497643519"   "  0.025343597194119"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "---------------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=2) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:534752.739581924$"
## [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] "HON"   "UPS"   "DE"    "LMT"   "ITW"   "NOC"   "TMO"   "ELV"   "MRK"
## [10] "JNJ"   "ISRG"  "UNH"   "AMD"   "NVDA"  "IBM"   "AVGO"  "MSFT"  "TXN"
## [19] "CHTR"  "ATVI"  "META"  "GOOGL" "NFLX"  "TTWO"  "SCHW"  "AXP"   "MS"
## [28] "GS"    "JPM"   "V"     "BKNG"  "ORLY"  "AZO"   "F"     "GM"    "NKE"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0368125759766849"   "  0.0178095366405737"   "  0.0402085951306461"
##  [4] "  0.0590470970884258"   "  0.0202545606500428"   "  0.0443115370367104"
##  [7] "  0.00370930030800357"  "  0.051551940076278"    "  0.027963573713397"
## [10] "  0.020227415553661"    "  0.056314427742258"    "  0.0421649180686866"
## [13] "  0.0578507911715801"   "  0.0131565955691205"   "  0.0318181283626666"
## [16] "  0.00580683337312957"  "  0.0450900019599243"   "  0.004244641918767"
## [19] "  0.0215525170651754"   "  0.0122861711593233"   "  0.0277327053939243"
## [22] "  0.0268016964627965"   "  0.0534869101534925"   "  0.0450592281431489"
## [25] "  0.0132360851545302"   "  0.0576669574013219"   "  0.000649264894684792"
## [28] "  0.0180280025935487"   "  0.0242047053450021"   "  0.0175195101293601"
## [31] "  0.0214448325336906"   "  0.052860988699021"    "  0.00268407187727184"
## [34] "  0.0457238027410223"   "  0.000365041554529438" "  0.0337321007685932"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=3) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:538873.360964793$"
## [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] "ITW"  "RTX"  "UNP"  "ETN"  "GE"   "FDX"  "ABT"  "ABBV" "UNH"  "MRK"
## [11] "JNJ"  "PFE"  "ADBE" "TXN"  "NVDA" "IBM"  "ORCL" "AMD"  "VZ"   "TTWO"
## [21] "CHTR" "GOOG" "ATVI" "NFLX" "MS"   "GS"   "CB"   "BAC"  "JPM"  "SPGI"
## [31] "MCD"  "TSLA" "F"    "GM"   "MAR"  "NKE"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0363733809322285"   "  0.0492707035576554"   "  0.0400165203622876"
##  [4] "  0.0378533236922976"   "  0.0616993247305046"   "  0.065353650279607"
##  [7] "  0.00915990804993791"  "  0.0268208125967121"   "  0.00589142020295684"
## [10] "  0.0399018998767024"   "  0.0534080796824024"   "  0.0488537348297488"
## [13] "  0.036907954113145"    "  0.00921871200755916"  "  0.0527390813687531"
## [16] "  0.0149724934797092"   "  0.0337963786761371"   "  0.000772274309627654"
## [19] "  0.061192027632991"    "  0.0170427958964804"   "  0.0306346136084164"
## [22] "  0.0627392452488368"   "  0.0603438199164281"   "  0.000448242335767599"
## [25] "  0.0527580887945438"   "  0.0246398727851509"   "  0.0369890123628387"
## [28] "  0.0410464086386937"   "  0.053438528232557"    "  0.0415232553395287"
## [31] "  0.0172143596897606"   "  0.0305787491323788"   "  0.01113500356526"
## [34] "  0.0190484623403364"   "  0.0287156216939922"   "  0.000591598289625449"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=4) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:561555.355243455$"
## [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] "BA"    "DE"    "ITW"   "GE"    "ETN"   "HON"   "LLY"   "DHR"   "TMO"
## [10] "JNJ"   "MRK"   "AMGN"  "ORCL"  "ACN"   "AAPL"  "INTU"  "AMD"   "TXN"
## [19] "CMCSA" "DIS"   "OMC"   "T"     "GOOG"  "ATVI"  "JPM"   "CB"    "BAC"
## [28] "WFC"   "SPGI"  "C"     "GM"    "SBUX"  "AMZN"  "CMG"   "ABNB"  "TJX"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0242547958271112"   "  0.0365646714939488"   "  0.00659281783304112"
##  [4] "  0.0200129988587194"   "  0.0464501717840791"   "  0.017303826899157"
##  [7] "  0.0242479722104605"   "  0.00565021910681316"  "  0.00872689695178074"
## [10] "  0.0267914188615778"   "  0.0525820449016653"   "  0.0381236442456288"
## [13] "  0.0343318874013399"   "  0.0428442263964685"   "  0.0181022191305776"
## [16] "  0.00466535635627405"  "  0.0206497616991898"   "  0.0415820797781831"
## [19] "  0.0185504092691437"   "  0.00925046285341602"  "  0.0149145547394548"
## [22] "  0.00145774813699796"  "  0.00198487385833742"  "  0.0538386968591985"
## [25] "  0.0510498527574849"   "  0.0506058014091939"   "  0.0374045427390466"
## [28] "  0.000632167467622229" "  0.0293750866866383"   "  0.00813272859449821"
## [31] "  0.00562590896178808"  "  0.0063410930997081"   "  0.0513465673589715"
## [34] "  0.0418522238864005"   "  0.00762214002017013"  "  0.0116764772489881"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=5) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:605672.240643508$"
## [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] "DE"    "ADP"   "FDX"   "UNP"   "ITW"   "UPS"   "DHR"   "GILD"  "TMO"
## [10] "MDT"   "MRK"   "ELV"   "ORCL"  "ADBE"  "QCOM"  "NVDA"  "ACN"   "TXN"
## [19] "VZ"    "TTWO"  "WBD"   "ATVI"  "GOOGL" "GOOG"  "SPGI"  "WFC"   "C"
## [28] "MA"    "MS"    "BLK"   "ABNB"  "GM"    "F"     "NKE"   "MAR"   "BKNG"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0201558889762147"   "  0.0161575411772801"   "  0.014642595254972"
##  [4] "  0.0270402168638538"   "  0.0156059280792005"   "  0.00944451779485001"
##  [7] "  0.0230229242290331"   "  0.00293799282472245"  "  0.0303518803410252"
## [10] "  0.0163440177064211"   "  0.0190617988151027"   "  0.0420562046288211"
## [13] "  0.0430737911826343"   "  0.0300389158128435"   "  0.00646063597241311"
## [16] "  0.00836029497873428"  "  0.0171150261581656"   "  0.000983813050193622"
## [19] "  0.022383331994101"    "  0.0496751721567725"   "  0.0258224384847889"
## [22] "  0.0264131642310339"   "  0.0400895903498157"   "  0.0339004694842015"
## [25] "  0.0349058584436052"   "  0.0411701589973144"   "  0.0254378372530393"
## [28] "  0.0448734676167979"   "  0.0318483795933985"   "  0.048941045442974"
## [31] "  0.0131635671997965"   "  0.00299479756828324"  "  0.039948594637201"
## [34] "  0.0215289076668465"   "  0.0168065743895494"   "  0.0425608150416995"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
```

SECTOR_PROCEDURE

τ and window logic

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
#get a sample stock xts data
sample_xts <- sp500_stocks$Industrials$ADP
head(sample_xts, 5)
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-01-06             -1      -0.04944231             NA            NA
## 2016-01-13              1       0.01131390    -0.04944231            NA
## 2016-01-20              1       0.02848332     0.01131390   -0.04944231
## 2016-01-27              1       0.02053790     0.02848332    0.01131390
## 2016-02-03             -1      -0.01619834     0.02053790    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   -0.04944231            NA  NA  NA    50 NA          NA  NA  NA
## 2016-02-03    0.01131390   -0.04944231  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
```

The follwoing function extracts the specific window

```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_window <- f_extract_window(sample_xts, # stock xts
                                       tau=tau, # current run
                                       n_months = N_window # size of window
                                       )

# display some columns for the extracted data
head(sample_xts_window[,c("direction_lead", "clv", "volat", "month_index")], 10)
```

```
##            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
## 2016-04-13             -1  0.179603681 0.1614380           4
## 2016-04-20             -1 -0.024327317 0.1423489           4
## 2016-04-27             -1  0.019058758 0.1369465           4
## 2016-05-04             -1  0.051121037 0.1102818           5
```

EXTRACT_DYNAMIC_FEATURES

Three functions: - `f_add_garch_forecast()`: Computes the GARCH - `f_add_arima_forecast()`: Computes additional ARIMA features - `f_extract_dynamic_features()`: Combines the previous two functions

```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    0.01230597
## 2022-11-23              1      0.005923635    0.034718700    0.05361609
## 2022-11-30             NA               NA    0.005923635    0.03471870
##            adjclose_lag2 adjclose_lag3      atr      adx aaron        bb
## 2022-11-16   0.009733979   0.008113075 10.23247 14.68326   100 0.8325740
## 2022-11-23   0.012305970   0.009733979 10.24301 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012305970 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      0.009733979    0.008113075   0.039930970
## 2022-11-02              1      0.012305970    0.009733979   0.008113075
## 2022-11-09              1      0.053616090    0.012305970   0.009733979
## 2022-11-16              1      0.034718700    0.053616090   0.012305970
## 2022-11-23              1      0.005923635    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923635   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535800   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535800  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113075   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733979   0.008113075 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012305970   0.009733979 10.243009 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012305970 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   0.005473016
## 2022-11-02  5.546375 0.2606250          83    0.2759187   0.003833988
## 2022-11-09  3.943960 0.2653165          83    0.2633755   0.003715045
## 2022-11-16  6.291102 0.2641173          83    0.2642679   0.003708274
## 2022-11-23 11.099826 0.2624611          83    0.2651389   0.003707889
## 2022-11-30 16.713518 0.2759187          83    0.2659892   0.003707867
##             arima_010_001 arima_110_001 arima_020_001 arima_120_001
## 2022-10-26    0.034718700    0.04342609    0.01582131    0.05513176
## 2022-11-02    0.005923635    0.01919160   -0.02287143   -0.01640908
## 2022-11-09    0.005923635    0.01307809   -0.05166649   -0.04296117
## 2022-11-16    0.005923635    0.01589502   -0.08046156   -0.06675836
## 2022-11-23    0.005923635    0.01459706   -0.10925662   -0.09235426
## 2022-11-30    0.005923635    0.01519513   -0.13805169   -0.11677576
##             arima_100_011 arima_010_011 arima_110_011 arima_020_011
## 2022-10-26    0.005473016   0.034718700    0.04342609    0.01582131
## 2022-11-02    0.003833988   0.005923635    0.01919160   -0.02287143
## 2022-11-09    0.003715045   0.005923635    0.01307809   -0.05166649
## 2022-11-16    0.003708274   0.005923635    0.01589502   -0.08046156
## 2022-11-23    0.003707889   0.005923635    0.01459706   -0.10925662
## 2022-11-30    0.003707867   0.005923635    0.01519513   -0.13805169
##             arima_120_011
## 2022-10-26     0.05513176
## 2022-11-02    -0.01640908
## 2022-11-09    -0.04296117
## 2022-11-16    -0.06675836
## 2022-11-23    -0.09235426
## 2022-11-30    -0.11677576
```

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

```
##             actual_returns vol_forecast
## 2016-01-06              NA           NA
## 2016-01-13   -0.0494423100           NA
## 2016-01-20    0.0113139000           NA
## 2016-01-27    0.0284833200           NA
## 2016-02-03    0.0205379000           NA
## 2016-02-10   -0.0161983400    0.2380100
## 2016-02-17    0.0541779200    0.2389290
## 2016-02-24   -0.0008206329    0.2214060
## 2016-03-02    0.0045637700    0.1992566
## 2016-03-09    0.0070355470    0.1872713
##        ...
## 2022-09-28    0.0066181370    0.2269538
## 2022-10-05    0.0301509800    0.2606250
## 2022-10-12   -0.0645358000    0.2653165
## 2022-10-19    0.0399309700    0.2641173
## 2022-10-26    0.0081130750    0.2624611
## 2022-11-02    0.0097339790    0.2759187
## 2022-11-09    0.0123059700    0.2633755
## 2022-11-16    0.0536160900    0.2642679
## 2022-11-23   0.0347187000    0.2651389
## 2022-11-30   0.0059236350    0.2659892
```

```
# 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      0.009733979    0.008113075   0.039930970
## 2022-11-02              1      0.012305970    0.009733979   0.008113075
## 2022-11-09              1      0.053616090    0.012305970   0.009733979
## 2022-11-16              1      0.034718700    0.053616090   0.012305970
## 2022-11-23              1      0.005923635    0.034718700   0.053616090
## 2022-11-30             NA               NA    0.005923635   0.034718700
##            adjclose_lag2 adjclose_lag3       atr      adx aaron        bb
## 2022-10-26  -0.064535800   0.030150980  9.676399 13.39493   100 0.6110784
## 2022-11-02   0.039930970  -0.064535800  9.885942 13.58997   100 0.6303335
## 2022-11-09   0.008113075   0.039930970  9.762661 13.77107    50 0.6307783
## 2022-11-16   0.009733979   0.008113075 10.232471 14.68326   100 0.8325740
## 2022-11-23   0.012305970   0.009733979 10.243009 15.95273   100 0.9310325
## 2022-11-30   0.053616090   0.012305970 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   0.005473016
## 2022-11-02  5.546375 0.2606250          83    0.2759187   0.003833988
## 2022-11-09  3.943960 0.2653165          83    0.2633755   0.003715045
## 2022-11-16  6.291102 0.2641173          83    0.2642679   0.003708274
## 2022-11-23 11.099826 0.2624611          83    0.2651389   0.003707889
## 2022-11-30 16.713518 0.2759187          83    0.2659892   0.003707867
##            arima_010_001 arima_110_001 arima_020_001 arima_120_001
## 2022-10-26   0.034718700    0.04342609    0.01582131    0.05513176
## 2022-11-02   0.005923635    0.01919160   -0.02287143   -0.01640908
## 2022-11-09   0.005923635    0.01307809   -0.05166649   -0.04296117
## 2022-11-16   0.005923635    0.01589502   -0.08046156   -0.06675836
## 2022-11-23   0.005923635    0.01459706   -0.10925662   -0.09235426
## 2022-11-30   0.005923635    0.01519513   -0.13805169   -0.11677576
##            arima_100_011 arima_010_011 arima_110_011 arima_020_011
## 2022-10-26   0.005473016   0.034718700    0.04342609    0.01582131
## 2022-11-02   0.003833988   0.005923635    0.01919160   -0.02287143
## 2022-11-09   0.003715045   0.005923635    0.01307809   -0.05166649
## 2022-11-16   0.003708274   0.005923635    0.01589502   -0.08046156
## 2022-11-23   0.003707889   0.005923635    0.01459706   -0.10925662
## 2022-11-30   0.003707867   0.005923635    0.01519513   -0.13805169
##            arima_120_011
## 2022-10-26    0.05513176
## 2022-11-02   -0.01640908
## 2022-11-09   -0.04296117
## 2022-11-16   -0.06675836
## 2022-11-23   -0.09235426
## 2022-11-30   -0.11677576
```

SECTOR PROCEDURE

```r
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 current window of data for all stocks
  list_xts_sector <- lapply(sector_data,
                                f_extract_window,
                                tau=tau, # current run
                                n_months = N_window# size of window
                                )

  # 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

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

  # should return the 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)"
```

```r
names(sector_stocks_window) # names are tickers, values are list of xts
```

```
## [1] "BA"  "HON" "LMT" "RTX" "UNP" "UPS"
```

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

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-05-04              1      0.010394900   -0.021982980   0.002254581
## 2016-05-11             -1     -0.007327315    0.010394900  -0.021982980
## 2016-05-18              1      0.017853670   -0.007327315   0.010394900
## 2016-05-25             -1     -0.006822527    0.017853670  -0.007327315
## 2016-06-01              1      0.018521580   -0.006822527   0.017853670
## 2016-06-08             -1     -0.004749823    0.018521580  -0.006822527
##            adjclose_lag2 adjclose_lag3      atr adx aaron        bb chaikin_vol
## 2016-05-04   0.003827244   0.014750400 2.806036  NA  -100        NA          NA
## 2016-05-11   0.002254581   0.003827244 2.767678  NA   -50        NA          NA
## 2016-05-18  -0.021982980   0.002254581 2.706182  NA   -50 0.7060498          NA
## 2016-05-25   0.010394900  -0.021982980 2.687214  NA   -50 0.7553511  0.07668952
## 2016-06-01  -0.007327315   0.010394900 2.649171  NA    50 0.6851934 -0.34262110
## 2016-06-08   0.017853670  -0.007327315 2.626784  NA    50 0.7742400 -0.11146500
##                     clv          emv macd      mfi      sar smi     volat
## 2016-05-04  0.029468253 2.466634e-03   NA 73.71111 105.7217  NA 0.1344915
## 2016-05-11 -0.117303753 2.874767e-03   NA 72.79594 106.7908  NA 0.1172719
## 2016-05-18 -0.152995711 1.470097e-03   NA 66.65111 107.4352  NA 0.1266975
## 2016-05-25 -0.178394555 1.198783e-03   NA 63.39762 107.4352  NA 0.1199174
## 2016-06-01 -0.007431002 1.425215e-05   NA 69.99315 108.1274  NA 0.1289614
## 2016-06-08  0.068982862 1.278696e-03   NA 68.79011 108.6269  NA 0.1299072
##            month_index arima_100_001 arima_010_001 arima_110_001 arima_020_001
## 2016-05-04           5   0.003137127  -0.006822527   0.005426618   -0.03149872
## 2016-05-11           5   0.002049163   0.018521580   0.005940887    0.04386569
## 2016-05-18           5   0.003048150  -0.004749823   0.006801989   -0.02802123
## 2016-05-25           5   0.002455831   0.009048252   0.002198954    0.02284633
## 2016-06-01           6   0.003628388  -0.018266460  -0.004707569   -0.04558117
## 2016-06-08           6   0.001968150   0.020408760   0.001210567    0.05908398
##            arima_120_001 arima_100_011 arima_010_011 arima_110_011
## 2016-05-04   0.003209257   0.003137127  -0.006822527   0.005426618
## 2016-05-11   0.009044149   0.002049163   0.018521580   0.005940887
## 2016-05-18   0.005822367   0.003048150  -0.004749823   0.006801989
## 2016-05-25  -0.002959518   0.002455831   0.009048252   0.002198954
## 2016-06-01  -0.016960585   0.003628388  -0.018266460  -0.004707569
## 2016-06-08   0.013145217   0.001968150   0.020408760   0.001210567
##            arima_020_011 arima_120_011 vol_forecast
## 2016-05-04   -0.03149872   0.003209257    0.1289614
## 2016-05-11    0.04386569   0.009044149    0.1299072
## 2016-05-18   -0.02802123   0.005822367    0.1234845
## 2016-05-25    0.02284633  -0.002959518    0.1212023
## 2016-06-01   -0.04558117  -0.016960585    0.1308565
## 2016-06-08    0.05908398   0.013145217    0.1399425
```

MODELLING_PROCEDURE

Recall that the **SECTOR_PROCEDURE**(G, τ) function takes the argument G, which is the **sector name**, and **tau**, which is the current run in the backtesting.

This procedure happens in a loop, for every sector G. Here, we fix one sector only, and a specific τ. The code does the following:

1. Retrieves the actual sector stock data (list of key-value pairs, keys are stock tickers, values are xts full data for that stock.)
2. Creates a variable to store the subset of data that goes into the current window.
3. The `f_extract_window()` function extracts the appropriate window of data corresponding to the τ, with the appropriate window size, for all sectors.
4. Extracts the dynamic features (ARIMA and GARCH) for that each stock in the sector.

```r
# parameters
G <- names(sp500_stocks)[1] # sample sector
tau <- 10 # 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_xts_sector <- lapply(sector_data,
                          f_extract_window,
                          tau=tau, # current run
                          n_months = N_window# size of window
                          )

# compute GARCH features for all stocks
list_xts_sector <- lapply(list_xts_sector,
                          f_extract_dynamic_features,
                          return_col = "realized_returns",
                           volat_col = "volat"
                          )

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


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

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

```r
# each stock has the xts subset (for window)
head(list_xts_sector[[1]])
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-10-05             -1     -0.008140096    0.001485647  -0.016220080
## 2016-10-12              1      0.006425871   -0.008140096   0.001485647
## 2016-10-19             -1     -0.002749151    0.006425871  -0.008140096
## 2016-10-26              1      0.031498020   -0.002749151   0.006425871
## 2016-11-02              1      0.010172550    0.031498020  -0.002749151
## 2016-11-09              1      0.025738470    0.010172550   0.031498020
##            adjclose_lag2 adjclose_lag3      atr      adx aaron        bb
## 2016-10-05   0.024948810  -0.037026670 1.900259 15.44565   -50 0.2934560
## 2016-10-12  -0.016220080   0.024948810 1.872384 15.23639  -100 0.2289285
## 2016-10-19   0.001485647  -0.016220080 1.800070 14.75791   -50 0.3060118
## 2016-10-26  -0.008140096   0.001485647 1.722923 14.44363   100 0.2860935
## 2016-11-02   0.006425871  -0.008140096 1.864142 14.04553    50 0.4910556
## 2016-11-09  -0.002749151   0.006425871 1.989560 13.44222   100 0.5094234
##             chaikin_vol         clv           emv      macd      mfi      sar
## 2016-10-05   -0.4622892  0.18091008 -0.0006643160 1.3477744 46.50802 95.02127
## 2016-10-12    0.3990933  0.24064338 -0.0026850063 1.1358585 37.92195 94.68802
## 2016-10-19   -0.4336751  0.09899013 -0.0019094937 0.9402188 36.19915 94.36810
## 2016-10-26   -1.0188680 -0.01496489 -0.0021492280 0.7585276 30.28217 94.06097
## 2016-11-02 -324.8278000  0.05096933 -0.0009225739 0.6437468 48.88575 93.76613
## 2016-11-09    1.1391500  0.19338517 -0.0009562142 0.5919089 59.37208 93.48309
##                   smi      volat month_index arima_100_001 arima_010_001
## 2016-10-05  -5.331162 0.10247324          10   0.003087229   0.031498020
## 2016-10-12 -11.930732 0.10506831          10   0.005293380   0.010172550
## 2016-10-19 -17.430099 0.10335977          10   0.003683063   0.025738470
## 2016-10-26 -19.828752 0.09985285          10   0.002663082   0.035597980
## 2016-11-02 -18.073978 0.13389984          11   0.007022322  -0.006540046
## 2016-11-09 -13.909935 0.16512456          11   0.004073038   0.021968830
##            arima_110_001 arima_020_001 arima_120_001 arima_100_011
## 2016-10-05   0.012973583    0.06574519  3.470592e-02   0.003087229
## 2016-10-12   0.021707585   -0.01115292  2.857179e-02   0.005293380
## 2016-10-19   0.017318799    0.04130439  1.493350e-02   0.003683063
## 2016-10-26   0.030264930    0.04545749  4.953657e-02   0.002663082
## 2016-11-02   0.016252587   -0.04867807 -1.150893e-02   0.007022322
## 2016-11-09   0.006548261    0.05047771 -2.246684e-05   0.004073038
##            arima_010_011 arima_110_011 arima_020_011 arima_120_011 vol_forecast
## 2016-10-05   0.031498020   0.012973583    0.06574519  3.470592e-02    0.1338998
## 2016-10-12   0.010172550   0.021707585   -0.01115292  2.857179e-02    0.1651246
## 2016-10-19   0.025738470   0.017318799    0.04130439  1.493350e-02    0.1746223
## 2016-10-26   0.035597980   0.030264930    0.04545749  4.953657e-02    0.1752898
## 2016-11-02  -0.006540046   0.016252587   -0.04867807 -1.150893e-02    0.1772747
## 2016-11-09   0.021968830   0.006548261    0.05047771 -2.246684e-05    0.1757262
```

The result is the `list_train_val_sector` oject, which is a list of lists. - The first level are the stock tickers - The second level are train and val xts for each stock.

```r
# Check num of rows (weeks) for window
nrow(list_xts_sector[[1]])
```

```
## [1] 103
```

Feature Selection

Notes: - This will use **forward selection** to extract the features from a sample stock for the current sector. - The `target_var` argument specifies the target variable, in this case is called "realized_returns". - `f_select_features()` is found under `functions/feature_engineering.R`

```r
# Extract a sample stock in the list_xts_sector
sample_sector_stock <- list_xts_sector[[1]]

# 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 = sample_sector_stock, # for one stock of current sector
                    target_var = "realized_returns", # y
```

```
                     volat_col = "volat", # we always want to keep the volatility col
                     garch_col = "forecasted_volatility", # gach col
                     nvmax = 15, # examine all possible subsets
                     method="exhaustive") #  we always want to use forward selection
```

```
## Loading required package: leaps

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 5 linear dependencies found

## Reordering variables and trying again:
```

```
best_feat_list
```

```
## $featnames
##  [1] "direction_lead"        "actual_returns"        "adjclose_lag1"
##  [4] "adjclose_lag2"         "adjclose_lag3"         "clv"
##  [7] "macd"                  "mfi"                   "sar"
## [10] "smi"                   "arima_110_001"         "arima_020_001"
## [13] "arima_120_001"         "vol_forecast"          "volat"
## [16] "forecasted_volatility"
##
## $fmla
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + clv + macd + mfi + sar +
##     smi + arima_110_001 + arima_020_001 + arima_120_001 + vol_forecast +
##     volat + forecasted_volatility
## <environment: 0x000001f0e98ca400>
```

Regularized MLR (Elasticnet)

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

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

# 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 = 0.005))  # 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,
  msr = NA, # modified sharpe ratio
  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"         "msr"            "rmse"
## [5] "data"
```

Fitting all the models

Next, we loop through every stock doing the following: 1. Extracting the train and validation sets, and filter NAs 2. Perform feature selection for every stock 3. Fit an Elasticnet model for that stock, and obtain predictions for the returns 4. Compute the RMSE 5. Compute the Sharpe Ratio and Modified Sharpe 6. Save everything

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

  ### Step 0: Data Preparation

  ###########################################################################
  ### NOTE: Need to refactor

  # fetch data for that ticker
  full_train <- list_xts_sector[[ticker]]

  # Re-extract train and val with full features
  full_train <- f_extract_train_val_no_window(full_train,
                                              val_lag = 1) # number of months in val

  # Reassign to train and val
  ticker_data_train <- full_train$train
  ticker_data_val <- full_train$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
```

```
                      volat_col = "volat", # always keep the actual volatility
                      garch_col = "vol_forecast",
                      nvmax = 20, # total number of max subsets
                      method="exhaustive")

print(best_feat_list$fmla)

### Step 2: Elasticnet

# Set up time-slice cross-validation parameters
ctr_train <- trainControl(method = "timeslice", # cross validation
                            initialWindow = 52,  # Consecutive number of weeks
                            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


# 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 = Elasticnet
                           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

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

### Step 3: Sharpe Ratio

# Calculate the Sharpe Ratio and MSR
stock_sharpe <- SharpeRatio(ticker_data_train[, "realized_returns"], Rf=0.02, FUN="StdDev")
stock_msr <- SharpeRatio(ticker_data_train[, "realized_returns"], Rf=0.02, FUN="ES")

### Step 4: Track the measures

sector_tracker[[ticker]]$forecasted_ret = pred_enet_best
sector_tracker[[ticker]]$rmse = enet_rmse
sector_tracker[[ticker]]$sharpe = stock_sharpe
sector_tracker[[ticker]]$msr = stock_msr
# sector_tracker[[ticker]]$data = rbind.xts(ticker_data_train, ticker_data_val) # This should be included at


# show values
print("*******************************************")
print(paste("predicted return: ", pred_enet_best))
print(paste("rmse: ", enet_rmse))
print(paste("sharpe: ", stock_sharpe))
```

```
  print(paste("msr: ", stock_msr))
  print("******************************************")

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

```
## [1] "ticker: ADP"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + clv + macd + mfi + sar +
##     smi + arima_010_001 + arima_120_001 + arima_020_011 + vol_forecast +
##     volat
## <environment: 0x000001f0ef2015f0>
## [1] "******************************************"
## [1] "predicted return:  0.00555963107252525"
## [1] "rmse:  0.00730296639740403"
## [1] "sharpe:  -0.54059527255065"
## [1] "msr:  -0.203838815258719"
## [1] "******************************************"
## [1] "###########################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + adx + bb + chaikin_vol +
##     clv + emv + macd + mfi + sar + smi + arima_120_001 + arima_010_011 +
##     arima_020_011 + volat + vol_forecast
## <environment: 0x000001f0efd91bf8>
## [1] "******************************************"
## [1] "predicted return:  0.0102579568260425"
## [1] "rmse:  0.0336373270693948"
## [1] "sharpe:  -0.326256523551611"
## [1] "msr:  -0.2003316865472"
## [1] "******************************************"
## [1] "###########################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adjclose_lag2 + atr + bb +
##     chaikin_vol + macd + mfi + smi + arima_110_011 + vol_forecast +
##     volat
## <environment: 0x000001f0e51d0748>
## [1] "******************************************"
## [1] "predicted return:  0.00858855824362219"
## [1] "rmse:  0.0285403742049561"
## [1] "sharpe:  -0.446026446350027"
## [1] "msr:  -0.262219343410875"
## [1] "******************************************"
## [1] "###########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adx + bb + emv + macd + mfi +
##     volat + arima_010_001 + arima_020_001 + arima_120_011 + vol_forecast
## <environment: 0x000001f0ed9f6388>
## [1] "******************************************"
## [1] "predicted return:  0.00923644920909091"
## [1] "rmse:  0.0099386566144085"
## [1] "sharpe:  -0.259991091575958"
## [1] "msr:  -0.119501302003582"
## [1] "******************************************"
## [1] "###########################################"
## [1] "ticker: DE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + aaron + clv + smi +
##     arima_110_011 + vol_forecast + volat
## <environment: 0x000001f0e75ff958>
## [1] "*****************************************"
## [1] "predicted return:  0.00571415535454546"
## [1] "rmse:  0.0235265210194567"
## [1] "sharpe:  -0.463224815776683"
## [1] "msr:  -0.247710596545545"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adx + bb + clv + emv + mfi +
##     sar + smi + volat + arima_120_001 + arima_100_011 + arima_110_011 +
##     vol_forecast
## <environment: 0x000001f0f06ff5c0>
## [1] "*****************************************"
## [1] "predicted return:  0.00322157953328978"
## [1] "rmse:  0.0147718312307693"
## [1] "sharpe:  -0.691845933979725"
## [1] "msr:  -0.390497046683367"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + adx + aaron + bb +
##     clv + mfi + sar + arima_120_001 + vol_forecast + volat
## <environment: 0x000001f0ea2c14b0>
## [1] "*****************************************"
## [1] "predicted return:  0.00299405127068077"
## [1] "rmse:  0.027843238413773"
## [1] "sharpe:  -0.642659021342466"
## [1] "msr:  -0.273002232090648"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + mfi + smi +
##     volat + arima_100_011 + vol_forecast
## <environment: 0x000001f0f4c26c60>
## [1] "*****************************************"
## [1] "predicted return:  -0.00526246590248153"
## [1] "rmse:  0.0776111248386338"
## [1] "sharpe:  -0.895359118928226"
## [1] "msr:  -0.354019032284728"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     adjclose_lag3 + adx + aaron + bb + clv + emv + macd + smi +
##     volat + arima_100_001 + arima_020_011 + arima_120_011 + vol_forecast
## <environment: 0x000001f0f2765690>
## [1] "*****************************************"
## [1] "predicted return:  0.00383980127484849"
## [1] "rmse:  0.00674149605501714"
## [1] "sharpe:  -0.84104697577433"
## [1] "msr:  -0.305105348185681"
## [1] "*****************************************"
## [1] "############################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag3 +
##     atr + adx + bb + emv + macd + mfi + sar + smi + volat + arima_120_001 +
##     arima_010_011 + arima_110_011 + vol_forecast
## <environment: 0x000001f0e59d2360>
## [1] "*****************************************"
## [1] "predicted return:  0.00207220665656566"
## [1] "rmse:  0.0223991066342854"
## [1] "sharpe:  -0.742470084183254"
## [1] "msr:  -0.294107517878324"
## [1] "*****************************************"
## [1] "############################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + chaikin_vol +
##     macd + mfi + arima_110_011 + arima_120_011 + vol_forecast +
##     volat
## <environment: 0x000001f0e89fd7b0>
## [1] "*****************************************"
## [1] "predicted return:  0.00360129961111111"
## [1] "rmse:  0.0206389300056138"
## [1] "sharpe:  -0.690321983996585"
## [1] "msr:  -0.315760526460759"
## [1] "*****************************************"
## [1] "############################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + atr + adx + aaron + clv + smi + vol_forecast +
##     volat
## <environment: 0x000001f0f5811328>
## [1] "*****************************************"
## [1] "predicted return:  0.00366770876666667"
## [1] "rmse:  0.0159474539006449"
## [1] "sharpe:  -0.614257011106587"
## [1] "msr:  -0.225579070605889"
## [1] "*****************************************"
## [1] "############################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + atr + chaikin_vol + clv + mfi + smi + volat +
##     arima_120_001 + arima_110_011 + vol_forecast
## <environment: 0x000001f0f102dd78>
## [1] "*****************************************"
## [1] "predicted return:  0.00326231875013594"
## [1] "rmse:  0.02342305404528"
## [1] "sharpe:  -0.813518719068738"
## [1] "msr:  -0.291497002129236"
## [1] "*****************************************"
## [1] "############################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + clv + emv + macd +
##     smi + volat + arima_110_001 + arima_120_011 + vol_forecast
## <environment: 0x000001f0e756c820>
## [1] "*******************************************"
## [1] "predicted return:  0.0066685227575514"
## [1] "rmse:  0.0164475925591865"
## [1] "sharpe:  -0.561155391279589"
## [1] "msr:  -0.265298091229738"
## [1] "*******************************************"
## [1] "############################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     atr + adx + aaron + bb + chaikin_vol + clv + macd + mfi +
##     smi + arima_010_001 + volat + vol_forecast
## <environment: 0x000001f0f3de8658>
## [1] "*******************************************"
## [1] "predicted return:  0.00195920532626263"
## [1] "rmse:  0.0243205951739076"
## [1] "sharpe:  -0.648506102824741"
## [1] "msr:  -0.178658926164713"
## [1] "*******************************************"
## [1] "############################################"
```

Aside: Format for Portfolio Optimization

```r
## 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] "ETN"  "FDX"  "HON"  "ITW"  "LMT"  "UPS"  "BMY"  "ELV"  "GILD" "MDT"
## [11] "MRK"  "UNH"  "AMD"  "CSCO" "INTU" "ORCL" "QCOM" "TXN"  "CHTR" "DIS"
## [21] "EA"   "GOOG" "TTWO" "WBD"  "BLK"  "CB"   "MA"   "SPGI" "V"    "WFC"
## [31] "ABNB" "BKNG" "F"    "MAR"  "MCD"  "ORLY"
```

```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] "ETN"  "FDX"  "HON"  "ITW"  "LMT"  "UPS"  "BMY"  "ELV"  "GILD" "MDT"
## [11] "MRK"  "UNH"  "AMD"  "CSCO" "INTU" "ORCL" "QCOM" "TXN"  "CHTR" "DIS"
## [21] "EA"   "GOOG" "TTWO" "WBD"  "BLK"  "CB"   "MA"   "SPGI" "V"    "WFC"
## [31] "ABNB" "BKNG" "F"    "MAR"  "MCD"  "ORLY"
```

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

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-03-02              1      0.020356052    0.020044608   0.014005956
## 2016-03-09              1      0.037208676    0.020356052   0.020044608
## 2016-03-16              1      0.027585493    0.037208676   0.020356052
## 2016-03-23              1      0.004312571    0.027585493   0.037208676
## 2016-03-30             -1     -0.052342510    0.004312571   0.027585493
## 2016-04-06              1      0.056953329   -0.052342510   0.004312571
##            adjclose_lag2 adjclose_lag3 atr adx aaron bb chaikin_vol        clv
## 2016-03-02    0.04211721    0.02298077  NA  NA    50 NA          NA         NA
## 2016-03-09    0.01400596    0.04211721  NA  NA   100 NA          NA 0.21957927
## 2016-03-16    0.02004461    0.01400596  NA  NA   100 NA          NA 0.26896962
## 2016-03-23    0.02035605    0.02004461  NA  NA   100 NA          NA 0.07207380
## 2016-03-30    0.03720868    0.02035605  NA  NA    50 NA          NA 0.02567948
## 2016-04-06    0.02758549    0.03720868  NA  NA  -100 NA          NA 0.06467860
##                     emv macd mfi      sar smi     volat month_index
## 2016-03-02           NA   NA  NA 49.03166  NA        NA           3
## 2016-03-09 0.002778481   NA  NA 49.92549  NA 0.3551345           3
## 2016-03-16 0.004656278   NA  NA 50.98324  NA 0.3550879           3
## 2016-03-23 0.006559789   NA  NA 52.39538  NA 0.3346551           3
## 2016-03-30 0.005135875   NA  NA 54.13372  NA 0.3094887           3
## 2016-04-06 0.003131404   NA  NA 55.59393  NA 0.2903115           4
##            arima_100_001 arima_010_001 arima_110_001 arima_020_001
## 2016-03-02  0.0037167907   0.004312571   0.018761505   -0.01896035
## 2016-03-09  0.0156405687  -0.052342510  -0.017168348   -0.10899759
## 2016-03-16 -0.0073621223   0.056953329  -0.010902713    0.16624917
## 2016-03-23  0.0075831061  -0.014057984   0.030029208   -0.08506930
## 2016-03-30 -0.0009552907   0.026511672   0.001324105    0.06708133
## 2016-04-06  0.0123682811  -0.036794442   0.002508997   -0.10010056
##            arima_120_001 arima_100_011 arima_010_011 arima_110_011
## 2016-03-02   -0.00872481  0.0037167907   0.004312571   0.018761505
## 2016-03-09   -0.08396528  0.0156405687  -0.052342510  -0.017168348
## 2016-03-16    0.04180741 -0.0073621223   0.056953329  -0.010902713
## 2016-03-23    0.05013778  0.0075831061  -0.014057984   0.030029208
## 2016-03-30   -0.01658999 -0.0009552907   0.026511672   0.001324105
## 2016-04-06   -0.02220714  0.0123682811  -0.036794442   0.002508997
##            arima_020_011 arima_120_011 vol_forecast
## 2016-03-02   -0.01896035   -0.00872481    0.3094887
## 2016-03-09   -0.10899759   -0.08396528    0.2903115
## 2016-03-16    0.16624917    0.04180741    0.2503830
## 2016-03-23   -0.08506930    0.05013778    0.2431052
## 2016-03-30    0.06708133   -0.01658999    0.2244655
## 2016-04-06   -0.10010056   -0.02220714    0.2168477
```