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.

```
# 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.

```
# 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

```
# 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 0.0717740247   161899523
## 2    ACN Information Technology 0.0054548923     6950153
## 3   ADBE Information Technology 0.0065736519     5022037
## 4    ADI Information Technology 0.0024098652     5524656
## 5   ADSK Information Technology 0.0012173370     2354824
## 6   AKAM Information Technology 0.0004503764     1681739
```

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)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146075      0.009733913
## 2022-11-02       232.4444              1      0.009781442      0.012306041
## 2022-11-09       235.3226              1      0.012382070      0.053616027
## 2022-11-16       248.2840              1      0.055079400      0.034718645
## 2022-11-23       257.0555              1      0.035328370      0.005923635
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3       atr
## 2022-10-26   0.008113074   0.039930970  -0.064535796   0.030150911  9.676399
## 2022-11-02   0.009733913   0.008113074   0.039930970  -0.064535796  9.885942
## 2022-11-09   0.012306041   0.009733913   0.008113074   0.039930970  9.762661
## 2022-11-16   0.053616027   0.012306041   0.009733913   0.008113074 10.232471
## 2022-11-23   0.034718645   0.053616027   0.012306041   0.009733913 10.243009
## 2022-11-30   0.005923635   0.034718645   0.053616027   0.012306041 10.247795
##                 adx aaron        bb chaikin_vol        clv         emv     macd
## 2022-10-26 13.39493   100 0.6110784 -1.49750300 -0.1320576 -0.01707202 2.049576
## 2022-11-02 13.58997   100 0.6303335  2.90314600 -0.2863719  0.02711271 1.939312
## 2022-11-09 13.77107    50 0.6307783 -0.09676625 -0.3920529  0.04765004 1.866926
## 2022-11-16 14.68326   100 0.8325740 -0.38397100 -0.4461119  0.09074850 1.906715
## 2022-11-23 15.95273   100 0.9310325 -0.20180520 -0.3205142  0.11758529 2.068291
## 2022-11-30 16.53998   100 0.8907336  0.48394890 -0.1089895  0.12144667 2.300754
##                 mfi      sar       smi     volat month_index
## 2022-10-26 51.52422 260.0428  8.131402 0.2269538          82
## 2022-11-02 49.23300 258.6055  5.546375 0.2606250          83
## 2022-11-09 49.20839 257.2257  3.943960 0.2653165          83
## 2022-11-16 48.83463 256.7200  6.291102 0.2641173          83
## 2022-11-23 49.31528 224.1100 11.099826 0.2624611          83
## 2022-11-30 42.97382 224.1100 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)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-09-28       27.24851              1     -0.051818743      0.006853026
## 2022-10-05       27.43588             -1      0.006876562     -0.042966085
## 2022-10-12       26.28203              1     -0.042056122      0.046554183
## 2022-10-19       27.53450              1      0.047654843      0.029989991
## 2022-10-26       28.37277             -1      0.030444220     -0.008377096
## 2022-11-02       28.13608              1     -0.008342106      0.031058456
## 2022-11-09       29.02365              1      0.031545802      0.059684778
## 2022-11-16       30.80866              1      0.061501885      0.026221586
## 2022-11-23       31.62720              1      0.026568397      0.022307781
## 2022-11-30       32.34066             NA      0.022558460               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2022-09-28  -0.053209596  -0.069267349  -0.020913351   0.007554347 1.441481
## 2022-10-05   0.006853026  -0.053209596  -0.069267349  -0.020913351 1.384232
```

```
## 2022-10-12  -0.042966085   0.006853026  -0.053209596  -0.069267349 1.379644
## 2022-10-19   0.046554183  -0.042966085   0.006853026  -0.053209596 1.394670
## 2022-10-26   0.029989991   0.046554183  -0.042966085   0.006853026 1.398622
## 2022-11-02  -0.008377096   0.029989991   0.046554183  -0.042966085 1.385863
## 2022-11-09   0.031058456  -0.008377096   0.029989991   0.046554183 1.385444
## 2022-11-16   0.059684778   0.031058456  -0.008377096   0.029989991 1.429341
## 2022-11-23   0.026221586   0.059684778   0.031058456  -0.008377096 1.395102
## 2022-11-30   0.022307781   0.026221586   0.059684778   0.031058456 1.369024
##                 adx aaron         bb chaikin_vol         clv           emv
## 2022-09-28 16.24190  -100 0.04467755  2.43234200  0.21475805 -1.787304e-04
## 2022-10-05 17.10559   -50 0.13495813 -0.44268680  0.22116568 -2.096124e-04
## 2022-10-12 18.24157   -50 0.07457368  0.43839330  0.07934922 -3.472192e-04
## 2022-10-19 18.58490    50 0.23730603 -1.12835800  0.03125187 -3.458817e-04
## 2022-10-26 18.20787   100 0.36428555  0.36773750 -0.10430028 -2.858648e-04
## 2022-11-02 17.63796   100 0.36718737 -8.91414900 -0.26417408 -1.913069e-04
## 2022-11-09 17.00435    50 0.43456871 -0.08886197 -0.35167976 -1.696224e-04
## 2022-11-16 16.04316   100 0.61239403 -0.69757770 -0.28307675 -6.177828e-05
## 2022-11-23 15.54651   100 0.68335600 -2.77541900 -0.16462184  6.920197e-05
## 2022-11-30 15.36369   100 0.70213009 -0.65517410  0.02947430  2.043992e-04
##                 macd      mfi      sar       smi     volat month_index
## 2022-09-28 -2.031918 46.90353 34.67000 -18.01681 0.2279791          81
## 2022-10-05 -2.290153 46.43088 34.38840 -22.89976 0.2353109          82
## 2022-10-12 -2.649750 46.62430 34.11806 -28.89441 0.2481376          82
## 2022-10-19 -2.983549 54.92321 33.66998 -32.89471 0.2465206          82
## 2022-10-26 -3.232381 56.20916 33.24878 -34.78229 0.2484444          82
## 2022-11-02 -3.420978 48.82911 32.85285 -36.26677 0.2806964          83
## 2022-11-09 -3.505779 48.94612 32.48068 -36.24474 0.2819226          83
## 2022-11-16 -3.415472 46.83053 32.13084 -32.84559 0.2767814          83
## 2022-11-23 -3.168499 45.87661 26.65000 -26.53377 0.2587499          83
## 2022-11-30 -2.797269 55.72098 26.65000 -18.89848 0.2672197          83
```

```
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
  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:490952.348255087$"
## [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] "ADP"  "BA"   "UPS"  "CSX"  "LMT"  "NOC"  "BMY"  "TMO"  "ABBV" "UNH"
## [11] "JNJ"  "LLY"  "MSFT" "ADBE" "IBM"  "INTC" "QCOM" "AMD"  "TTWO" "ATVI"
## [21] "DIS"  "T"    "GOOG" "WBD"  "MA"   "PGR"  "MS"   "MMC"  "CB"   "GS"
## [31] "MCD"  "ORLY" "BKNG" "NKE"  "F"    "ABNB"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0223536466183415"  "  0.0261455350020615"  "  0.0469474641767733"
##  [4] "  0.00368343832409166" "  0.0382514344705162"  "  0.0169501074389094"
##  [7] "  0.0415263711947884"  "  0.0335682900424783"  "  0.0335824600678399"
## [10] "  0.00675440484798133" "  0.0310019652212197"  "  0.0215548823423776"
## [13] "  0.025894461558644"   "  0.00528201133596659" "  0.0181858765887429"
## [16] "  0.0353441609087209"  "  0.0455728009770214"  "  0.0408325980898728"
## [19] "  0.0186907553364563"  "  0.0484314897412203"  "  0.0268389540048464"
## [22] "  0.0477168554054769"  "  0.0203622953370955"  "  0.0188269737092323"
## [25] "  0.0118593306762246"  "  0.0326738515080283"  "  0.0464210598024342"
## [28] "  0.0266171412412176"  "  0.0106036325704112"  "  0.0149290007275187"
## [31] "  0.0526679523145118"  "  0.00210219844412625" "  0.0536257689557608"
## [34] "  0.0100755407169814"  "  0.0547182721585927"  "  0.0384609626531496"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "--------------------------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=2) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:473785.249344554$"
## [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"   "CSX"   "DE"    "HON"   "ETN"   "RTX"   "DHR"   "GILD"  "ELV"
## [10] "PFE"   "JNJ"   "MRK"   "AVGO"  "CRM"   "INTU"  "AMD"   "TXN"   "ORCL"
## [19] "META"  "DIS"   "GOOGL" "T"     "WBD"   "GOOG"  "AXP"   "SCHW"  "V"
## [28] "MA"    "WFC"   "CB"    "GM"    "NKE"   "HD"    "MCD"   "F"     "BKNG"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0108905167864498"  "  0.0134925468136992"  "  0.0466190250200083"
##  [4] "  0.0116607933707783"  "  0.0479048192016295"  "  0.05171790614282"
##  [7] "  0.017663710808513"   "  0.0372783559290647"  "  0.0502543920588377"
## [10] "  0.00568529260164149" "  0.0235474475171104"  "  0.035005316482656"
## [13] "  0.0198568491250851"  "  0.0178007411497233"  "  0.0473919542892904"
## [16] "  0.0283049780442274"  "  0.0361050183514911"  "  0.0375690895713401"
## [19] "  0.0462309059981464"  "  0.0186168733715989"  "  0.0233170229002597"
## [22] "  0.023309864835035"   "  0.0288352728174562"  "  0.0149163855984248"
## [25] "  0.0321764802884576"  "  0.0512762465054228"  "  0.0504705955157736"
## [28] "  0.00290629147963721" "  0.025560173237342"   "  0.0248136397702203"
## [31] "  0.0395263896197288"  "  0.0490789184268561"  "  0.00745053410577993"
## [34] "  0.0275990350192033"  "  0.0211540873133843"  "  0.0177117569911326"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
```

```
## [1] "-------------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=3) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:470646.614367015$"
## [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] "GE"   "ITW"  "RTX"  "UNP"  "ETN"  "DE"   "ABBV" "PFE"  "MDT"  "ABT"
## [11] "UNH"  "BMY"  "INTC" "QCOM" "AMD"  "AVGO" "CRM"  "AAPL" "OMC"  "TMUS"
## [21] "DIS"  "GOOG" "WBD"  "ATVI" "SPGI" "BAC"  "GS"   "CB"   "C"    "JPM"
## [31] "AZO"  "SBUX" "ORLY" "HD"   "F"    "NKE"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0127500501115986"  "  0.0481903605969442"  "  0.0400053490531243"
##  [4] "  0.025269760547846"   "  0.0441200413668565"  "  0.0113366845277623"
##  [7] "  0.0536773845439774"  "  0.0124154875251505"  "  0.0451849658847928"
## [10] "  0.0224029422813571"  "  0.00679891302846038" "  0.0321800912883213"
## [13] "  0.0502049480133984"  "  0.030443286020743"   "  0.0507643816867861"
## [16] "  0.0148267728954112"  "  0.0232666856070041"  "  0.0510047763947615"
## [19] "  0.0228525466055836"  "  0.015994213573512"   "  0.0407419509291485"
## [22] "  0.0156302426396512"  "  0.0463400178612213"  "  0.00279768880912773"
## [25] "  0.0339489059887994"  "  0.0231623191557359"  "  0.0244635747822878"
## [28] "  0.00611920875146448" "  0.00607364871415112" "  0.0401476109568602"
## [31] "  0.0110012893046269"  "  0.0121844047818753"  "  0.0527511330334819"
## [34] "  0.0284479414850789"  "  0.0502989628042742"  "  0.039700902347885"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "-------------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=4) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:465276.426439167$"
## [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] "CSX"  "RTX"  "LMT"  "NOC"  "UNP"  "ETN"  "GILD" "AMGN" "DHR"  "UNH"
## [11] "BMY"  "JNJ"  "CRM"  "AMD"  "ACN"  "INTU" "CSCO" "AAPL" "EA"   "NFLX"
## [21] "GOOG" "TMUS" "T"    "META" "MA"   "CB"   "BAC"  "BLK"  "AXP"  "C"
## [31] "TJX"  "NKE"  "ORLY" "AMZN" "ABNB" "CMG"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
```

```
##  [1] "  0.0362088294644337"   "  0.00557546401675466"  "  0.0346588897683708"
##  [4] "  0.0147830126613249"   "  0.0176033163201746"   "  0.032701937399728"
##  [7] "  0.0228393100769671"   "  0.0204239940468075"   "  0.00369389372388949"
## [10] "  0.0236509697527587"   "  0.00964517731473515"  "  0.0333667290083559"
## [13] "  0.0328418527314704"   "  0.0401112897245397"   "  0.0112792255452912"
## [16] "  0.0257635458495605"   "  0.008168654783667"    "  0.0137939539144675"
## [19] "  0.0100069895963854"   "  0.0130828635756801"   "  0.0366247913546242"
## [22] "  0.0387678537469631"   "  0.00695432838934901"  "  0.0353306249714557"
## [25] "  0.0281168516925988"   "  0.0114250840290401"   "  0.0290029312432826"
## [28] "  0.0348184179084519"   "  0.00508300597160924"  "  0.0112218475088008"
## [31] "  0.0137259580429104"   "  0.000201797992751805" "  0.00818581844397517"
## [34] "  0.0333355616245486"   "  0.00129193745205192"  "  0.0349366558652423"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "-------------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=5) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:454350.326116661$"
## [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] "HON"  "UPS"  "ETN"  "BA"   "ADP"  "DE"   "ELV"  "PFE"  "GILD" "JNJ"
## [11] "MDT"  "LLY"  "INTU" "AVGO" "TXN"  "ORCL" "NVDA" "CSCO" "TMUS" "META"
## [21] "VZ"   "ATVI" "NFLX" "CHTR" "MMC"  "SPGI" "AXP"  "SCHW" "MA"   "JPM"
## [31] "SBUX" "TSLA" "F"    "AZO"  "GM"   "ABNB"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.00101574574301197" "  0.0548482207928486"  "  0.0432220049482565"
##  [4] "  0.0214153699226716"  "  0.0617330236401901"  "  0.0361458190768657"
##  [7] "  0.0174070444655022"  "  0.0615342548627835"  "  0.0236319897524712"
## [10] "  0.0207626416266238"  "  0.0441346071834107"  "  0.0293795684248937"
## [13] "  0.0105845130600178"  "  0.0400833335431165"  "  0.0364906546527535"
## [16] "  0.0235908592720511"  "  0.0385124571948957"  "  0.0222464030373761"
## [19] "  0.0147359996471523"  "  0.00446017554643796" "  0.0601024955069976"
## [22] "  0.0183244599691058"  "  0.0520314887981228"  "  0.0634920040939522"
## [25] "  0.0181818242710972"  "  0.00804363253675548" "  0.00524547675066786"
## [28] "  0.0120456016501699"  "  0.012155227079791"   "  0.0549648797522819"
## [31] "  0.0412684555970201"  "  0.00835236481726289" "  0.0479220010408734"
## [34] "  0.0140148852626642"  "  0.0604889588326591"  "  0.0631096534557047"
## [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:

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-11-02       232.4444              1      0.009781442      0.012306041
## 2022-11-09       235.3226              1      0.012382070      0.053616027
## 2022-11-16       248.2840              1      0.055079400      0.034718645
## 2022-11-23       257.0555              1      0.035328370      0.005923635
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3       atr
## 2022-11-02   0.009733913   0.008113074   0.039930970  -0.064535796  9.885942
## 2022-11-09   0.012306041   0.009733913   0.008113074   0.039930970  9.762661
## 2022-11-16   0.053616027   0.012306041   0.009733913   0.008113074 10.232471
## 2022-11-23   0.034718645   0.053616027   0.012306041   0.009733913 10.243009
## 2022-11-30   0.005923635   0.034718645   0.053616027   0.012306041 10.247795
##                 adx aaron        bb chaikin_vol        clv        emv     macd
## 2022-11-02 13.58997   100 0.6303335  2.90314600 -0.2863719 0.02711271 1.939312
## 2022-11-09 13.77107    50 0.6307783 -0.09676625 -0.3920529 0.04765004 1.866926
## 2022-11-16 14.68326   100 0.8325740 -0.38397100 -0.4461119 0.09074850 1.906715
## 2022-11-23 15.95273   100 0.9310325 -0.20180520 -0.3205142 0.11758529 2.068291
## 2022-11-30 16.53998   100 0.8907336  0.48394890 -0.1089895 0.12144667 2.300754
##                 mfi      sar       smi     volat month_index
## 2022-11-02 49.23300 258.6055  5.546375 0.2606250          83
## 2022-11-09 49.20839 257.2257  3.943960 0.2653165          83
## 2022-11-16 48.83463 256.7200  6.291102 0.2641173          83
## 2022-11-23 49.31528 224.1100 11.099826 0.2624611          83
## 2022-11-30 42.97382 224.1100 16.713518 0.2759187          83
```

The follwoing function extracts the specific window

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

# test out for a sample run
tau = 10 # 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
## 2016-10-05             -1  0.18091008 0.10247324          10
## 2016-10-12              1  0.24064338 0.10506831          10
## 2016-10-19             -1  0.09899013 0.10335977          10
## 2016-10-26              1 -0.01496489 0.09985285          10
## 2016-11-02              1  0.05096933 0.13389984          11
## 2016-11-09              1  0.19338517 0.16512456          11
## 2016-11-16              1  0.32341865 0.17462225          11
## 2016-11-23             -1  0.15097908 0.17528980          11
## 2016-11-30              1 -0.05591444 0.17727467          11
## 2016-12-07              1  0.11324740 0.17572623          12
```

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

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

# display
tail(sample_xts_with_garch, 3)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-11-16       248.2840              1      0.055079400      0.034718645
## 2022-11-23       257.0555              1      0.035328370      0.005923635
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2022-11-16   0.053616027    0.01230604   0.009733913   0.008113074 10.23247
## 2022-11-23   0.034718645    0.05361603   0.012306041   0.009733913 10.24301
## 2022-11-30   0.005923635    0.03471865   0.053616027   0.012306041 10.24779
##                 adx aaron        bb chaikin_vol        clv       emv     macd
## 2022-11-16 14.68326   100 0.8325740  -0.3839710 -0.4461119 0.0907485 1.906715
## 2022-11-23 15.95273   100 0.9310325  -0.2018052 -0.3205142 0.1175853 2.068291
## 2022-11-30 16.53998   100 0.8907336   0.4839489 -0.1089895 0.1214467 2.300754
##                mfi    sar       smi     volat month_index vol_forecast
## 2022-11-16 48.83463 256.72  6.291102 0.2641173          83    0.2642679
## 2022-11-23 49.31528 224.11 11.099826 0.2624611          83    0.2651389
## 2022-11-30 42.97382 224.11 16.713518 0.2759187          83    0.2659892
```

```
# Example usage
sample_xts_with_arima <- f_add_arima_forecast(sample_xts_with_garch,
                                              arima_col="adjusted_close")
```

```
## Warning in value[[3L]](cond): error for SARIMA(1,0,0,0,0,1) -> skipping...
```

```
## Warning in value[[3L]](cond): error for SARIMA(1,0,0,0,1,1) -> skipping...
```

```
tail(sample_xts_with_arima)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146075      0.009733913
## 2022-11-02       232.4444              1      0.009781442      0.012306041
## 2022-11-09       235.3226              1      0.012382070      0.053616027
## 2022-11-16       248.2840              1      0.055079400      0.034718645
## 2022-11-23       257.0555              1      0.035328370      0.005923635
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3       atr
## 2022-10-26   0.008113074   0.039930970  -0.064535796   0.030150911  9.676399
## 2022-11-02   0.009733913   0.008113074   0.039930970  -0.064535796  9.885942
## 2022-11-09   0.012306041   0.009733913   0.008113074   0.039930970  9.762661
## 2022-11-16   0.053616027   0.012306041   0.009733913   0.008113074 10.232471
## 2022-11-23   0.034718645   0.053616027   0.012306041   0.009733913 10.243009
## 2022-11-30   0.005923635   0.034718645   0.053616027   0.012306041 10.247795
##                 adx aaron         bb chaikin_vol         clv         emv     macd
## 2022-10-26 13.39493   100 0.6110784 -1.49750300 -0.1320576 -0.01707202 2.049576
## 2022-11-02 13.58997   100 0.6303335  2.90314600 -0.2863719  0.02711271 1.939312
## 2022-11-09 13.77107    50 0.6307783 -0.09676625 -0.3920529  0.04765004 1.866926
## 2022-11-16 14.68326   100 0.8325740 -0.38397100 -0.4461119  0.09074850 1.906715
## 2022-11-23 15.95273   100 0.9310325 -0.20180520 -0.3205142  0.11758529 2.068291
## 2022-11-30 16.53998   100 0.8907336  0.48394890 -0.1089895  0.12144667 2.300754
##                 mfi      sar       smi     volat month_index vol_forecast
## 2022-10-26 51.52422 260.0428  8.131402 0.2269538          82    0.2624611
## 2022-11-02 49.23300 258.6055  5.546375 0.2606250          83    0.2759187
## 2022-11-09 49.20839 257.2257  3.943960 0.2653165          83    0.2633755
## 2022-11-16 48.83463 256.7200  6.291102 0.2641173          83    0.2642679
## 2022-11-23 49.31528 224.1100 11.099826 0.2624611          83    0.2651389
## 2022-11-30 42.97382 224.1100 16.713518 0.2759187          83    0.2659892
##            sarima_010_001 sarima_110_001 sarima_020_001 sarima_120_001
## 2022-10-26       257.0555       258.0605       265.8269       267.6830
## 2022-11-02       258.5827       258.7577       260.1099       263.3190
## 2022-11-09       258.5827       258.7777       261.6371       266.6338
## 2022-11-16       258.5827       258.7800       263.1643       270.5783
## 2022-11-23       258.5827       258.7803       264.6916       274.2438
## 2022-11-30       258.5827       258.7803       266.2188       278.0330
##            sarima_010_011 sarima_110_011 sarima_020_011 sarima_120_011
## 2022-10-26       257.0555       258.0605       265.8269       267.6830
## 2022-11-02       258.5827       258.7577       260.1099       263.3190
## 2022-11-09       258.5827       258.7777       261.6371       266.6338
## 2022-11-16       258.5827       258.7800       263.1643       270.5783
## 2022-11-23       258.5827       258.7803       264.6916       274.2438
## 2022-11-30       258.5827       258.7803       266.2188       278.0330
```

```
sample_xts_with_arima[, c("discrete_returns", "vol_forecast")]
```

```
##            discrete_returns vol_forecast
## 2016-01-06               NA           NA
## 2016-01-13    -0.0482404700           NA
## 2016-01-20     0.0113781400           NA
## 2016-01-27     0.0288930800           NA
## 2016-02-03     0.0207508100           NA
## 2016-02-10    -0.0160681700    0.2380100
## 2016-02-17     0.0556720700    0.2389290
## 2016-02-24    -0.0008204018    0.2214060
## 2016-03-02     0.0045743070    0.1992566
## 2016-03-09     0.0070603550    0.1872713
##        ...
## 2022-09-28     0.0066400850    0.2269538
## 2022-10-05     0.0306100500    0.2606250
```

```
## 2022-10-12    -0.0624974400    0.2653165
## 2022-10-19     0.0407389300    0.2641173
## 2022-10-26     0.0081460750    0.2624611
## 2022-11-02     0.0097814420    0.2759187
## 2022-11-09     0.0123820700    0.2633755
## 2022-11-16     0.0550794000    0.2642679
## 2022-11-23     0.0353283700    0.2651389
## 2022-11-30     0.0059412150    0.2659892
```

```
# Example usage
sample_xts_full <- f_extract_dynamic_features(sample_xts_with_garch,
                                              arima_col = "adjusted_close",
                                              volat_col = "volat")
```

```
## Warning in value[[3L]](cond): error for SARIMA(1,0,0,0,0,1) -> skipping...
```

```
## Warning in value[[3L]](cond): error for SARIMA(1,0,0,0,1,1) -> skipping...
```

```
tail(sample_xts_full)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146075      0.009733913
## 2022-11-02       232.4444              1      0.009781442      0.012306041
## 2022-11-09       235.3226              1      0.012382070      0.053616027
## 2022-11-16       248.2840              1      0.055079400      0.034718645
## 2022-11-23       257.0555              1      0.035328370      0.005923635
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3       atr
## 2022-10-26   0.008113074   0.039930970  -0.064535796   0.030150911  9.676399
## 2022-11-02   0.009733913   0.008113074   0.039930970  -0.064535796  9.885942
## 2022-11-09   0.012306041   0.009733913   0.008113074   0.039930970  9.762661
## 2022-11-16   0.053616027   0.012306041   0.009733913   0.008113074 10.232471
## 2022-11-23   0.034718645   0.053616027   0.012306041   0.009733913 10.243009
## 2022-11-30   0.005923635   0.034718645   0.053616027   0.012306041 10.247795
##                 adx aaron        bb chaikin_vol        clv         emv     macd
## 2022-10-26 13.39493   100 0.6110784 -1.49750300 -0.1320576 -0.01707202 2.049576
## 2022-11-02 13.58997   100 0.6303335  2.90314600 -0.2863719  0.02711271 1.939312
## 2022-11-09 13.77107    50 0.6307783 -0.09676625 -0.3920529  0.04765004 1.866926
## 2022-11-16 14.68326   100 0.8325740 -0.38397100 -0.4461119  0.09074850 1.906715
## 2022-11-23 15.95273   100 0.9310325 -0.20180520 -0.3205142  0.11758529 2.068291
## 2022-11-30 16.53998   100 0.8907336  0.48394890 -0.1089895  0.12144667 2.300754
##                 mfi      sar       smi     volat month_index vol_forecast
## 2022-10-26 51.52422 260.0428  8.131402 0.2269538          82    0.2624611
## 2022-11-02 49.23300 258.6055  5.546375 0.2606250          83    0.2759187
## 2022-11-09 49.20839 257.2257  3.943960 0.2653165          83    0.2633755
## 2022-11-16 48.83463 256.7200  6.291102 0.2641173          83    0.2642679
## 2022-11-23 49.31528 224.1100 11.099826 0.2624611          83    0.2651389
## 2022-11-30 42.97382 224.1100 16.713518 0.2759187          83    0.2659892
##            sarima_010_001 sarima_110_001 sarima_020_001 sarima_120_001
## 2022-10-26       257.0555       258.0605       265.8269       267.6830
## 2022-11-02       258.5827       258.7577       260.1099       263.3190
## 2022-11-09       258.5827       258.7777       261.6371       266.6338
## 2022-11-16       258.5827       258.7800       263.1643       270.5783
## 2022-11-23       258.5827       258.7803       264.6916       274.2438
## 2022-11-30       258.5827       258.7803       266.2188       278.0330
##            sarima_010_011 sarima_110_011 sarima_020_011 sarima_120_011
## 2022-10-26       257.0555       258.0605       265.8269       267.6830
## 2022-11-02       258.5827       258.7577       260.1099       263.3190
```

```
## 2022-11-09       258.5827       258.7777       261.6371       266.6338
## 2022-11-16       258.5827       258.7800       263.1643       270.5783
## 2022-11-23       258.5827       258.7803       264.6916       274.2438
## 2022-11-30       258.5827       258.7803       266.2188       278.0330
```

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_stocks <- names(sector_data)

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

  # 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 dynamic features for all stocks
  list_xts_sector <- lapply(list_xts_sector,
                            f_extract_dynamic_features,
                            arima_col = "adjusted_close",
                            volat_col = "volat"
                            )

  # 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

  # should return the list for the chosen stocks
  chosen_stocks <- sector_data[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 xts
```

```
## [1] "UNP" "ITW" "ADP" "HON" "GE"  "RTX"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-01-06       72.98834             -1               NA     -0.059310988
## 2016-01-13       68.78521             -1     -0.057586355     -0.013918571
## 2016-01-20       67.83445              1     -0.013822156      0.058148491
## 2016-01-27       71.89586              1      0.059872366      0.042801329
## 2016-02-03       75.03990              1      0.043730515      0.004877857
## 2016-02-10       75.40683              1      0.004889773      0.048674604
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3 atr adx
## 2016-01-06            NA            NA            NA            NA  NA  NA
## 2016-01-13   -0.059310988            NA            NA            NA  NA  NA
## 2016-01-20   -0.013918571   -0.05931099            NA            NA  NA  NA
## 2016-01-27    0.058148491   -0.01391857   -0.05931099            NA  NA  NA
## 2016-02-03    0.042801329    0.05814849   -0.01391857   -0.05931099  NA  NA
## 2016-02-10    0.004877857    0.04280133    0.05814849   -0.01391857  NA  NA
##            aaron bb chaikin_vol clv emv macd mfi      sar smi volat month_index
## 2016-01-06    NA NA          NA  NA  NA   NA  NA 86.02146  NA    NA           1
## 2016-01-13   -50 NA          NA  NA  NA   NA  NA 88.67000  NA    NA           1
## 2016-01-20  -100 NA          NA  NA  NA   NA  NA 88.67000  NA    NA           1
## 2016-01-27    50 NA          NA  NA  NA   NA  NA 88.37680  NA    NA           1
## 2016-02-03   100 NA          NA  NA  NA   NA  NA 81.34000  NA    NA           2
## 2016-02-10   100 NA          NA  NA  NA   NA  NA 81.34000  NA    NA           2
```

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.

```
# 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 dynamic features for all stocks
list_xts_sector <- lapply(list_xts_sector,
                          f_extract_dynamic_features,
                          arima_col = "adjusted_close",
                          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"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-10-05       75.58762             -1      0.001486852     -0.008140198
## 2016-10-12       74.97482              1     -0.008107156      0.006425973
## 2016-10-19       75.45815             -1      0.006446663     -0.002748745
## 2016-10-26       75.25102              1     -0.002744971      0.031497619
## 2016-11-02       77.65897              1      0.031998920      0.010172453
## 2016-11-09       78.45299              1      0.010224370      0.025738667
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2016-10-05   0.001485748  -0.016219884   0.024948305  -0.037026373 1.900259
## 2016-10-12  -0.008140198   0.001485748  -0.016219884   0.024948305 1.872384
## 2016-10-19   0.006425973  -0.008140198   0.001485748  -0.016219884 1.800070
## 2016-10-26  -0.002748745   0.006425973  -0.008140198   0.001485748 1.722923
## 2016-11-02   0.031497619  -0.002748745   0.006425973  -0.008140198 1.864142
## 2016-11-09   0.010172453   0.031497619  -0.002748745   0.006425973 1.989560
##                 adx aaron        bb  chaikin_vol         clv           emv
## 2016-10-05 15.44565   -50 0.2934560   -0.4622892  0.18091008 -0.0006643160
## 2016-10-12 15.23639  -100 0.2289285    0.3990933  0.24064338 -0.0026850063
```

```
## 2016-10-19 14.75791   -50 0.3060118   -0.4336751  0.09899013 -0.0019094937
## 2016-10-26 14.44363   100 0.2860935   -1.0188680 -0.01496489 -0.0021492280
## 2016-11-02 14.04553    50 0.4910556 -324.8278000  0.05096933 -0.0009225739
## 2016-11-09 13.44222   100 0.5094234    1.1391500  0.19338517 -0.0009562142
##                 macd      mfi      sar        smi      volat month_index
## 2016-10-05 1.3477744 46.50802 95.02127  -5.331162 0.10247324          10
## 2016-10-12 1.1358585 37.92195 94.68802 -11.930732 0.10506831          10
## 2016-10-19 0.9402188 36.19915 94.36810 -17.430099 0.10335977          10
## 2016-10-26 0.7585276 30.28217 94.06097 -19.828752 0.09985285          10
## 2016-11-02 0.6437468 48.88575 93.76613 -18.073978 0.13389984          11
## 2016-11-09 0.5919089 59.37208 93.48309 -13.909935 0.16512456          11
##            sarima_100_001 sarima_010_001 sarima_110_001 sarima_020_001
## 2016-10-05       75.41208       75.25102       75.26198       75.04389
## 2016-10-12       77.80647       77.65897       77.53156       80.06692
## 2016-10-19       78.59601       78.45299       78.41098       79.24701
## 2016-10-26       80.62997       80.49847       80.39024       82.54395
## 2016-11-02       83.53071       83.41564       83.26129       86.33281
## 2016-11-09       82.99002       82.87189       82.90066       82.32814
##            sarima_120_001 sarima_100_011 sarima_010_011 sarima_110_011
## 2016-10-05       75.41139       75.41208       75.25102       75.26198
## 2016-10-12       78.67503       77.80647       77.65897       77.53156
## 2016-10-19       80.10604       78.59601       78.45299       78.41098
## 2016-10-26       81.87785       80.62997       80.49847       80.39024
## 2016-11-02       85.86885       83.53071       83.41564       83.26129
## 2016-11-09       84.17024       82.99002       82.87189       82.90066
##            sarima_020_011 sarima_120_011 vol_forecast
## 2016-10-05       75.04389       75.41139    0.1338998
## 2016-10-12       80.06692       78.67503    0.1651246
## 2016-10-19       79.24701       80.10604    0.1746223
## 2016-10-26       82.54395       81.87785    0.1752898
## 2016-11-02       86.33281       85.86885    0.1772747
## 2016-11-09       82.32814       84.17024    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.

```
# 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`

```
# 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", # future-lagged log-returns
```

```
                    volat_col = "volat", # we always want to keep the volatility col
                    garch_col = "vol_forecast", # GARCH column
                    nvmax = 20, # 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:
```

```
print("")
```

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

```
best_feat_list
```

```
## $featnames
##  [1] "adjusted_close" "adjclose_lag1"  "atr"            "emv"
##  [5] "sar"            "volat"          "sarima_100_001" "sarima_010_001"
##  [9] "sarima_110_001" "sarima_020_001" "sarima_120_001" "sarima_020_011"
## [13] "vol_forecast"
##
## $fmla
## realized_returns ~ adjusted_close + adjclose_lag1 + atr + emv +
##     sar + volat + sarima_100_001 + sarima_010_001 + sarima_110_001 +
##     sarima_020_001 + sarima_120_001 + sarima_020_011 + vol_forecast
## <environment: 0x00000260a940b158>
```

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 = 0.05))  # 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", # forecast future log returns
```

```
                    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

# re-stack train and val
full_train <- rbind.xts(ticker_data_train, ticker_data_val)

# Calculate the Sharpe Ratio and MSR (on historical discrete returns)
scaling_factor <- as.vector(ticker_data_val$month_index)[1] - as.vector(ticker_data_train$month_index)[1]

# Pack returns and compute mean and std
hist_returns <- na.trim(as.vector(full_train[, "discrete_returns"]))
mean_rets <- mean(hist_returns)
std_rets <- sd(hist_returns)

# Calculate the ES and set risk-free
VaR <- quantile(hist_returns, 0.05)
ES <- mean(hist_returns[hist_returns < VaR])
Rf <- 0

# Calculate the Sharpe and MSR
stock_sharpe <- ((mean_rets- Rf)/ std_rets ) * sqrt(scaling_factor) # annualized
```

```
  stock_msr <- ((mean_rets- Rf)/ ES ) * sqrt(scaling_factor) # annualized

  ### 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("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 ~ adjusted_close + adjclose_lag1 + atr + emv +
##     sar + volat + sarima_010_001 + sarima_110_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x00000260a5952f90>
## [1] "****************************************"
## [1] "rmse:  0.0673500010500403"
## [1] "sharpe:  1.1017797869661"
## [1] "msr:  -0.443650534881901"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + sar +
##     volat + sarima_110_001 + sarima_120_011 + vol_forecast
## <environment: 0x00000260a6d5b888>
## [1] "****************************************"
## [1] "rmse:  0.0727237633718505"
## [1] "sharpe:  1.70973518970817"
## [1] "msr:  -1.04056598848848"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adx + bb +
##     clv + emv + smi + sarima_010_001 + sarima_120_001 + sarima_010_011 +
##     sarima_110_011 + sarima_020_011 + volat + vol_forecast
## <environment: 0x00000260a0da31d8>
## [1] "****************************************"
## [1] "rmse:  0.118558401232087"
## [1] "sharpe:  0.943881891705896"
## [1] "msr:  -0.50746836006855"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
```

```
##     adx + macd + sarima_020_001 + sarima_120_001 + sarima_100_011 +
##     volat + vol_forecast
## <environment: 0x00000260a4838258>
## [1] "****************************************"
## [1] "rmse:  0.0102614431530874"
## [1] "sharpe:  1.10475622320284"
## [1] "msr:  -0.804116760122263"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: DE"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + discrete_returns + atr +
##     adx + emv + mfi + sar + volat + sarima_010_001 + sarima_110_001 +
##     sarima_120_011 + vol_forecast
## <environment: 0x00000260a97259a0>
## [1] "****************************************"
## [1] "rmse:  0.0683859148216069"
## [1] "sharpe:  1.02762379773608"
## [1] "msr:  -0.52279308074409"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + aaron + sarima_020_001 + sarima_120_001 +
##     sarima_100_011 + volat + vol_forecast
## <environment: 0x00000260aa7e1e48>
## [1] "****************************************"
## [1] "rmse:  0.028952329752354"
## [1] "sharpe:  0.740520555565995"
## [1] "msr:  -0.40930720237189"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + emv + sarima_110_001 +
##     sarima_120_001 + sarima_010_011 + vol_forecast + volat
## <environment: 0x00000260a5fa0588>
## [1] "****************************************"
## [1] "rmse:  0.096584432726626"
## [1] "sharpe:  0.673510765856097"
## [1] "msr:  -0.325673553237636"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + discrete_returns + adjclose_lag0 +
##     atr + adx + bb + chaikin_vol + mfi + sar + smi + sarima_110_001 +
##     sarima_100_011 + sarima_120_011 + volat + vol_forecast
## <environment: 0x00000260a5b82800>
## [1] "****************************************"
## [1] "rmse:  0.150522202778538"
## [1] "sharpe:  -1.20827269236267"
## [1] "msr:  0.475164552513197"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + adjclose_lag0 + adjclose_lag1 +
```

```
##     adjclose_lag2 + adjclose_lag3 + atr + adx + bb + macd + mfi +
##     smi + sarima_110_011 + sarima_020_011 + sarima_120_011 +
##     volat + vol_forecast
## <environment: 0x00000260a858f420>
## [1] "****************************************"
## [1] "rmse:  0.0265834402368272"
## [1] "sharpe:  1.01565125995052"
## [1] "msr:  -0.442513328958101"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + aaron +
##     mfi + sar + sarima_100_001 + sarima_110_001 + sarima_120_001 +
##     volat + vol_forecast
## <environment: 0x00000260a8da49e8>
## [1] "****************************************"
## [1] "rmse:  0.0220822531710652"
## [1] "sharpe:  0.465646876394665"
## [1] "msr:  -0.200113286918133"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag0 +
##     adjclose_lag2 + adx + bb + chaikin_vol + clv + sar + sarima_020_001 +
##     sarima_120_001 + sarima_100_011 + vol_forecast + volat
## <environment: 0x00000260ae511b00>
## [1] "****************************************"
## [1] "rmse:  0.10855754499316"
## [1] "sharpe:  0.800549266292788"
## [1] "msr:  -0.39672651167248"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     adx + aaron + smi + volat + sarima_110_001 + sarima_020_001 +
##     sarima_120_001 + vol_forecast
## <environment: 0x00000260a82334e8>
## [1] "****************************************"
## [1] "rmse:  0.0590002304322071"
## [1] "sharpe:  0.765602768510297"
## [1] "msr:  -0.320177344716425"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag3 +
##     adx + clv + macd + sar + smi + sarima_010_001 + sarima_020_011 +
##     sarima_120_011 + vol_forecast + volat
## <environment: 0x00000260a4908948>
## [1] "****************************************"
## [1] "rmse:  0.0774771585573895"
## [1] "sharpe:  0.848489211705653"
## [1] "msr:  -0.412356334471595"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
```

```
## realized_returns ~ adjusted_close + adx + aaron + bb + chaikin_vol +
##     emv + smi + sarima_110_001 + sarima_120_001 + volat + vol_forecast
## <environment: 0x00000260ab4b3c50>
## [1] "****************************************"
## [1] "rmse:  0.0822328803204907"
## [1] "sharpe:  1.06425663441386"
## [1] "msr:  -0.574004700519673"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + adjclose_lag2 + atr + bb + chaikin_vol +
##     clv + macd + mfi + smi + volat + sarima_110_001 + sarima_020_001 +
##     sarima_120_011 + vol_forecast
## <environment: 0x00000260a27e1000>
## [1] "****************************************"
## [1] "rmse:  0.121215946886314"
## [1] "sharpe:  0.283233648015239"
## [1] "msr:  -0.115447075847568"
## [1] "****************************************"
## [1] "########################################"
```

Now that all the models have been trained and the metrics recorded, we now simply choose the top 3 stocks based on the return, and the top 3 based on the best sharpe or modified sharpe ratio.

Let's first show some values for the `sector_tracker` object:

```
names(sector_tracker)
```

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

```
names(sector_tracker[[1]])
```

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

```
sector_tracker
```

```
## $ADP
## $ADP$forecasted_ret
## [1] -0.05679376
##
## $ADP$sharpe
## [1] 1.10178
##
## $ADP$msr
## [1] -0.4436505
##
## $ADP$rmse
## [1] 0.06735
##
## $ADP$data
## [1] NA
##
##
## $BA
```

```
## $BA$forecasted_ret
## [1] -0.03679627
##
## $BA$sharpe
## [1] 1.709735
##
## $BA$msr
## [1] -1.040566
##
## $BA$rmse
## [1] 0.07272376
##
## $BA$data
## [1] NA
##
##
## $CAT
## $CAT$forecasted_ret
## [1] -0.08742809
##
## $CAT$sharpe
## [1] 0.9438819
##
## $CAT$msr
## [1] -0.5074684
##
## $CAT$rmse
## [1] 0.1185584
##
## $CAT$data
## [1] NA
##
##
## $CSX
## $CSX$forecasted_ret
## [1] -0.005070253
##
## $CSX$sharpe
## [1] 1.104756
##
## $CSX$msr
## [1] -0.8041168
##
## $CSX$rmse
## [1] 0.01026144
##
## $CSX$data
## [1] NA
##
##
## $DE
## $DE$forecasted_ret
## [1] -0.0433946
##
## $DE$sharpe
## [1] 1.027624
##
## $DE$msr
## [1] -0.5227931
##
```

```
## $DE$rmse
## [1] 0.06838591
##
## $DE$data
## [1] NA
##
##
## $ETN
## $ETN$forecasted_ret
## [1] -0.01642924
##
## $ETN$sharpe
## [1] 0.7405206
##
## $ETN$msr
## [1] -0.4093072
##
## $ETN$rmse
## [1] 0.02895233
##
## $ETN$data
## [1] NA
##
##
## $FDX
## $FDX$forecasted_ret
## [1] 0.08942451
##
## $FDX$sharpe
## [1] 0.6735108
##
## $FDX$msr
## [1] -0.3256736
##
## $FDX$rmse
## [1] 0.09658443
##
## $FDX$data
## [1] NA
##
##
## $GE
## $GE$forecasted_ret
## [1] 0.1298861
##
## $GE$sharpe
## [1] -1.208273
##
## $GE$msr
## [1] 0.4751646
##
## $GE$rmse
## [1] 0.1505222
##
## $GE$data
## [1] NA
##
##
## $HON
## $HON$forecasted_ret
```

```
## [1] -0.01786361
##
## $HON$sharpe
## [1] 1.015651
##
## $HON$msr
## [1] -0.4425133
##
## $HON$rmse
## [1] 0.02658344
##
## $HON$data
## [1] NA
##
##
## $ITW
## $ITW$forecasted_ret
## [1] 0.006359932
##
## $ITW$sharpe
## [1] 0.4656469
##
## $ITW$msr
## [1] -0.2001133
##
## $ITW$rmse
## [1] 0.02208225
##
## $ITW$data
## [1] NA
##
##
## $LMT
## $LMT$forecasted_ret
## [1] -0.08823057
##
## $LMT$sharpe
## [1] 0.8005493
##
## $LMT$msr
## [1] -0.3967265
##
## $LMT$rmse
## [1] 0.1085575
##
## $LMT$data
## [1] NA
##
##
## $NOC
## $NOC$forecasted_ret
## [1] -0.04134811
##
## $NOC$sharpe
## [1] 0.7656028
##
## $NOC$msr
## [1] -0.3201773
##
## $NOC$rmse
```

```
## [1] 0.05900023
##
## $NOC$data
## [1] NA
##
##
## $RTX
## $RTX$forecasted_ret
## [1] -0.0597347
##
## $RTX$sharpe
## [1] 0.8484892
##
## $RTX$msr
## [1] -0.4123563
##
## $RTX$rmse
## [1] 0.07747716
##
## $RTX$data
## [1] NA
##
##
## $UNP
## $UNP$forecasted_ret
## [1] -0.06688999
##
## $UNP$sharpe
## [1] 1.064257
##
## $UNP$msr
## [1] -0.5740047
##
## $UNP$rmse
## [1] 0.08223288
##
## $UNP$data
## [1] NA
##
##
## $UPS
## $UPS$forecasted_ret
## [1] 0.1056516
##
## $UPS$sharpe
## [1] 0.2832336
##
## $UPS$msr
## [1] -0.1154471
##
## $UPS$rmse
## [1] 0.1212159
##
## $UPS$data
## [1] NA
```

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 NA
## [26] 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 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 NA
## [51] NA 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

```
# 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] ""
```

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

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

```
# 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)"
```

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

```
# 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.

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

```
##  [1] "HON"   "RTX"   "ADP"   "FDX"   "CSX"   "CAT"   "ELV"   "ABBV"  "ISRG"
## [10] "JNJ"   "TMO"   "DHR"   "TXN"   "ACN"   "CRM"   "ORCL"  "ADBE"  "IBM"
## [19] "CHTR"  "GOOGL" "GOOG"  "NFLX"  "T"     "EA"    "SCHW"  "JPM"   "GS"
## [28] "V"     "MS"    "AXP"   "ABNB"  "TSLA"  "CMG"   "MAR"   "GM"    "AMZN"
```

```
portfolio$capital
```

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

```
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
```

```
print("")
```

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

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

```
##  [1] "HON"   "RTX"   "ADP"   "FDX"   "CSX"   "CAT"   "ELV"   "ABBV"  "ISRG"
## [10] "JNJ"   "TMO"   "DHR"   "TXN"   "ACN"   "CRM"   "ORCL"  "ADBE"  "IBM"
## [19] "CHTR"  "GOOGL" "GOOG"  "NFLX"  "T"     "EA"    "SCHW"  "JPM"   "GS"
## [28] "V"     "MS"    "AXP"   "ABNB"  "TSLA"  "CMG"   "MAR"   "GM"    "AMZN"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-01-06       83.12702             -1               NA     -0.051581500
## 2016-01-13       78.94791             -1     -0.050273760     -0.008896456
## 2016-01-20       78.24867              1     -0.008857000      0.007453184
## 2016-01-27       78.83405              1      0.007481029      0.052241580
## 2016-02-03       83.06194              1      0.053630250      0.004297960
## 2016-02-10       83.41971              1      0.004307210      0.033826840
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3 atr adx
## 2016-01-06            NA            NA            NA            NA  NA  NA
## 2016-01-13  -0.051581500            NA            NA            NA  NA  NA
## 2016-01-20  -0.008896456  -0.051581500            NA            NA  NA  NA
## 2016-01-27   0.007453184  -0.008896456  -0.051581500            NA  NA  NA
## 2016-02-03   0.052241580   0.007453184  -0.008896456  -0.051581500  NA  NA
## 2016-02-10   0.004297960   0.052241580   0.007453184  -0.008896456  NA  NA
##            aaron bb chaikin_vol clv emv macd mfi      sar smi volat month_index
## 2016-01-06    NA NA          NA  NA  NA   NA  NA 96.11829  NA    NA           1
## 2016-01-13   -50 NA          NA  NA  NA   NA  NA 97.72038  NA    NA           1
## 2016-01-20  -100 NA          NA  NA  NA   NA  NA 97.72038  NA    NA           1
## 2016-01-27   -50 NA          NA  NA  NA   NA  NA 97.48165  NA    NA           1
## 2016-02-03   100 NA          NA  NA  NA   NA  NA 91.75228  NA    NA           2
## 2016-02-10   100 NA          NA  NA  NA   NA  NA 91.75228  NA    NA           2
```