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. This is done automatically by `fetch_sp500_sectors.R`.

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 0.0721380790   160545598
## 2    ACN Information Technology 0.0053982462     6892028
## 3   ADBE Information Technology 0.0066170326     4980032
## 4    ADI Information Technology 0.0023836598     5478466
## 5   ADSK Information Technology 0.0012343184     2335139
## 6   AKAM Information Technology 0.0004426694     1667789
```

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 = 20, only_tickers=TRUE)
sector_list
```

```
## $Industrials
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "EMR" "ETN" "FDX" "GD"  "GE"  "HON" "ITW"
## [13] "LMT" "MMM" "NOC" "PH"  "RTX" "UNP" "UPS" "WM"
##
## $`Health Care`
##  [1] "ABBV" "ABT"  "AMGN" "BMY"  "CI"   "CVS"  "DHR"  "ELV"  "GILD" "ISRG"
## [11] "JNJ"  "LLY"  "MDT"  "MRK"  "PFE"  "REGN" "SYK"  "TMO"  "UNH"  "VRTX"
##
## $`Information Technology`
##  [1] "AAPL" "ACN"  "ADBE" "ADI"  "AMAT" "AMD"  "AVGO" "CRM"  "CSCO" "IBM"
## [11] "INTC" "INTU" "LRCX" "MSFT" "NOW"  "NVDA" "ORCL" "PANW" "QCOM" "TXN"
##
## $`Communication Services`
##  [1] "ATVI"  "CHTR"  "CMCSA" "DIS"   "EA"    "FOXA"  "GOOG"  "GOOGL" "IPG"
## [10] "LYV"   "META"  "MTCH"  "NFLX"  "NWSA"  "OMC"   "T"     "TMUS"  "TTWO"
## [19] "VZ"    "WBD"
##
## $Financials
##  [1] "AON"  "AXP"  "BAC"  "BLK"  "BX"   "C"    "CB"   "CME"  "FI"   "GS"
## [11] "ICE"  "JPM"  "MA"   "MMC"  "MS"   "PGR"  "SCHW" "SPGI" "V"    "WFC"
##
## $`Consumer Discretionary`
##  [1] "ABNB" "AMZN" "AZO"  "BKNG" "CMG"  "DHI"  "F"    "GM"   "HD"   "HLT"
## [11] "LEN"  "MAR"  "MCD"  "NKE"  "ORLY" "ROST" "SBUX" "TJX"  "TSLA" "YUM"
```

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")
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ADP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ADP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker BA,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CAT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CAT, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CSX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CSX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker DE,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker EMR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## EMR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ETN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ETN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker FDX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## FDX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker GD,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker GE,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker HON, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## HON, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ITW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ITW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LMT, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## LMT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MMM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MMM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NOC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NOC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker PH, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker PH,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker RTX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## RTX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker UNP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## UNP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker UPS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## UPS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker WM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker WM,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ABBV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BMY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## BMY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CI, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker CI,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CVS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CVS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DHR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## DHR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ELV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ELV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GILD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## GILD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## LLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MDT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MDT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MRK, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## PFE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker REGN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## REGN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker SYK, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## SYK, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TMO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## UNH, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker VRTX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## VRTX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ACN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ACN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ADI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ADI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AMAT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## AMAT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AMD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## AMD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AVGO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## AVGO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CRM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CRM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker IBM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## IBM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker INTC, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## INTC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker INTU, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## INTU, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MSFT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NOW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NOW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NVDA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ORCL, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker PANW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## PANW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker QCOM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## QCOM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TXN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TXN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ATVI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ATVI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CHTR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CHTR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CMCSA, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CMCSA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DIS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## DIS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker EA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker EA,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker FOXA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## FOXA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GOOG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## GOOG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GOOGL, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## GOOGL, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker IPG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## IPG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LYV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## LYV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## META, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MTCH, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MTCH, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NFLX, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NFLX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NWSA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NWSA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker OMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## OMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker T, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker T,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TMUS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TMUS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TTWO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TTWO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker VZ, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker VZ,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker WBD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## WBD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AON, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## AON, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker BX,
## skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CB, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker CB,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CME, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CME, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker FI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker FI,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ICE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ICE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker MA,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker MS,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker PGR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## PGR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker SCHW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## SCHW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker SPGI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## SPGI, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker V, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker V,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## WFC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ABNB, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ABNB, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AZO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## AZO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BKNG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## BKNG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CMG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## CMG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DHI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## DHI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker GM,
## skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker HLT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## HLT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LEN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## LEN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MAR, skipping...
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MAR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MCD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## MCD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## NKE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ORLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ORLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ROST, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## ROST, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## SBUX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TJX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TJX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## TSLA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker YUM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No IV data for ticker
## YUM, skipping...
```

```
# clean the environment memory
xts_fama_french <- NULL
xts_financial_ratios <- NULL
xts_realized_vol <- NULL
```

```
# 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"  "EMR" "ETN" "FDX" "GD"  "GE"  "HON" "ITW"
## [13] "LMT" "MMM" "NOC" "PH"  "RTX" "UNP" "UPS" "WM"
```

```
# access the xts of the stocks in industrials
tail(sp500_stocks$Industrials[[5]])
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       386.3109             -1      0.053484764     -0.013459947
## 2022-11-02       381.1460              1     -0.013369767      0.028455140
## 2022-11-09       392.1473              1      0.028863855      0.023174956
## 2022-11-16       401.3415              1      0.023445581      0.073784434
## 2022-11-23       432.0741              1      0.076574708      0.007922509
## 2022-11-30       435.5108             NA      0.007953976               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-10-26      0.052103492       0.02760472       0.01517504       0.03140698
## 2022-11-02     -0.013459947       0.05210349       0.02760472       0.01517504
## 2022-11-09      0.028455140      -0.01345995       0.05210349       0.02760472
## 2022-11-16      0.023174956       0.02845514      -0.01345995       0.05210349
## 2022-11-23      0.073784434       0.02317496       0.02845514      -0.01345995
## 2022-11-30      0.007922509       0.07378443       0.02317496       0.02845514
##                 atr      adx aaron        bb chaikin_vol         clv        emv
## 2022-10-26 16.65889 10.90895   100 0.9253776  -0.7033540 -0.09383278 0.02591424
## 2022-11-02 16.30325 11.44200   100 0.8612485  -3.0070669 -0.24924990 0.06672785
## 2022-11-09 16.49302 12.20740    50 0.8917193   1.1519438 -0.35705376 0.16789580
## 2022-11-16 16.13566 13.04944   100 0.8988852  -0.8350064 -0.23171407 0.20368870
## 2022-11-23 17.98311 15.00531   100 1.0842430  13.4687113 -0.21044883 0.42019450
## 2022-11-30 17.32503 16.82149    50 1.0301560  -0.4276570 -0.01897729 0.53655500
##                    macd      mfi      sar      smi  volume     volat
## 2022-10-26 -0.366656927 65.40085 317.7989 12.41054 1157500 0.2062547
## 2022-11-02  0.002997301 56.44849 322.4772 15.17568 1719300 0.2189202
## 2022-11-09  0.414252559 59.56372 328.4654 18.43638 2182800 0.2277602
## 2022-11-16  0.867010039 59.83537 336.1099 22.58421 1101600 0.2253009
## 2022-11-23  1.447660474 67.42008 344.8063 27.55272 5080300 0.2610497
## 2022-11-30  2.082816118 69.08992 359.3094 32.75519 2397200 0.2627691
##            month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2022-10-26          82          -0.0066          0.0070         0.0089
## 2022-11-02          83          -0.0267         -0.0087         0.0161
## 2022-11-09          83          -0.0225         -0.0052         0.0055
## 2022-11-16          83          -0.0103         -0.0107         0.0057
## 2022-11-23          83           0.0063         -0.0024        -0.0094
## 2022-11-30          83           0.0312         -0.0015        -0.0207
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2022-10-26          -0.0080                        0.0067        0.00011
## 2022-11-02           0.0021                        0.0105        0.00014
## 2022-11-09           0.0095                        0.0106        0.00014
## 2022-11-16           0.0119                        0.0093        0.00014
## 2022-11-23          -0.0075                       -0.0057        0.00014
## 2022-11-30          -0.0077                       -0.0141        0.00014
##            Momentum
## 2022-10-26   0.0049
## 2022-11-02   0.0216
## 2022-11-09   0.0164
## 2022-11-16   0.0269
## 2022-11-23  -0.0184
## 2022-11-30  -0.0282
```

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      0.006852956
## 2022-10-05       27.43588             -1      0.006876492     -0.042965943
## 2022-10-12       26.28204              1     -0.042055986      0.046554111
## 2022-10-19       27.53450              1      0.047654767      0.029989923
## 2022-10-26       28.37277             -1      0.030444150     -0.008377028
## 2022-11-02       28.13608              1     -0.008342039      0.031058456
## 2022-11-09       29.02365              1      0.031545802      0.059684716
## 2022-11-16       30.80866              1      0.061501820      0.026221588
## 2022-11-23       31.62720              1      0.026568398      0.022307842
## 2022-11-30       32.34066             NA      0.022558522               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-09-28     -0.053209596     -0.069267411     -0.020913290      0.007554286
## 2022-10-05      0.006852956     -0.053209596     -0.069267411     -0.020913290
## 2022-10-12     -0.042965943      0.006852956     -0.053209596     -0.069267411
## 2022-10-19      0.046554111     -0.042965943      0.006852956     -0.053209596
## 2022-10-26      0.029989923      0.046554111     -0.042965943      0.006852956
## 2022-11-02     -0.008377028      0.029989923      0.046554111     -0.042965943
## 2022-11-09      0.031058456     -0.008377028      0.029989923      0.046554111
## 2022-11-16      0.059684716      0.031058456     -0.008377028      0.029989923
## 2022-11-23      0.026221588      0.059684716      0.031058456     -0.008377028
## 2022-11-30      0.022307842      0.026221588      0.059684716      0.031058456
##                 atr      adx aaron         bb chaikin_vol         clv
## 2022-09-28 1.441481 16.24190  -100 0.04467755  2.43234200  0.21475805
## 2022-10-05 1.384232 17.10559   -50 0.13495813 -0.44268680  0.22116568
## 2022-10-12 1.379644 18.24157   -50 0.07457368  0.43839330  0.07934922
## 2022-10-19 1.394670 18.58490    50 0.23730603 -1.12835800  0.03125187
## 2022-10-26 1.398622 18.20787   100 0.36428555  0.36773750 -0.10430028
## 2022-11-02 1.385863 17.63796   100 0.36718737 -8.91414900 -0.26417408
## 2022-11-09 1.385444 17.00435    50 0.43456871 -0.08886197 -0.35167976
## 2022-11-16 1.429341 16.04316   100 0.61239403 -0.69757770 -0.28307675
## 2022-11-23 1.395102 15.54651   100 0.68335600 -2.77541900 -0.16462184
## 2022-11-30 1.369024 15.36369   100 0.70213009 -0.65517410  0.02947430
##                        emv      macd      mfi      sar       smi   volume
## 2022-09-28 -1.787304e-04 -2.031918 46.90353 34.67000 -18.01681 18306500
## 2022-10-05 -2.096124e-04 -2.290153 46.43088 34.38840 -22.89976 16028700
## 2022-10-12 -3.472192e-04 -2.649750 46.62430 34.11806 -28.89441 13763100
## 2022-10-19 -3.458817e-04 -2.983549 54.92321 33.66998 -32.89471 15446400
## 2022-10-26 -2.858648e-04 -3.232381 56.20916 33.24878 -34.78229 21083400
## 2022-11-02 -1.913069e-04 -3.420978 48.82911 32.85285 -36.26677 15289700
## 2022-11-09 -1.696224e-04 -3.505779 48.94612 32.48068 -36.24474 10546600
## 2022-11-16 -6.177828e-05 -3.415472 46.83053 32.13084 -32.84559 10016300
## 2022-11-23  6.920197e-05 -3.168499 45.87661 26.65000 -26.53377  9659000
## 2022-11-30  2.043992e-04 -2.797269 55.72098 26.65000 -18.89848 24182500
##                   volat month_index Excess_Retun_Mkt Small_minus_Big
## 2022-09-28 0.2279791          81           0.0215          0.0092
## 2022-10-05 0.2353109          82          -0.0022         -0.0037
## 2022-10-12 0.2481376          82          -0.0027          0.0002
```

```
## 2022-10-19 0.2465206          82          -0.0087         -0.0120
## 2022-10-26 0.2484444          82          -0.0066          0.0070
## 2022-11-02 0.2806964          83          -0.0267         -0.0087
## 2022-11-09 0.2819226          83          -0.0225         -0.0052
## 2022-11-16 0.2767814          83          -0.0103         -0.0107
## 2022-11-23 0.2587499          83           0.0063         -0.0024
## 2022-11-30 0.2672197          83           0.0312         -0.0015
##            High_minus_Low Robus_minus_Weak Conservative_minus_Aggressive
## 2022-09-28        -0.0033          -0.0087                        -0.0071
## 2022-10-05         0.0006           0.0035                         0.0016
## 2022-10-12         0.0002          -0.0002                         0.0001
## 2022-10-19         0.0121           0.0070                         0.0077
## 2022-10-26         0.0089          -0.0080                         0.0067
## 2022-11-02         0.0161           0.0021                         0.0105
## 2022-11-09         0.0055           0.0095                         0.0106
## 2022-11-16         0.0057           0.0119                         0.0093
## 2022-11-23        -0.0094          -0.0075                        -0.0057
## 2022-11-30        -0.0207          -0.0077                        -0.0141
##            Risk_free_rate Momentum
## 2022-09-28        0.00009  -0.0135
## 2022-10-05        0.00011   0.0049
## 2022-10-12        0.00011  -0.0060
## 2022-10-19        0.00011   0.0196
## 2022-10-26        0.00011   0.0049
## 2022-11-02        0.00014   0.0216
## 2022-11-09        0.00014   0.0164
## 2022-11-16        0.00014   0.0269
## 2022-11-23        0.00014  -0.0184
## 2022-11-30        0.00014  -0.0282
```

```
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 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 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
##
## $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 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:505338.254873641$"
## [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] "GD"    "NOC"   "LMT"   "MMM"   "RTX"   "WM"    "LLY"   "TMO"   "BMY"
## [10] "SYK"   "VRTX"  "DHR"   "ACN"   "PANW"  "MSFT"  "ADBE"  "ADI"   "ORCL"
## [19] "GOOGL" "LYV"   "IPG"   "MTCH"  "EA"    "TMUS"  "BAC"   "AON"   "MA"
```

```
## [28] "CB"    "PGR"   "BX"    "YUM"   "GM"    "ORLY"  "HLT"   "ABNB"  "MCD"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0313248825008605"  "  0.0125519215254797"  "  0.0179303525123788"
##  [4] "  0.057617453511392"   "  0.018446047633784"   "  0.0235933589641199"
##  [7] "  0.0477700904094405"  "  0.0496979587486053"  "  0.0459740667904392"
## [10] "  0.0431520412436115"  "  0.0490445727873456"  "  0.00115623201966841"
## [13] "  0.00316378157229294" "  0.0392643714721249"  "  0.033864864508441"
## [16] "  0.0229570570690634"  "  0.00959055401731707" "  0.0299339567545933"
## [19] "  0.00573026139743527" "  0.0327375680656948"  "  0.00193666182932497"
## [22] "  0.0410489697785809"  "  0.00454880366191139" "  0.0491590980529433"
## [25] "  0.0559786206700804"  "  0.047506245250449"   "  0.0264182031685066"
## [28] "  0.0108143556732222"  "  0.00562356852204401" "  0.0241471876263942"
## [31] "  0.0361559701539604"  "  0.0066576741635158"  "  0.0420810124608173"
## [34] "  0.0171868379900526"  "  0.0476871690771444"  "  0.0139624124099669"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=2) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:548619.270459363$"
## [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] "CSX"   "RTX"   "ETN"   "WM"    "PH"    "CAT"   "ABT"   "PFE"   "ISRG"
## [10] "AMGN"  "JNJ"   "VRTX"  "NOW"   "AMD"   "NVDA"  "ADI"   "ACN"   "AAPL"
## [19] "VZ"    "CHTR"  "T"     "IPG"   "GOOGL" "WBD"   "WFC"   "ICE"   "FI"
## [28] "MMC"   "BX"    "CME"   "MCD"   "NKE"   "HD"    "ORLY"  "LEN"   "GM"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0216609113045332"  "  0.0139702549438667"  "  0.0352278176591832"
##  [4] "  0.0521391213534231"  "  0.0126608819018614"  "  0.0251405798930005"
##  [7] "  0.0563396939007866"  "  0.0471597195299122"  "  0.052555767057547"
## [10] "  0.00376436966453755" "  0.0308206421150817"  "  0.00651846792691254"
## [13] "  0.0302761448181505"  "  0.0406673171479236"  "  0.00900440272669749"
## [16] "  0.0145022609946093"  "  0.0495786635983077"  "  0.0516027677551046"
## [19] "  0.0019031792019598"  "  0.0269809516966921"  "  0.0539064950674996"
## [22] "  0.0163794589002182"  "  0.0476679807459897"  "  0.0242584584801801"
## [25] "  0.0445307104841346"  "  0.0566987060192553"  "  0.0458015347249237"
## [28] "  0.0228192044116687"  "  0.0332086147082977"  "  0.032013477926826"
## [31] "  0.0266550725813697"  "  0.0523296017294936"  "  0.014212275331432"
## [34] "  0.0428105352431287"  "  0.0180264473722788"  "  0.0334710019441724"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "----------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=3) ###"
## [1] "###############"
## [1] "CLOSE all positions"
```

```
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:603472.302850251$"
## [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] "WM"   "DE"   "ITW"  "UNP"  "NOC"  "EMR"  "MRK"  "PFE"  "SYK"  "ABBV"
## [11] "ELV"  "VRTX" "LRCX" "TXN"  "PANW" "AVGO" "INTC" "ORCL" "IPG"  "LYV"
## [21] "OMC"  "WBD"  "EA"   "MTCH" "BX"   "BLK"  "PGR"  "CB"   "V"    "MA"
## [31] "LEN"  "NKE"  "SBUX" "TSLA" "ABNB" "YUM"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0263413532762013"  "  0.0175325232061175"  "  0.0179392253243826"
##  [4] "  0.0103093750225764"  "  0.0400738663829502"  "  0.00743496040558757"
##  [7] "  0.0325243607109229"  "  0.0261769945276913"  "  0.0304798498117514"
## [10] "  0.0292401335832399"  "  0.035465118616177"   "  0.0165392206900433"
## [13] "  0.0240045982973226"  "  0.012360573952975"   "  0.0277715323028652"
## [16] "  0.00415938333534741" "  0.0325922098328024"  "  0.0399723659277854"
## [19] "  0.0253393484103103"  "  0.0470702264388273"  "  0.0121228428523178"
## [22] "  0.0297539767412965"  "  0.0412975749002662"  "  0.0337204069358843"
## [25] "  0.0180462504688656"  "  0.00377186370569232" "  0.0484412531490361"
## [28] "  0.0021187646229418"  "  0.0396382942169209"  "  0.043552617406923"
## [31] "  0.0228211719380625"  "  0.00754579562146788" "  0.015568635346927"
## [34] "  0.00544648286289685" "  0.0261566335113736"  "  0.0142107195254774"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "------------------------------------------------------------------------------------------------------------"
## [1] "###############"
## [1] "### (tau=4) ###"
## [1] "###############"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:580032.434233238$"
## [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] "WM"   "ADP"  "UPS"  "ETN"  "CAT"  "ITW"  "VRTX" "SYK"  "REGN" "ISRG"
## [11] "ELV"  "DHR"  "AMD"  "ADBE" "CSCO" "ADI"  "TXN"  "AAPL" "EA"   "OMC"
## [21] "FOXA" "VZ"   "TTWO" "CHTR" "BLK"  "CB"   "JPM"  "MS"   "FI"   "MMC"
## [31] "TJX"  "HD"   "MAR"  "YUM"  "SBUX" "MCD"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0388593840687058"   "  0.00621658046230298"  "  0.0476653558464543"
##  [4] "  0.0304894038876893"   "  0.0139894379127845"   "  0.0226291647442979"
##  [7] "  0.0073777682098491"   "  0.0357344060062508"   "  0.0105573116150207"
## [10] "  0.0406217833591008"   "  0.0347468678077991"   "  0.0465751062671547"
## [13] "  0.0247738887324821"   "  0.00771334482815055"  "  0.0285124533965843"
```

```
## [16] "  0.0376295035578642"   "  0.00320126217653165"  "  0.00355428930154462"
## [19] "  0.0147052004159125"   "  0.00779798628187835"  "  0.0258093012651675"
## [22] "  0.0307826753912053"   "  0.000382574311939216" "  0.00490891511962274"
## [25] "  0.00879938220299929"  "  0.0358730611560208"   "  0.0491260228300795"
## [28] "  0.0457560618229043"   "  0.0229735386703242"   "  0.0208836372331035"
## [31] "  0.0400862874308055"   "  0.0428365969110048"   "  0.0488028236825381"
## [34] "  0.0283782886985043"   "  0.044364055888861"    "  0.0437916958611087"
## [1] ""
## [1] "(4) LONG PORTFOLIO()"
## [1] "-------------------------------------------------------------------------------------------------------"
## [1] "################"
## [1] "### (tau=5) ###"
## [1] "################"
## [1] "CLOSE all positions"
## [1] "(1) COMPUTE_P/L(portfolio)"
## [1] "--> Capital:593617.760955192$"
## [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] "CAT"   "UNP"   "NOC"   "PH"    "MMM"   "HON"   "ISRG"  "JNJ"   "UNH"
## [10] "CVS"   "GILD"  "CI"    "NVDA"  "NOW"   "TXN"   "IBM"   "ACN"   "INTU"
## [19] "IPG"   "TTWO"  "WBD"   "CMCSA" "OMC"   "NFLX"  "BLK"   "PGR"   "MA"
## [28] "AXP"   "SPGI"  "ICE"   "MAR"   "GM"    "LEN"   "TJX"   "AMZN"  "ORLY"
## [1] ""
## [1] "(3) OPTIMIZE_PORTFOLIO(portfolio)"
## [1] "weights: "
##  [1] "  0.0431322879744082"   "  0.0289100749431143"   "  0.0337600628946881"
##  [4] "  0.0093259160896511"   "  0.0328571069827642"   "  0.01621088858498"
##  [7] "  0.0527768413458262"   "  0.0400050775075412"   "  0.00732984075364884"
## [10] "  0.0383708821299155"   "  0.0246071541072853"   "  0.0469549947082321"
## [13] "  0.0374105037609372"   "  0.0200437553833946"   "  0.0521293664266327"
## [16] "  0.0275426959191566"   "  0.0505476729879179"   "  0.00391796650583436"
## [19] "  0.00224243185420131"  "  0.0528351688356047"   "  0.0290966865847319"
## [22] "  0.0338396778051208"   "  0.0259572985053851"   "  0.0291622853255128"
## [25] "  0.026471312214402"    "  0.0411902985311203"   "  0.0411749094069736"
## [28] "  0.0227477221986397"   "  0.0166229202973263"   "  0.015663946318032"
## [31] "  0.0137146936258895"   "  0.044545014749542"    "  0.0283872279842291"
## [34] "  0.000794063499954476" "  0.00441933399251779"  "  0.0471975902934115"
## [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      0.009781376      0.012306170
## 2022-11-09       235.3226              1      0.012382200      0.053616020
## 2022-11-16       248.2840              1      0.055079400      0.034718700
## 2022-11-23       257.0555              1      0.035328430      0.005923517
## 2022-11-30       258.5827             NA      0.005941096               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-11-02      0.009733848      0.008113075      0.039930900     -0.064535660
## 2022-11-09      0.012306170      0.009733848      0.008113075      0.039930900
## 2022-11-16      0.053616020      0.012306170      0.009733848      0.008113075
## 2022-11-23      0.034718700      0.053616020      0.012306170      0.009733848
## 2022-11-30      0.005923517      0.034718700      0.053616020      0.012306170
##                  atr      adx aaron        bb chaikin_vol        clv        emv
## 2022-11-02  9.885942 13.58997   100 0.6303335  2.90314600 -0.2863719 0.02711271
## 2022-11-09  9.762661 13.77107    50 0.6307783 -0.09676625 -0.3920529 0.04765004
## 2022-11-16 10.232471 14.68326   100 0.8325740 -0.38397100 -0.4461119 0.09074850
## 2022-11-23 10.243009 15.95273   100 0.9310325 -0.20180520 -0.3205142 0.11758529
## 2022-11-30 10.247795 16.53998   100 0.8907336  0.48394890 -0.1089895 0.12144667
##                macd      mfi      sar       smi  volume     volat month_index
## 2022-11-02 1.939312 49.23300 258.6055  5.546375 1592400 0.2606250          83
## 2022-11-09 1.866926 49.20839 257.2257  3.943960 1242900 0.2653165          83
## 2022-11-16 1.906715 48.83463 256.7200  6.291102 1430800 0.2641173          83
## 2022-11-23 2.068291 49.31528 224.1100 11.099826 1386300 0.2624611          83
## 2022-11-30 2.300754 42.97382 224.1100 16.713518 4155500 0.2759187          83
##            Excess_Retun_Mkt Small_minus_Big High_minus_Low Robus_minus_Weak
## 2022-11-02          -0.0267         -0.0087         0.0161           0.0021
## 2022-11-09          -0.0225         -0.0052         0.0055           0.0095
## 2022-11-16          -0.0103         -0.0107         0.0057           0.0119
## 2022-11-23           0.0063         -0.0024        -0.0094          -0.0075
## 2022-11-30           0.0312         -0.0015        -0.0207          -0.0077
##            Conservative_minus_Aggressive Risk_free_rate Momentum
## 2022-11-02                        0.0105        0.00014   0.0216
## 2022-11-09                        0.0106        0.00014   0.0164
## 2022-11-16                        0.0093        0.00014   0.0269
## 2022-11-23                       -0.0057        0.00014  -0.0184
## 2022-11-30                       -0.0141        0.00014  -0.0282
```

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 = 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
tail(sample_xts_window[,c("direction_lead", "clv", "volat", "month_index")], 10)
```

```
##            direction_lead       clv     volat month_index
## 2018-07-25             -1 0.1811558 0.1467714          31
## 2018-08-01              1 0.2867010 0.1667055          32
## 2018-08-08              1 0.2601853 0.1653636          32
## 2018-08-15              1 0.3181427 0.1662480          32
## 2018-08-22              1 0.3526076 0.1629356          32
## 2018-08-29             -1 0.4276124 0.1629302          32
## 2018-09-05              1 0.5004016 0.1650939          33
## 2018-09-12              1 0.4764095 0.1513876          33
## 2018-09-19              1 0.3144651 0.1481179          33
## 2018-09-26              1 0.2036089 0.1538378          33
```

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)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-11-16       248.2840              1      0.055079400      0.034718700
## 2022-11-23       257.0555              1      0.035328430      0.005923517
## 2022-11-30       258.5827             NA      0.005941096               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-11-16      0.053616020       0.01230617      0.009733848      0.008113075
## 2022-11-23      0.034718700       0.05361602      0.012306170      0.009733848
## 2022-11-30      0.005923517       0.03471870      0.053616020      0.012306170
##                 atr      adx aaron        bb chaikin_vol        clv       emv
## 2022-11-16 10.23247 14.68326   100 0.8325740  -0.3839710 -0.4461119 0.0907485
## 2022-11-23 10.24301 15.95273   100 0.9310325  -0.2018052 -0.3205142 0.1175853
## 2022-11-30 10.24779 16.53998   100 0.8907336   0.4839489 -0.1089895 0.1214467
##                macd      mfi    sar       smi  volume     volat month_index
## 2022-11-16 1.906715 48.83463 256.72  6.291102 1430800 0.2641173          83
## 2022-11-23 2.068291 49.31528 224.11 11.099826 1386300 0.2624611          83
## 2022-11-30 2.300754 42.97382 224.11 16.713518 4155500 0.2759187          83
##            Excess_Retun_Mkt Small_minus_Big High_minus_Low Robus_minus_Weak
## 2022-11-16          -0.0103         -0.0107         0.0057           0.0119
## 2022-11-23           0.0063         -0.0024        -0.0094          -0.0075
## 2022-11-30           0.0312         -0.0015        -0.0207          -0.0077
##            Conservative_minus_Aggressive Risk_free_rate Momentum vol_forecast
```

```
## 2022-11-16                        0.0093        0.00014   0.0269    0.2782794
## 2022-11-23                       -0.0057        0.00014  -0.0184    0.2794421
## 2022-11-30                       -0.0141        0.00014  -0.0282    0.2805933
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146075      0.009733848
## 2022-11-02       232.4444              1      0.009781376      0.012306170
## 2022-11-09       235.3226              1      0.012382200      0.053616020
## 2022-11-16       248.2840              1      0.055079400      0.034718700
## 2022-11-23       257.0555              1      0.035328430      0.005923517
## 2022-11-30       258.5827             NA      0.005941096               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-10-26      0.008113075      0.039930900     -0.064535660      0.030150910
## 2022-11-02      0.009733848      0.008113075      0.039930900     -0.064535660
## 2022-11-09      0.012306170      0.009733848      0.008113075      0.039930900
## 2022-11-16      0.053616020      0.012306170      0.009733848      0.008113075
## 2022-11-23      0.034718700      0.053616020      0.012306170      0.009733848
## 2022-11-30      0.005923517      0.034718700      0.053616020      0.012306170
##                  atr      adx aaron        bb chaikin_vol        clv
## 2022-10-26  9.676399 13.39493   100 0.6110784 -1.49750300 -0.1320576
## 2022-11-02  9.885942 13.58997   100 0.6303335  2.90314600 -0.2863719
## 2022-11-09  9.762661 13.77107    50 0.6307783 -0.09676625 -0.3920529
## 2022-11-16 10.232471 14.68326   100 0.8325740 -0.38397100 -0.4461119
## 2022-11-23 10.243009 15.95273   100 0.9310325 -0.20180520 -0.3205142
## 2022-11-30 10.247795 16.53998   100 0.8907336  0.48394890 -0.1089895
##                     emv     macd      mfi      sar       smi  volume     volat
## 2022-10-26 -0.01707202 2.049576 51.52422 260.0428  8.131402 2942400 0.2269538
## 2022-11-02  0.02711271 1.939312 49.23300 258.6055  5.546375 1592400 0.2606250
## 2022-11-09  0.04765004 1.866926 49.20839 257.2257  3.943960 1242900 0.2653165
## 2022-11-16  0.09074850 1.906715 48.83463 256.7200  6.291102 1430800 0.2641173
## 2022-11-23  0.11758529 2.068291 49.31528 224.1100 11.099826 1386300 0.2624611
## 2022-11-30  0.12144667 2.300754 42.97382 224.1100 16.713518 4155500 0.2759187
##            month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2022-10-26          82          -0.0066          0.0070         0.0089
## 2022-11-02          83          -0.0267         -0.0087         0.0161
## 2022-11-09          83          -0.0225         -0.0052         0.0055
## 2022-11-16          83          -0.0103         -0.0107         0.0057
## 2022-11-23          83           0.0063         -0.0024        -0.0094
## 2022-11-30          83           0.0312         -0.0015        -0.0207
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2022-10-26          -0.0080                        0.0067        0.00011
## 2022-11-02           0.0021                        0.0105        0.00014
## 2022-11-09           0.0095                        0.0106        0.00014
## 2022-11-16           0.0119                        0.0093        0.00014
## 2022-11-23          -0.0075                       -0.0057        0.00014
## 2022-11-30          -0.0077                       -0.0141        0.00014
##            Momentum vol_forecast sarima_100_001 sarima_010_001 sarima_110_001
## 2022-10-26   0.0049    0.2624611    0.005473033    0.034718700     0.04342605
## 2022-11-02   0.0216    0.2759187    0.003833981    0.005923517     0.01919152
## 2022-11-09   0.0164    0.2771050    0.003715042    0.005923517     0.01307800
## 2022-11-16   0.0269    0.2782794    0.003708272    0.005923517     0.01589494
## 2022-11-23  -0.0184    0.2794421    0.003707887    0.005923517     0.01459697
## 2022-11-30  -0.0282    0.2805933    0.003707865    0.005923517     0.01519504
##            sarima_020_001 sarima_120_001 sarima_100_011 sarima_010_011
```

```
## 2022-10-26     0.01582138     0.05513158    0.005473033    0.034718700
## 2022-11-02    -0.02287167    -0.01640920    0.003833981    0.005923517
## 2022-11-09    -0.05166685    -0.04296136    0.003715042    0.005923517
## 2022-11-16    -0.08046203    -0.06675858    0.003708272    0.005923517
## 2022-11-23    -0.10925721    -0.09235454    0.003707887    0.005923517
## 2022-11-30    -0.13805240    -0.11677607    0.003707865    0.005923517
##            sarima_110_011 sarima_020_011 sarima_120_011 best_shifted_arima
## 2022-10-26     0.04342605     0.01582138     0.05513158         0.05513158
## 2022-11-02     0.01919152    -0.02287167    -0.01640920        -0.01640920
## 2022-11-09     0.01307800    -0.05166685    -0.04296136        -0.04296136
## 2022-11-16     0.01589494    -0.08046203    -0.06675858        -0.06675858
## 2022-11-23     0.01459697    -0.10925721    -0.09235454        -0.09235454
## 2022-11-30     0.01519504    -0.13805240    -0.11677607        -0.11677607
```

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

```
##            discrete_returns     volat vol_forecast
## 2016-01-06               NA        NA           NA
## 2016-01-13    -0.0482406900        NA           NA
## 2016-01-20     0.0113785000        NA           NA
## 2016-01-27     0.0288931900        NA           NA
## 2016-02-03     0.0207503600        NA           NA
## 2016-02-10    -0.0160682900        NA    0.2380100
## 2016-02-17     0.0556722000        NA    0.2389290
## 2016-02-24    -0.0008198745        NA    0.2214060
## 2016-03-02     0.0045742000        NA    0.1992566
## 2016-03-09     0.0070603540 0.2380100    0.1872713
##         ...
## 2022-09-28     0.0066400850 0.2449987    0.2269538
## 2022-10-05     0.0306100500 0.2057967    0.2606250
## 2022-10-12    -0.0624973200 0.1956467    0.2653165
## 2022-10-19     0.0407388600 0.1976342    0.2641173
## 2022-10-26     0.0081460750 0.2269538    0.2624611
## 2022-11-02     0.0097813760 0.2606250    0.2759187
## 2022-11-09     0.0123822000 0.2653165    0.2771050
## 2022-11-16     0.0550794000 0.2641173    0.2782794
## 2022-11-23     0.0353284300 0.2624611    0.2794421
## 2022-11-30     0.0059410960 0.2759187    0.2805933
```

```
# Example usage
sample_xts_full <- f_extract_dynamic_features(sample_xts_with_garch,
                                              arima_col = "realized_returns", # used as data for the ARIMA
                                              volat_col = "volat") # historical volat, used by GARCH
tail(sample_xts_full)
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146075      0.009733848
## 2022-11-02       232.4444              1      0.009781376      0.012306170
## 2022-11-09       235.3226              1      0.012382200      0.053616020
## 2022-11-16       248.2840              1      0.055079400      0.034718700
## 2022-11-23       257.0555              1      0.035328430      0.005923517
## 2022-11-30       258.5827             NA      0.005941096               NA
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2022-10-26      0.008113075      0.039930900     -0.064535660      0.030150910
## 2022-11-02      0.009733848      0.008113075      0.039930900     -0.064535660
## 2022-11-09      0.012306170      0.009733848      0.008113075      0.039930900
## 2022-11-16      0.053616020      0.012306170      0.009733848      0.008113075
## 2022-11-23      0.034718700      0.053616020      0.012306170      0.009733848
## 2022-11-30      0.005923517      0.034718700      0.053616020      0.012306170
```

```
##                  atr      adx aaron        bb chaikin_vol        clv
## 2022-10-26  9.676399 13.39493   100 0.6110784 -1.49750300 -0.1320576
## 2022-11-02  9.885942 13.58997   100 0.6303335  2.90314600 -0.2863719
## 2022-11-09  9.762661 13.77107    50 0.6307783 -0.09676625 -0.3920529
## 2022-11-16 10.232471 14.68326   100 0.8325740 -0.38397100 -0.4461119
## 2022-11-23 10.243009 15.95273   100 0.9310325 -0.20180520 -0.3205142
## 2022-11-30 10.247795 16.53998   100 0.8907336  0.48394890 -0.1089895
##                   emv     macd      mfi      sar       smi  volume     volat
## 2022-10-26 -0.01707202 2.049576 51.52422 260.0428  8.131402 2942400 0.2269538
## 2022-11-02  0.02711271 1.939312 49.23300 258.6055  5.546375 1592400 0.2606250
## 2022-11-09  0.04765004 1.866926 49.20839 257.2257  3.943960 1242900 0.2653165
## 2022-11-16  0.09074850 1.906715 48.83463 256.7200  6.291102 1430800 0.2641173
## 2022-11-23  0.11758529 2.068291 49.31528 224.1100 11.099826 1386300 0.2624611
## 2022-11-30  0.12144667 2.300754 42.97382 224.1100 16.713518 4155500 0.2759187
##            month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2022-10-26          82          -0.0066          0.0070         0.0089
## 2022-11-02          83          -0.0267         -0.0087         0.0161
## 2022-11-09          83          -0.0225         -0.0052         0.0055
## 2022-11-16          83          -0.0103         -0.0107         0.0057
## 2022-11-23          83           0.0063         -0.0024        -0.0094
## 2022-11-30          83           0.0312         -0.0015        -0.0207
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2022-10-26          -0.0080                        0.0067        0.00011
## 2022-11-02           0.0021                        0.0105        0.00014
## 2022-11-09           0.0095                        0.0106        0.00014
## 2022-11-16           0.0119                        0.0093        0.00014
## 2022-11-23          -0.0075                       -0.0057        0.00014
## 2022-11-30          -0.0077                       -0.0141        0.00014
##            Momentum vol_forecast sarima_100_001 sarima_010_001 sarima_110_001
## 2022-10-26   0.0049    0.2624611    0.005473033    0.034718700     0.04342605
## 2022-11-02   0.0216    0.2759187    0.003833981    0.005923517     0.01919152
## 2022-11-09   0.0164    0.2771050    0.003715042    0.005923517     0.01307800
## 2022-11-16   0.0269    0.2782794    0.003708272    0.005923517     0.01589494
## 2022-11-23  -0.0184    0.2794421    0.003707887    0.005923517     0.01459697
## 2022-11-30  -0.0282    0.2805933    0.003707865    0.005923517     0.01519504
##            sarima_020_001 sarima_120_001 sarima_100_011 sarima_010_011
## 2022-10-26     0.01582138     0.05513158    0.005473033    0.034718700
## 2022-11-02    -0.02287167    -0.01640920    0.003833981    0.005923517
## 2022-11-09    -0.05166685    -0.04296136    0.003715042    0.005923517
## 2022-11-16    -0.08046203    -0.06675858    0.003708272    0.005923517
## 2022-11-23    -0.10925721    -0.09235454    0.003707887    0.005923517
## 2022-11-30    -0.13805240    -0.11677607    0.003707865    0.005923517
##            sarima_110_011 sarima_020_011 sarima_120_011 best_shifted_arima
## 2022-10-26     0.04342605     0.01582138     0.05513158         0.05513158
## 2022-11-02     0.01919152    -0.02287167    -0.01640920        -0.01640920
## 2022-11-09     0.01307800    -0.05166685    -0.04296136        -0.04296136
## 2022-11-16     0.01589494    -0.08046203    -0.06675858        -0.06675858
## 2022-11-23     0.01459697    -0.10925721    -0.09235454        -0.09235454
## 2022-11-30     0.01519504    -0.13805240    -0.11677607        -0.11677607
```

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 list 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,
                          function(x, arima_col, volat_col) {
                            tryCatch({
                              f_extract_dynamic_features(x, arima_col, volat_col)
                            },
                            error = function(e){
                              warning("error with this dataframe:")
                              print(head(x))
                              print(tail(x))
                              print(colnames(x))
                              stop(e)
                            }
                            )
                          },
                          arima_col = "realized_returns",
                          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 = 10
sector_stocks_window <- SECTOR_PROCEDURE(G, tau)
```

```
## [1] "SECTOR_PROCEDURE(G=Industrials, tau=10)"
## [1] "  MODELLING_PROCEDURE(list_train_val_sector)"
```

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

```
## [1] "CAT" "ETN" "NOC" "ADP" "MMM" "UNP"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-01-06       41.75013             -1               NA      -0.05013623
## 2016-01-13       39.70854             -1      -0.04890016      -0.01917951
## 2016-01-20       38.95421              1      -0.01899676       0.03981973
## 2016-01-27       40.53665              1       0.04062317       0.05967236
## 2016-02-03       43.02920              1       0.06148870       0.02298085
## 2016-02-10       44.02950              1       0.02324695       0.04211713
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2016-01-06               NA               NA               NA               NA
## 2016-01-13      -0.05013623               NA               NA               NA
## 2016-01-20      -0.01917951      -0.05013623               NA               NA
## 2016-01-27       0.03981973      -0.01917951      -0.05013623               NA
## 2016-02-03       0.05967236       0.03981973      -0.01917951      -0.05013623
## 2016-02-10       0.02298085       0.05967236       0.03981973      -0.01917951
##            atr adx aaron bb chaikin_vol clv emv macd mfi      sar smi  volume
## 2016-01-06  NA  NA    NA NA          NA  NA  NA   NA  NA 49.91514  NA 2873000
## 2016-01-13  NA  NA   -50 NA          NA  NA  NA   NA  NA 51.29000  NA 3562300
## 2016-01-20  NA  NA  -100 NA          NA  NA  NA   NA  NA 51.29000  NA 4127000
## 2016-01-27  NA  NA    50 NA          NA  NA  NA   NA  NA 51.13880  NA 4581500
## 2016-02-03  NA  NA   100 NA          NA  NA  NA   NA  NA 47.51000  NA 7387400
## 2016-02-10  NA  NA   100 NA          NA  NA  NA   NA  NA 47.51000  NA 3496500
##            volat month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2016-01-06    NA           1          -0.0135         -0.0023         0.0000
## 2016-01-13    NA           1          -0.0267         -0.0062         0.0081
## 2016-01-20    NA           1          -0.0094          0.0173        -0.0127
## 2016-01-27    NA           1          -0.0111         -0.0042         0.0171
## 2016-02-03    NA           2           0.0046         -0.0025         0.0047
## 2016-02-10    NA           2           0.0001         -0.0021        -0.0055
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2016-01-06           0.0015                       0.0004          0e+00
## 2016-01-13           0.0040                       0.0063          0e+00
## 2016-01-20           0.0008                      -0.0052          0e+00
## 2016-01-27          -0.0013                       0.0092          0e+00
## 2016-02-03           0.0041                       0.0032          1e-05
## 2016-02-10          -0.0030                      -0.0069          1e-05
##            Momentum
## 2016-01-06   0.0192
## 2016-01-13   0.0016
## 2016-01-20  -0.0011
## 2016-01-27  -0.0048
## 2016-02-03  -0.0241
## 2016-02-10   0.0065
```

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 = "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"  "EMR" "ETN" "FDX" "GD"  "GE"  "HON" "ITW"
## [13] "LMT" "MMM" "NOC" "PH"  "RTX" "UNP" "UPS" "WM"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2018-08-22       128.5183              1      0.014584470      0.021061120
## 2018-08-29       131.2537             -1      0.021284470     -0.001436072
```

```
## 2018-09-05       131.0653             1     -0.001435041      0.006751360
## 2018-09-12       131.9532             1      0.006774202      0.003612816
## 2018-09-19       132.4308             1      0.003619350      0.017870280
## 2018-09-26       134.8186             1      0.018030910      0.013080880
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2018-08-22      0.014479140      0.024512530      0.038535930     -0.048139710
## 2018-08-29      0.021061120      0.014479140      0.024512530      0.038535930
## 2018-09-05     -0.001436072      0.021061120      0.014479140      0.024512530
## 2018-09-12      0.006751360     -0.001436072      0.021061120      0.014479140
## 2018-09-19      0.003612816      0.006751360     -0.001436072      0.021061120
## 2018-09-26      0.017870280      0.003612816      0.006751360     -0.001436072
##                 atr      adx aaron        bb   chaikin_vol       clv
## 2018-08-22 3.974683 29.53603   100 0.8527560  -5.811449000 0.3526076
## 2018-08-29 3.932205 30.10448   100 0.8897737  -0.787771100 0.4276124
## 2018-09-05 3.787762 30.63233    50 0.8649142  -2.118671000 0.5004016
## 2018-09-12 3.609350 31.25881   100 0.8774570 -12.753300000 0.4764095
## 2018-09-19 3.451540 31.90629   100 0.8750861   0.001593804 0.3144651
## 2018-09-26 3.491429 32.87097   100 0.9393049  -0.631312300 0.2036089
##                      emv     macd      mfi      sar      smi  volume     volat
## 2018-08-22 0.0004694374 4.401617 73.87555 135.0437 58.86006 1355200 0.1629356
## 2018-08-29 0.0047588354 4.511845 74.67040 136.5874 61.26690 2282800 0.1629302
## 2018-09-05 0.0123150419 4.622043 68.37961 138.6059 63.54092 1966900 0.1650939
## 2018-09-12 0.0130281368 4.725806 68.14414 140.2207 65.87340 1484600 0.1513876
## 2018-09-19 0.0123150585 4.806206 66.13959 141.8068 67.78665 1206000 0.1481179
## 2018-09-26 0.0145167999 4.885558 64.77485 143.2835 69.52159 1723600 0.1538378
##            month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2018-08-22          32           0.0005          0.0029        -0.0035
## 2018-08-29          32           0.0056         -0.0014        -0.0058
## 2018-09-05          33          -0.0041         -0.0004         0.0066
## 2018-09-12          33           0.0003         -0.0012        -0.0023
## 2018-09-19          33           0.0006         -0.0050         0.0128
## 2018-09-26          33          -0.0040         -0.0048        -0.0063
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2018-08-22          -0.0050                       -0.0027          7e-05
## 2018-08-29          -0.0015                       -0.0028          7e-05
## 2018-09-05           0.0046                        0.0059          8e-05
## 2018-09-12           0.0004                        0.0054          8e-05
## 2018-09-19          -0.0045                       -0.0001          8e-05
## 2018-09-26           0.0048                        0.0011          8e-05
##            Momentum sarima_100_001 sarima_010_001 sarima_110_001 sarima_020_001
## 2018-08-22   0.0097    0.005971989    0.003612816    0.005310459    0.000474272
## 2018-08-29   0.0044    0.004497078    0.017870280    0.010158398    0.032127744
## 2018-09-05  -0.0148    0.004992534    0.013080880    0.015671473    0.008291480
## 2018-09-12  -0.0086    0.005829260    0.013080880    0.014270218    0.003502080
## 2018-09-19  -0.0110    0.005742702    0.013080880    0.015028159   -0.001287320
## 2018-09-26   0.0012    0.005751656    0.013080880    0.014618187   -0.006076720
##            sarima_120_001 sarima_100_011 sarima_010_011 sarima_110_011
## 2018-08-22    0.008570346    0.005971989    0.003612816    0.005310459
## 2018-08-29    0.019692669    0.004497078    0.017870280    0.010158398
## 2018-09-05    0.021906626    0.004992534    0.013080880    0.015671473
## 2018-09-12    0.020999946    0.005829260    0.013080880    0.014270218
## 2018-09-19    0.027050232    0.005742702    0.013080880    0.015028159
## 2018-09-26    0.028127516    0.005751656    0.013080880    0.014618187
##            sarima_020_011 sarima_120_011 best_shifted_arima vol_forecast
## 2018-08-22    0.000474272    0.008570346        0.008570346    0.1481179
## 2018-08-29    0.032127744    0.019692669        0.019692669    0.1538378
## 2018-09-05    0.008291480    0.021906626        0.021906626    0.1580478
## 2018-09-12    0.003502080    0.020999946        0.020999946    0.1613970
## 2018-09-19   -0.001287320    0.027050232        0.027050232    0.1640755
## 2018-09-26   -0.006076720    0.028127516        0.028127516    0.1662261
```

```
# save data in tests
save(list_xts_sector, file = here("tests","jair", "sample_data.rda"))
```

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.

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 = 25, # maximum number of subsets to examine
                    method="backward") #  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, : 6 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
print("")
```

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

```
best_feat_list
```

```
## $featnames
##  [1] "adjusted_close"                "direction_lead"
##  [3] "log_returns_lag0"              "log_returns_lag1"
##  [5] "log_returns_lag2"              "log_returns_lag3"
##  [7] "clv"                           "emv"
##  [9] "macd"                          "mfi"
## [11] "smi"                           "volume"
## [13] "Conservative_minus_Aggressive" "sarima_110_001"
## [15] "sarima_020_001"                "sarima_120_001"
## [17] "vol_forecast"                  "volat"
##
## $fmla
## realized_returns ~ adjusted_close + direction_lead + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag2 + log_returns_lag3 +
##     clv + emv + macd + mfi + smi + volume + Conservative_minus_Aggressive +
##     sarima_110_001 + sarima_020_001 + sarima_120_001 + vol_forecast +
##     volat
## <environment: 0x000002acc54a44d8>
```

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 = 0.01))  # 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"  "EMR" "ETN" "FDX" "GD"  "GE"  "HON" "ITW"
## [13] "LMT" "MMM" "NOC" "PH"  "RTX" "UNP" "UPS" "WM"
```

```
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

```
library("glmnet")

system.time(
  # 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

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

##############################################################################

### 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="backward")

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)
```

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

# Subset features and targets for retraining
X_train <- model.matrix(best_feat_list$fmla, data = ticker_data_train)
X_test <- model.matrix(best_feat_list$fmla, data = ticker_data_val)
y_train <- ticker_data_train[, "realized_returns"]

# refit the model and assign test
refitted_model <- glmnet(X_train, y_train, alpha = best_alpha, lambda = best_lambda, standardize = TRUE)

# Use the best-fitted elastic net regression model to make predictions on the val_data
pred_enet_best <- predict(refitted_model, newx = X_test, s = refitted_model$lambda, type = "response")
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 (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 <- 0.0002 # 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 = full_train[, c("realized_returns",
                                               "best_shifted_arima",
                                               "volat",
                                               "vol_forecast")] # features to be kept

# show values
print("****************************************")
print(paste("forecasted_ret: ", 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 ~ adjusted_close + direction_lead + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag2 + log_returns_lag3 +
##     clv + emv + macd + mfi + smi + volume + Conservative_minus_Aggressive +
##     sarima_100_001 + sarima_110_001 + sarima_120_001 + vol_forecast +
##     volat
## <environment: 0x000002acc22f0650>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00555963317353535"
## [1] "rmse:  0.00730293586591441"
## [1] "sharpe:  1.06497763002774"
## [1] "msr:  -0.428831547803751"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adx + chaikin_vol +
##     clv + macd + mfi + sar + smi + volume + Excess_Retun_Mkt +
##     Conservative_minus_Aggressive + Risk_free_rate + sarima_100_001 +
##     sarima_110_001 + sarima_120_001 + vol_forecast + volat
## <environment: 0x000002acc4b4e970>
## [1] "****************************************"
## [1] "forecasted_ret:  0.0102809865145825"
## [1] "rmse:  0.0336232291758069"
## [1] "sharpe:  1.67821518062612"
## [1] "msr:  -1.02138321053705"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + log_returns_lag2 +
##     atr + adx + chaikin_vol + clv + emv + sar + smi + volat +
##     Excess_Retun_Mkt + Small_minus_Big + Robus_minus_Weak + Risk_free_rate +
##     sarima_110_001 + vol_forecast
## <environment: 0x000002acbffb91c8>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00858315252959075"
## [1] "rmse:  0.0285445097652689"
## [1] "sharpe:  0.915434655211531"
## [1] "msr:  -0.492173688074923"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     log_returns_lag0 + atr + adx + aaron + macd + sar + volume +
##     volat + Excess_Retun_Mkt + Small_minus_Big + Risk_free_rate +
##     Momentum + sarima_100_001 + sarima_110_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x000002acc2dc6ac0>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00923644257979798"
## [1] "rmse:  0.00993854792421411"
## [1] "sharpe:  1.08253418347485"
## [1] "msr:  -0.787941417977226"
```

```
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: DE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + clv + smi + volat +
##     Excess_Retun_Mkt + Small_minus_Big + Momentum + sarima_110_001 +
##     vol_forecast
## <environment: 0x000002acc2df69a0>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00571415043535353"
## [1] "rmse:  0.0235264722434734"
## [1] "sharpe:  0.996370534901365"
## [1] "msr:  -0.506893747178711"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: EMR"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     log_returns_lag1 + log_returns_lag3 + adx + clv + mfi + smi +
##     volume + volat + Excess_Retun_Mkt + Robus_minus_Weak + Conservative_minus_Aggressive +
##     Risk_free_rate + Momentum + sarima_110_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x000002acd0308548>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00275910222195631"
## [1] "rmse:  0.0126452006674113"
## [1] "sharpe:  0.805873423340971"
## [1] "msr:  -0.447114166747488"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adx + clv +
##     emv + macd + mfi + sar + volat + Momentum + sarima_100_001 +
##     sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000002accf4025e0>
## [1] "****************************************"
## [1] "forecasted_ret:  -0.0465963650291186"
## [1] "rmse:  0.0573840413673677"
## [1] "sharpe:  0.700964641929398"
## [1] "msr:  -0.387443199731368"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + adx +
##     bb + clv + mfi + sar + Excess_Retun_Mkt + Risk_free_rate +
##     sarima_110_001 + vol_forecast + volat
## <environment: 0x000002acb8b2e950>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00275803076203111"
## [1] "rmse:  0.0277911029266135"
## [1] "sharpe:  0.636040331038792"
## [1] "msr:  -0.307554568206937"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: GD"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag2 + atr + adx + aaron +
```

```
##     emv + mfi + sar + smi + volat + Excess_Retun_Mkt + Small_minus_Big +
##     Risk_free_rate + Momentum + sarima_100_001 + sarima_110_001 +
##     sarima_120_001 + vol_forecast
## <environment: 0x000002acc14f5570>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00265549675454546"
## [1] "rmse:  0.0214494878172814"
## [1] "sharpe:  0.597081362853966"
## [1] "msr:  -0.292821691321976"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + discrete_returns + mfi +
##     smi + volat + Small_minus_Big + sarima_120_001 + vol_forecast
## <environment: 0x000002accd984338>
## [1] "****************************************"
## [1] "forecasted_ret:  -0.00526787810385783"
## [1] "rmse:  0.0776115930561821"
## [1] "sharpe:  -1.23792027883352"
## [1] "msr:  0.4868235382397"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     log_returns_lag0 + atr + adx + bb + clv + emv + macd + mfi +
##     sar + smi + volume + volat + Robus_minus_Weak + Risk_free_rate +
##     sarima_100_001 + sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000002accaa5f0c8>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00383980331454545"
## [1] "rmse:  0.00674116040577253"
## [1] "sharpe:  0.964916696797536"
## [1] "msr:  -0.420408623572661"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + discrete_returns + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag2 + atr + adx + clv + emv +
##     macd + mfi + sar + smi + volat + Excess_Retun_Mkt + Small_minus_Big +
##     High_minus_Low + Risk_free_rate + sarima_100_001 + sarima_110_001 +
##     sarima_120_001 + vol_forecast
## <environment: 0x000002acbdb1b480>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00207221342727273"
## [1] "rmse:  0.0223991034721256"
## [1] "sharpe:  0.425629793854526"
## [1] "msr:  -0.182915387977227"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     log_returns_lag0 + log_returns_lag1 + log_returns_lag3 +
##     chaikin_vol + emv + macd + mfi + sar + smi + volume + volat +
##     High_minus_Low + Conservative_minus_Aggressive + sarima_100_001 +
##     sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000002acbe904f98>
```

```
## [1] "****************************************"
## [1] "forecasted_ret:  0.00360130020949495"
## [1] "rmse:  0.0206388219498158"
## [1] "sharpe:  0.760680242416787"
## [1] "msr:  -0.376968109805499"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: MMM"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + discrete_returns + log_returns_lag0 +
##     log_returns_lag2 + aaron + bb + clv + emv + sar + volume +
##     Excess_Retun_Mkt + Small_minus_Big + Risk_free_rate + Momentum +
##     sarima_100_001 + sarima_110_001 + sarima_120_001 + vol_forecast +
##     volat
## <environment: 0x000002accc9054c0>
## [1] "****************************************"
## [1] "forecasted_ret:  0.0025758053630303"
## [1] "rmse:  0.0225802734651489"
## [1] "sharpe:  0.447621471833301"
## [1] "msr:  -0.176995759691072"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + adx +
##     aaron + mfi + sar + smi + Excess_Retun_Mkt + Small_minus_Big +
##     Conservative_minus_Aggressive + Momentum + sarima_100_001 +
##     sarima_110_001 + sarima_120_001 + vol_forecast + volat
## <environment: 0x000002acccace860>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00688158657925475"
## [1] "rmse:  0.0133775499223356"
## [1] "sharpe:  0.729139959398044"
## [1] "msr:  -0.304928501534196"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: PH"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adx + bb + clv + macd + volume + Excess_Retun_Mkt + Risk_free_rate +
##     sarima_110_001 + sarima_120_001 + vol_forecast + volat
## <environment: 0x000002acc9099d70>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00357773505268912"
## [1] "rmse:  0.0274811671757754"
## [1] "sharpe:  0.732512891060597"
## [1] "msr:  -0.348823420007978"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     log_returns_lag0 + log_returns_lag1 + log_returns_lag3 +
##     atr + adx + aaron + bb + chaikin_vol + macd + smi + volat +
##     Excess_Retun_Mkt + Small_minus_Big + Robus_minus_Weak + Risk_free_rate +
##     sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000002acbdc5b1c8>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00335109503002554"
## [1] "rmse:  0.0233786468241116"
```

```
## [1] "sharpe:  0.802363376862245"
## [1] "msr:  -0.389938815283151"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag2 + log_returns_lag3 +
##     atr + adx + aaron + clv + emv + macd + smi + volat + Excess_Retun_Mkt +
##     Small_minus_Big + Conservative_minus_Aggressive + Risk_free_rate +
##     sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000002acc9a209d8>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00496811061919192"
## [1] "rmse:  0.0172740980664601"
## [1] "sharpe:  1.02821714314447"
## [1] "msr:  -0.554567567512559"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + log_returns_lag0 +
##     log_returns_lag1 + log_returns_lag3 + atr + adx + bb + clv +
##     emv + macd + sar + smi + volume + volat + Robus_minus_Weak +
##     Conservative_minus_Aggressive + sarima_100_001 + sarima_110_001 +
##     sarima_120_001 + vol_forecast
## <environment: 0x000002acc2416808>
## [1] "****************************************"
## [1] "forecasted_ret:  0.0019592046"
## [1] "rmse:  0.0243204882422511"
## [1] "sharpe:  0.248065854540176"
## [1] "msr:  -0.101112716036741"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: WM"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + discrete_returns + log_returns_lag0 +
##     adx + aaron + bb + chaikin_vol + clv + emv + macd + sar +
##     smi + volume + High_minus_Low + Conservative_minus_Aggressive +
##     sarima_100_001 + sarima_110_001 + sarima_120_001 + vol_forecast +
##     volat
## <environment: 0x000002acc157f418>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00214625276044243"
## [1] "rmse:  0.0105339024000658"
## [1] "sharpe:  0.96316624068571"
## [1] "msr:  -0.471664200184247"
## [1] "****************************************"
## [1] "########################################"

##    user  system elapsed
##   61.97    0.38   75.81
```

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"  "EMR" "ETN" "FDX" "GD"  "GE"  "HON" "ITW"
## [13] "LMT" "MMM" "NOC" "PH"  "RTX" "UNP" "UPS" "WM"
```

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

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

```
source(here("functions","modelling.R"))

# Obtain the top picks with the function
best_sector_stocks <- f_select_top_stocks(sector_tracker, n=3)
names(best_sector_stocks)
```

```
## [1] "BA"  "CSX" "ADP" "CAT"
```

```
best_sector_stocks
```

```
## $BA
## $BA$forecasted_ret
## [1] 0.01028099
##
## $BA$sharpe
## [1] 1.678215
##
## $BA$msr
## [1] -1.021383
##
## $BA$rmse
## [1] 0.03362323
##
## $BA$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05    -0.0112018172       -0.036045213 0.1215014    0.2190119
## 2016-10-12     0.0224263612       -0.006408496 0.1234677    0.2294365
## 2016-10-19     0.0664732141        0.048285037 0.1197529    0.2335310
## 2016-10-26    -0.0334655298        0.007057145 0.2165795    0.2328868
## 2016-11-02     0.0380187203        0.008819051 0.2190119    0.2368343
## 2016-11-09     0.0092616960        0.017965628 0.2294365    0.2391726
## 2016-11-16     0.0222846510        0.007418166 0.2335310    0.2369913
## 2016-11-23     0.0054611936        0.010411607 0.2328868    0.2381453
## 2016-11-30     0.0234997856       -0.006298071 0.2368343    0.2360335
## 2016-12-07     0.0021383860        0.005050070 0.2391726    0.1529022
##        ...
## 2018-07-25    -0.0089178466        0.065700917 0.2219085    0.2383571
## 2018-08-01    -0.0142179259        0.048047263 0.2317340    0.2350262
## 2018-08-08    -0.0422291468       -0.049422510 0.2311597    0.2307443
## 2018-08-15     0.0536069400        0.021594606 0.2433951    0.2134922
## 2018-08-22     0.0004569927        0.057101669 0.2383571    0.2207011
## 2018-08-29    -0.0100738217       -0.002107653 0.2350262    0.2103641
## 2018-09-05     0.0192268721        0.072968420 0.2307443    0.2101598
## 2018-09-12     0.0328710043        0.122334956 0.2134922    0.2099818
## 2018-09-19    -0.0005203785        0.138944040 0.2207011    0.2098267
## 2018-09-26     0.0720473115        0.177702804 0.2103641    0.2096916
##
```

```
##
## $CSX
## $CSX$forecasted_ret
## [1] 0.009236443
##
## $CSX$sharpe
## [1] 1.082534
##
## $CSX$msr
## [1] -0.7879414
##
## $CSX$rmse
## [1] 0.009938548
##
## $CSX$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05    -0.0164158336      -0.0163787535 0.1525324    0.1623525
## 2016-10-12     0.0280700058       0.1450202544 0.1445769    0.2170363
## 2016-10-19    -0.0224582118       0.0368837911 0.1564746    0.2146174
## 2016-10-26     0.0117805359      -0.0327851305 0.1562740    0.2039793
## 2016-11-02     0.0972597870       0.0526981183 0.1623525    0.2093311
## 2016-11-09    -0.0002950762       0.0372368137 0.2170363    0.2207179
## 2016-11-16     0.0308163334      -0.0273183512 0.2146174    0.2191759
## 2016-11-23     0.0300095227      -0.0342928663 0.2039793    0.2193127
## 2016-11-30     0.0361977282       0.0139516690 0.2093311    0.2120542
## 2016-12-07    -0.0176609459       0.0036564661 0.2207179    0.2157915
##        ...
## 2018-07-25    -0.0046619281      -0.0059180440 0.2144367    0.2012677
## 2018-08-01     0.0268245202       0.0159481565 0.2152854    0.1948769
## 2018-08-08     0.0085109320      -0.0038759344 0.2122352    0.2010700
## 2018-08-15     0.0077607768      -0.0154303986 0.2063714    0.1845599
## 2018-08-22     0.0146756166      -0.0006111292 0.2012677    0.1781337
## 2018-08-29    -0.0040298895      -0.0009078430 0.1948769    0.1537379
## 2018-09-05    -0.0021556315       0.0194216653 0.2010700    0.1624511
## 2018-09-12    -0.0020252107       0.0323569623 0.1845599    0.1698206
## 2018-09-19    -0.0012170981       0.0392889188 0.1781337    0.1761119
## 2018-09-26     0.0146418325       0.0506400104 0.1537379    0.1815207
##
##
## $ADP
## $ADP$forecasted_ret
## [1] 0.005559633
##
## $ADP$sharpe
## [1] 1.064978
##
## $ADP$msr
## [1] -0.4288315
##
## $ADP$rmse
## [1] 0.007302936
##
## $ADP$data
##            realized_returns best_shifted_arima      volat vol_forecast
## 2016-10-05     -0.008139995       3.470576e-02 0.10247324    0.1338998
## 2016-10-12      0.006425770       2.857093e-02 0.10506831    0.1651246
## 2016-10-19     -0.002748644       1.493389e-02 0.10335977    0.1746223
## 2016-10-26      0.031497620       4.953651e-02 0.09985285    0.1752898
## 2016-11-02      0.010172360      -1.150857e-02 0.13389984    0.1772747
## 2016-11-09      0.025738570      -2.227795e-05 0.16512456    0.1757262
```

```
## 2016-11-16      0.035597800       1.569318e-02 0.17462225    0.1786333
## 2016-11-23     -0.006539587       4.621815e-02 0.17528980    0.1788125
## 2016-11-30      0.021968640       2.689968e-02 0.17727467    0.1800747
## 2016-12-07      0.001229222      -2.413688e-02 0.17572623    0.1792691
##        ...
## 2018-07-25     -0.048139710       1.593597e-03 0.14677140    0.1629356
## 2018-08-01      0.038535930       1.576604e-02 0.16670545    0.1629302
## 2018-08-08      0.024512530      -3.146789e-03 0.16536364    0.1650939
## 2018-08-15      0.014479140      -6.995298e-03 0.16624802    0.1513876
## 2018-08-22      0.021061120       8.570346e-03 0.16293565    0.1481179
## 2018-08-29     -0.001436072       1.969267e-02 0.16293021    0.1538378
## 2018-09-05      0.006751360       2.190663e-02 0.16509395    0.1580478
## 2018-09-12      0.003612816       2.099995e-02 0.15138758    0.1613970
## 2018-09-19      0.017870280       2.705023e-02 0.14811786    0.1640755
## 2018-09-26      0.013080880       2.812752e-02 0.15383781    0.1662261
##
##
## $CAT
## $CAT$forecasted_ret
## [1] 0.008583153
##
## $CAT$sharpe
## [1] 0.9154347
##
## $CAT$msr
## [1] -0.4921737
##
## $CAT$rmse
## [1] 0.02854451
##
## $CAT$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05     -0.020791769       -0.064210656 0.1566718    0.1741676
## 2016-10-12      0.004784024        0.167443895 0.1514927    0.2232971
## 2016-10-19     -0.036185055        0.095848267 0.1583610    0.2283652
## 2016-10-26     -0.036556734       -0.031115941 0.1664951    0.2319594
## 2016-11-02      0.117248536       -0.013972325 0.1741676    0.2346487
## 2016-11-09      0.023301071        0.002077006 0.2232971    0.2272987
## 2016-11-16      0.029865560       -0.039248234 0.2283652    0.2279674
## 2016-11-23     -0.006467314       -0.024130169 0.2319594    0.2267511
## 2016-11-30      0.018352868        0.016032037 0.2346487    0.2293654
## 2016-12-07     -0.037582250        0.002935356 0.2272987    0.2220408
##        ...
## 2018-07-25     -0.013906324        0.049293966 0.2327273    0.2499901
## 2018-08-01      0.008410123        0.078368360 0.2445062    0.2482561
## 2018-08-08     -0.056615515       -0.044152732 0.2386050    0.2493593
## 2018-08-15      0.056042745        0.020663914 0.2547456    0.2356708
## 2018-08-22      0.015844167        0.090800736 0.2499901    0.2277853
## 2018-08-29     -0.008992546        0.005797269 0.2482561    0.2237265
## 2018-09-05      0.025907977        0.005618772 0.2493593    0.2241639
## 2018-09-12      0.057112446        0.016820621 0.2356708    0.2245925
## 2018-09-19      0.002680031        0.002471373 0.2277853    0.2250125
## 2018-09-26      0.032438286        0.005292960 0.2237265    0.2254241
```

```
# pack the data into a format for modelling (only keep the data)
top_sector_stocks <- lapply(best_sector_stocks, function(x) x$data)
top_sector_stocks[[1]]
```

```
##            realized_returns best_shifted_arima     volat vol_forecast
```

```
## 2016-10-05    -0.0112018172       -0.036045213 0.1215014    0.2190119
## 2016-10-12     0.0224263612       -0.006408496 0.1234677    0.2294365
## 2016-10-19     0.0664732141        0.048285037 0.1197529    0.2335310
## 2016-10-26    -0.0334655298        0.007057145 0.2165795    0.2328868
## 2016-11-02     0.0380187203        0.008819051 0.2190119    0.2368343
## 2016-11-09     0.0092616960        0.017965628 0.2294365    0.2391726
## 2016-11-16     0.0222846510        0.007418166 0.2335310    0.2369913
## 2016-11-23     0.0054611936        0.010411607 0.2328868    0.2381453
## 2016-11-30     0.0234997856       -0.006298071 0.2368343    0.2360335
## 2016-12-07     0.0021383860        0.005050070 0.2391726    0.1529022
##        ...
## 2018-07-25    -0.0089178466        0.065700917 0.2219085    0.2383571
## 2018-08-01    -0.0142179259        0.048047263 0.2317340    0.2350262
## 2018-08-08    -0.0422291468       -0.049422510 0.2311597    0.2307443
## 2018-08-15     0.0536069400        0.021594606 0.2433951    0.2134922
## 2018-08-22     0.0004569927        0.057101669 0.2383571    0.2207011
## 2018-08-29    -0.0100738217       -0.002107653 0.2350262    0.2103641
## 2018-09-05     0.0192268721        0.072968420 0.2307443    0.2101598
## 2018-09-12     0.0328710043        0.122334956 0.2134922    0.2099818
## 2018-09-19    -0.0005203785        0.138944040 0.2207011    0.2098267
## 2018-09-26     0.0720473115        0.177702804 0.2103641    0.2096916
```

```
save(top_sector_stocks, file = here("tests","jair", "top_sector_stocks.rda"))
```

Aside: extracting returns for all stocks

```
names(list_xts_sector) # list of tickers and data
```

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

```
head(list_xts_sector[[1]]) # data for first ticker
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-10-05       75.58761             -1      0.001486549     -0.008139995
## 2016-10-12       74.97482              1     -0.008106955      0.006425770
## 2016-10-19       75.45815             -1      0.006446459     -0.002748644
## 2016-10-26       75.25102              1     -0.002744870      0.031497620
## 2016-11-02       77.65897              1      0.031998920      0.010172360
## 2016-11-09       78.45298              1      0.010224270      0.025738570
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2016-10-05      0.001485445     -0.016219680      0.024948510     -0.037026480
## 2016-10-12     -0.008139995      0.001485445     -0.016219680      0.024948510
## 2016-10-19      0.006425770     -0.008139995      0.001485445     -0.016219680
## 2016-10-26     -0.002748644      0.006425770     -0.008139995      0.001485445
## 2016-11-02      0.031497620     -0.002748644      0.006425770     -0.008139995
## 2016-11-09      0.010172360      0.031497620     -0.002748644      0.006425770
##                 atr      adx aaron        bb  chaikin_vol         clv
## 2016-10-05 1.900259 15.44565   -50 0.2934560   -0.4622892  0.18091008
## 2016-10-12 1.872384 15.23639  -100 0.2289285    0.3990933  0.24064338
## 2016-10-19 1.800070 14.75791   -50 0.3060118   -0.4336751  0.09899013
## 2016-10-26 1.722923 14.44363   100 0.2860935   -1.0188680 -0.01496489
## 2016-11-02 1.864142 14.04553    50 0.4910556 -324.8278000  0.05096933
## 2016-11-09 1.989560 13.44222   100 0.5094234    1.1391500  0.19338517
##                       emv      macd      mfi      sar        smi  volume
## 2016-10-05 -0.0006643160 1.3477744 46.50802 95.02127  -5.331162 1315500
```

```
## 2016-10-12 -0.0026850063 1.1358585 37.92195 94.68802 -11.930732 1139000
## 2016-10-19 -0.0019094937 0.9402188 36.19915 94.36810 -17.430099  906400
## 2016-10-26 -0.0021492280 0.7585276 30.28217 94.06097 -19.828752 1331500
## 2016-11-02 -0.0009225739 0.6437468 48.88575 93.76613 -18.073978 5356600
## 2016-11-09 -0.0009562142 0.5919089 59.37208 93.48309 -13.909935 2861300
##                 volat month_index Excess_Retun_Mkt Small_minus_Big
## 2016-10-05 0.10247324          10           0.0058          0.0042
## 2016-10-12 0.10506831          10           0.0006         -0.0022
## 2016-10-19 0.10335977          10           0.0025          0.0013
## 2016-10-26 0.09985285          10          -0.0023         -0.0073
## 2016-11-02 0.13389984          11          -0.0073         -0.0056
## 2016-11-09 0.16512456          11           0.0146          0.0213
##            High_minus_Low Robus_minus_Weak Conservative_minus_Aggressive
## 2016-10-05         0.0080          -0.0048                         0.0044
## 2016-10-12         0.0034           0.0067                         0.0014
## 2016-10-19         0.0094          -0.0011                         0.0049
## 2016-10-26         0.0070           0.0012                         0.0051
## 2016-11-02         0.0025           0.0097                         0.0011
## 2016-11-09         0.0107          -0.0081                         0.0072
##            Risk_free_rate Momentum sarima_100_001 sarima_010_001 sarima_110_001
## 2016-10-05          1e-05  -0.0075    0.003087353    0.031497620    0.012973769
## 2016-10-12          1e-05   0.0051    0.005293415    0.010172360    0.021707222
## 2016-10-19          1e-05  -0.0033    0.003683117    0.025738570    0.017318786
## 2016-10-26          1e-05  -0.0081    0.002663196    0.035597800    0.030264929
## 2016-11-02          1e-05   0.0008    0.007022238   -0.006539587    0.016252583
## 2016-11-09          1e-05  -0.0200    0.004073110    0.021968640    0.006548501
##            sarima_020_001 sarima_120_001 sarima_100_011 sarima_010_011
## 2016-10-05     0.06574388   3.470576e-02    0.003087353    0.031497620
## 2016-10-12    -0.01115290   2.857093e-02    0.005293415    0.010172360
## 2016-10-19     0.04130478   1.493389e-02    0.003683117    0.025738570
## 2016-10-26     0.04545703   4.953651e-02    0.002663196    0.035597800
## 2016-11-02    -0.04867697  -1.150857e-02    0.007022238   -0.006539587
## 2016-11-09     0.05047687  -2.227795e-05    0.004073110    0.021968640
##            sarima_110_011 sarima_020_011 sarima_120_011 best_shifted_arima
## 2016-10-05    0.012973769     0.06574388   3.470576e-02       3.470576e-02
## 2016-10-12    0.021707222    -0.01115290   2.857093e-02       2.857093e-02
## 2016-10-19    0.017318786     0.04130478   1.493389e-02       1.493389e-02
## 2016-10-26    0.030264929     0.04545703   4.953651e-02       4.953651e-02
## 2016-11-02    0.016252583    -0.04867697  -1.150857e-02      -1.150857e-02
## 2016-11-09    0.006548501     0.05047687  -2.227795e-05      -2.227795e-05
##            vol_forecast
## 2016-10-05    0.1338998
## 2016-10-12    0.1651246
## 2016-10-19    0.1746223
## 2016-10-26    0.1752898
## 2016-11-02    0.1772747
## 2016-11-09    0.1757262
```

```
# want to extract the returns for all stock in the list

# how to extract the returns for one stock only!
# realized_returns  best_shifted_arima     volat vol_forecast

select_col <- "realized_returns"
```

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] "GE"    "UPS"   "ETN"   "BA"    "DE"    "MMM"   "MDT"   "JNJ"   "TMO"
## [10] "AMGN"  "CVS"   "ABBV"  "AMD"   "PANW"  "AVGO"  "NVDA"  "ADBE"  "TXN"
## [19] "IPG"   "CHTR"  "NFLX"  "GOOGL" "OMC"   "FOXA"  "MMC"   "ICE"   "C"
## [28] "FI"    "BLK"   "BAC"   "AMZN"  "TSLA"  "ORLY"  "GM"    "CMG"   "SBUX"
```

```
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] "GE"    "UPS"   "ETN"   "BA"    "DE"    "MMM"   "MDT"   "JNJ"   "TMO"
## [10] "AMGN"  "CVS"   "ABBV"  "AMD"   "PANW"  "AVGO"  "NVDA"  "ADBE"  "TXN"
## [19] "IPG"   "CHTR"  "NFLX"  "GOOGL" "OMC"   "FOXA"  "MMC"   "ICE"   "C"
## [28] "FI"    "BLK"   "BAC"   "AMZN"  "TSLA"  "ORLY"  "GM"    "CMG"   "SBUX"
```

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

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-01-06       162.5969             -1               NA      -0.06875702
## 2016-01-13       151.7929             -1     -0.066446514      -0.00853459
## 2016-01-20       150.5030             -1     -0.008498274       0.00000000
## 2016-01-27       150.5030              1      0.000000000       0.02364675
## 2016-02-03       154.1043             -1      0.023928553      -0.01298985
## 2016-02-10       152.1155              1     -0.012905843       0.03608994
##            log_returns_lag0 log_returns_lag1 log_returns_lag2 log_returns_lag3
## 2016-01-06               NA               NA               NA               NA
## 2016-01-13      -0.06875702               NA               NA               NA
## 2016-01-20      -0.00853459      -0.06875702               NA               NA
## 2016-01-27       0.00000000      -0.00853459      -0.06875702               NA
## 2016-02-03       0.02364675       0.00000000      -0.00853459      -0.06875702
## 2016-02-10      -0.01298985       0.02364675       0.00000000      -0.00853459
##            atr adx aaron bb chaikin_vol clv emv macd mfi      sar smi   volume
## 2016-01-06  NA  NA    NA NA          NA  NA  NA   NA  NA 180.6148  NA 10433554
## 2016-01-13  NA  NA   -50 NA          NA  NA  NA   NA  NA 184.4713  NA  9278669
## 2016-01-20  NA  NA  -100 NA          NA  NA  NA   NA  NA 184.4713  NA 14665098
## 2016-01-27  NA  NA   -50 NA          NA  NA  NA   NA  NA 183.8179  NA  7679764
## 2016-02-03  NA  NA   100 NA          NA  NA  NA   NA  NA 183.1907  NA  7370884
## 2016-02-10  NA  NA   100 NA          NA  NA  NA   NA  NA 182.5886  NA  5410993
##            volat month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2016-01-06    NA           1          -0.0135         -0.0023         0.0000
## 2016-01-13    NA           1          -0.0267         -0.0062         0.0081
## 2016-01-20    NA           1          -0.0094          0.0173        -0.0127
## 2016-01-27    NA           1          -0.0111         -0.0042         0.0171
## 2016-02-03    NA           2           0.0046         -0.0025         0.0047
## 2016-02-10    NA           2           0.0001         -0.0021        -0.0055
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2016-01-06           0.0015                        0.0004          0e+00
## 2016-01-13           0.0040                        0.0063          0e+00
## 2016-01-20           0.0008                       -0.0052          0e+00
## 2016-01-27          -0.0013                        0.0092          0e+00
## 2016-02-03           0.0041                        0.0032          1e-05
## 2016-02-10          -0.0030                       -0.0069          1e-05
##            Momentum
## 2016-01-06   0.0192
```

```
## 2016-01-13   0.0016
## 2016-01-20  -0.0011
## 2016-01-27  -0.0048
## 2016-02-03  -0.0241
## 2016-02-10   0.0065
```