Group28_CS971

Celia Rubio Madrigal, Girish Koushik

Introduction and background

Algorithmic trading refers to any form of trading using sophisticated algorithms to automate all or some part of the trade cycle. The use of technical indicators to figure out the best time to buy and sell stocks has been around for a long time. The most popular technical indicators are the simple moving average (SMA), the exponential moving average (EMA), the relative strength index (RSI), the moving average convergence divergence (MACD), and the Bollinger bands.

The MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. A standard MACD is calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA. The result of that calculation is the MACD line. A 9-day EMA of the MACD called the "signal line," is then plotted on top of the MACD line, which can function as a trigger for buy and sell signals. Traders may buy the security when the MACD crosses above its signal line and sell —or short— the security when the MACD crosses below the signal line. The MACD indicator is used to identify the trend direction as well as the momentum. Our project will use optimization to find the best parameters for the MACD, as well as build other custom indicators to increase profit based on trading signals.

The authors in [1] take a combined signal approach to increase profitability by combining the signals from multiple technical indicators. They make use of different trading rules such as the moving average crossover rule (MAC-O) and trading range break-outs (TRB-O). The trading rules are applied on the different stock indices like NASDAQ, DJIA, and TSX300 and a majority voting technique is adopted to find the best trading signal. While the results obtained were statistically significant, the authors stress the need to find better parameters for the technical indicators and an AI-based method to implement the combined signal approach. We are taking this idea of combining signals, first with a majority voting, and secondly by letting a genetic programming algorithm find the best grammar combination. We will be also using stock indices for our profit evaluation, given that they are a global view of the market.

In [2], the authors propose to combine the technical indicators RSI and MA and optimize them using genetic algorithms for the foreign exchange market. The focus is on calculating the most appropriate trade timing rather than predicting trading prices; this will also be the focus in our study. The fittest rules are selected for reproduction, where crossover and mutation operators are applied to create new offspring, and repeated for several generations until the optimal trading rule is found. In their paper, the proposed GA system outperforms a Neural Network in terms of profitability and stability. We will also use a genetic algorithm to combine technical indicators, but it will be based on grammar trees —combinations of the optimized indicators found previously.

The different components of algorithmic trading (AT) and the techniques involved are explained in the survey paper [3]. Algorithmic trading involves several steps such as data cleaning, pre-trade analysis, trade signal generation, trade execution, and post-trade handling. In our project, we are focusing solely on the trade signal generation; therefore, we are choosing the remaining system to be simple, such as the adoption of a

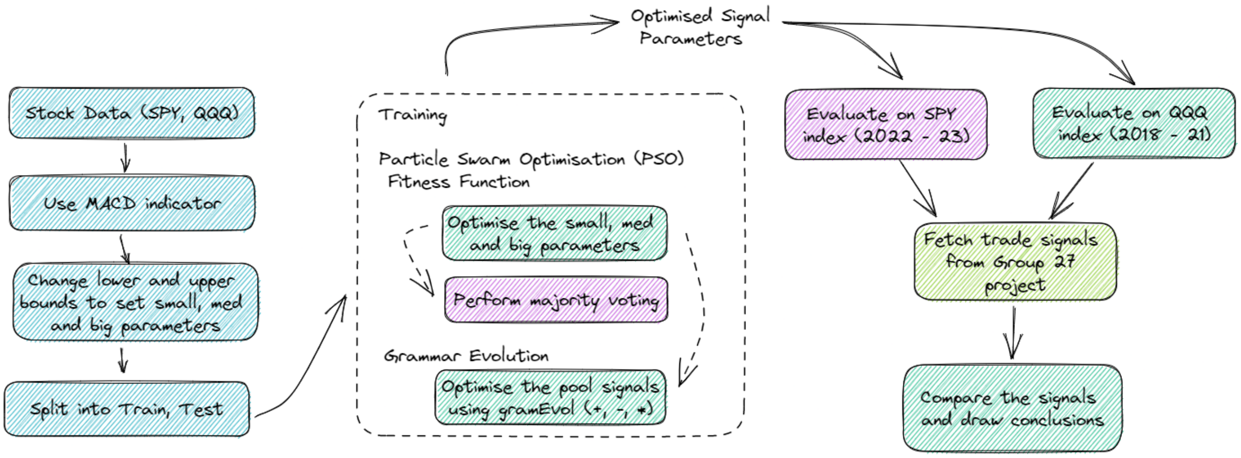
[1] Lento, C. and Gradojevic, N., 2007. The profitability of technical trading rules: A combined signal approach. Journal of Applied Business Research (JABR), 23(1).

[2] Hirabayashi, A., Aranha, C. and Iba, H., 2009, July. Optimization of the trading rule in foreign exchange using genetic algorithm. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 1529-1536).

[3] Treleaven, P., Galas, M. and Lalchand, V., 2013. Algorithmic trading review. Communications of the ACM, 56(11), pp.76-85.

risk-neutral strategy. Finally, their paper highlights that the competitive nature of algorithmic trading makes for a secretive community where implementation details are difficult to find. We would therefore like to find more examples in the literature of real-world techniques.

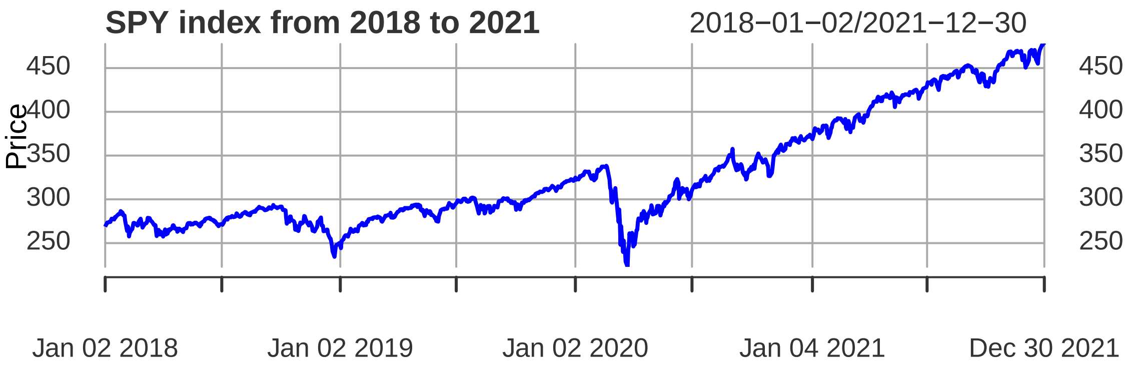
In summary, this project focuses on generating optimal trade signals, and then evaluating them with backtesting. We will be trying to optimize the MACD parameters `nFast`, `nSlow`, and `nSig` using the Particle Swarm Optimization (PSO) algorithm, replacing the usual 12, 26 and 9 values. We are training the algorithm by finding the final profits of an investing period from 2018 to 2021, using the S&P 500 stock index. The optimized parameters are then used to generate the trade signal, and backtesting is performed to evaluate the performance of the trading strategy. Backtesting is done on the historical data of the NASDAQ for the same period, as well as for the S&P 500 index for 2022. The results are compared with the results obtained using the default parameters of the MACD indicator. Moreover, we are creating two new signals: one as the majority voting for all individual MACD signals, and another one by running a genetic programming optimization on the combination of these signals.


Optimised Signal Parameters
Stock Data (SPY, QQQ)
Use MACD indicator
Change lower and upper bounds to set small, med and big parameters
Split into Train, Test
Training
Particle Swarm Optimisation (PSO) Fitness Function
Optimise the small, med and big parameters
Perform majority voting
Grammar Evolution
Optimise the pool signals using gramEvol (+, -, *)
Evaluate on SPY index (2022 - 23)
Evaluate on QQQ index (2018 - 21)
Fetch trade signals from Group 27 project
Compare the signals and draw conclusions

Overview of data

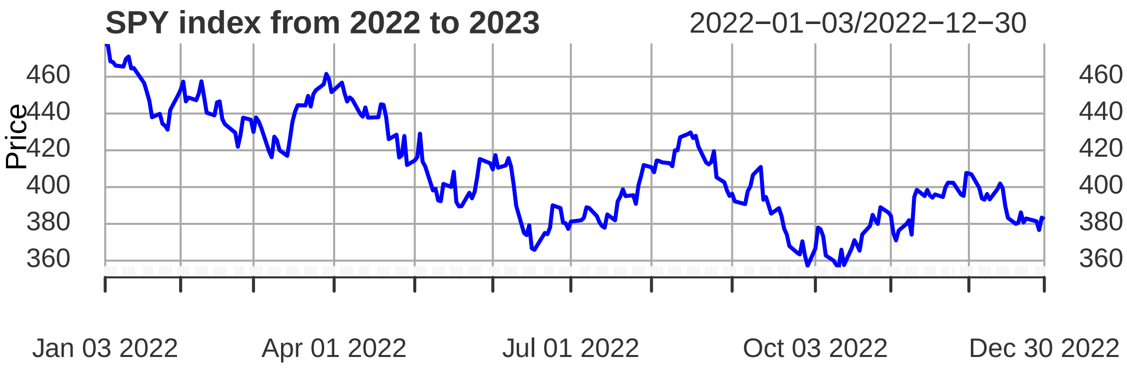
The SPY index is a market capitalization-weighted index of the 500 largest publicly traded companies in the US. The index is a good choice for our project as it is 1/10 expense ratio of the S&P 500 index and hence, with the initial investment amount of $10000, we can figure out the best trading strategy.

```r
# Training data: SPY index from 2018 to 2021
getSymbols("SPY", src = "yahoo", from = "2018-01-01", to = "2021-12-31")
training_index <- Cl(SPY)
```

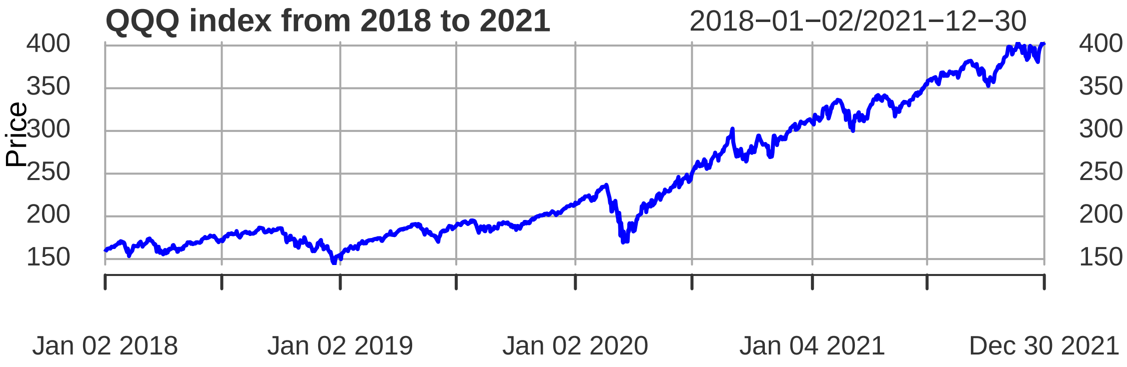


The above plot shows the closing prices of the SPY index from 2018 to 2021. This will be our training data. We can see that it has a big decrease during the pandemic, and a steady increase afterwards. For our testing data, we will follow two parallel approaches: changing the dates (`test_year`), and changing the index (`test_index`).

```r
# Test data: SPY index from 2022 to 2023 and QQQ index from 2018 to 2021
getSymbols("SPY", src = "yahoo", from = "2022-01-01", to = "2023-01-01")
test_year <- Cl(SPY) # (Changing the date)
getSymbols("QQQ", src = "yahoo", from = "2018-01-01", to = "2021-12-31")
test_index <- Cl(QQQ) # (Changing the index)
```



The above plot shows the closing prices of the SPY index from 2022 to 2023, which is the first test data. It has a deep decrease for the whole period.



The above plot shows the closing prices of the QQQ index from 2018 to 2021. This index is 1/5 expense ratio of the NASDAQ index and hence, with the initial investment amount of $10000, we can test our trading strategy with the same characteristics as the training index. This helps in figuring out how good the trading strategy is in generalizing to other indices.

Trading strategy

Our main strategy will be to create optimized signals based on the profits they obtain, using the MACD indicator as a base.

Optimising MACD parameters with PSO

First, we will choose the best parameters for the MACD indicator. We will need a function that, given a set of parameters and some closing prices, will give the signal to buy, sell or hold for the following day. This is automatically calculated by `TTR`'s `MACD` function, which we wrap into `get_signal` and set to hold the first days of the sliding window.

```r
get_signal <- function(params, index) {
    fast = as.integer(params[1]); slow = as.integer(params[2]); signal = as.integer(params[3])
    macd <- MACD(index, nFast = fast, nSlow = slow, nSig = signal, maType = "EMA")
    signal <- ifelse(macd$macd-macd$signal>0.1, 1, ifelse(macd$macd-macd$signal< -0.1, -1, 0))
    signal[is.na(signal)] <- 0
    return(signal)
}
```

To know which signal is best we need a way to calculate its generated profit. We are following a naive approach where we buy or sell a single stock at each signal call, starting with an initial capital of 10000. In `backtest_signal` we return a dataframe with vectors that store our cash value, asset value, the number of shares and total value per day.

```r
backtest_signal <- function(trading_signal, prices, initial_capital = 10000) {
    cash_value <- rep(NA, length(trading_signal) + 1)
    asset_value <- rep(NA, length(trading_signal) + 1)
    number_of_shares <- rep(NA, length(trading_signal) + 1)
    trading_signal <- as.vector(trading_signal); prices <- as.vector(prices)
    cash_value[1] <- initial_capital; asset_value[1] <- 0; number_of_shares[1] <- 0
    buy_hold <- rep(NA, length(trading_signal) + 1); buy_hold[1] <- initial_capital

    for (i in 1:(length(trading_signal))) {
        if (trading_signal[i] == 1 && cash_value[i] >= prices[i]) { # buy
            number_of_shares[i + 1] <- number_of_shares[i] + 1
            cash_value[i + 1] <- cash_value[i] - prices[i]
        } else if (trading_signal[i] == -1 && number_of_shares[i] > 0) { # sell
            number_of_shares[i + 1] <- number_of_shares[i] - 1
            cash_value[i + 1] <- cash_value[i] + prices[i]
        } else { # hold
            number_of_shares[i + 1] <- number_of_shares[i]
            cash_value[i + 1] <- cash_value[i]
        }
        asset_value[i + 1] <- number_of_shares[i + 1] * prices[i]
        buy_hold[i+1] = floor(initial_capital / prices[1]) * prices[i]
    }
    return(data.frame(
        cash_value = cash_value, asset_value = asset_value, buy_hold = buy_hold,
        number_of_shares = number_of_shares, total_value = cash_value + asset_value
    ))
}
```

In `get_profit` we obtain the difference between the initial capital and the total value on the last day. The downside to this naive approach is that it is very dependent on the time period we specified. It could be perhaps interesting to evaluate by how well it integrates into a model that predicts the next day's price as its fitness value.

```r
get_profit <- function(trading_signal, index) {
    backtest <- backtest_signal(trading_signal, index, 10000)
    return(backtest$total_value[length(backtest$total_value)] - backtest$total_value[1])
}
```

Now that we have a way of comparing signals, we will use the PSO algorithm to find the best combination of parameters that yield the best profits for its signals. The fitness function of a set of parameters will be the (negated) profit of its signal, as the algorithm finds minimum points and we want maximum profit. Here we train our PSO algorithm, which comes from the `hydroPSO` library. We give it the lower and upper bounds of every set of parameters, and the training closing prices.

```r
get_optimal_parameter <- function(lower_bounds, upper_bounds, index) {
    fitness <- function(params) { return(-get_profit(get_signal(params, index), index)) }
    opt_results <- hydroPSO(fn = fitness, lower = lower_bounds, upper = upper_bounds,
                        control = list(c1 = 2, c2 = 2, maxit = 100))
    return(as.integer(opt_results$par))
}
```

We will find 3 sets of optimal parameters for different sizes of ranges; we will call them `small`, `med` and `big`. We will also consider the common set of parameters for MACD (12, 26, 9) and call it `common`. Note that this process is easily generalisable to other types of indicators such as RSI.

```r
parameter_small <- get_optimal_parameter(c(1,5,5),c(5,20,50),training_index)
parameter_med <- get_optimal_parameter(c(1,15,5),c(15,50,50),training_index)
parameter_big <- get_optimal_parameter(c(1,50,5),c(50,200,50),training_index)
parameter_common <- c(12, 26, 9)
```

```
##                  Fast Slow Signal
## parameter_small     1    7     46
## parameter_med       1   46      7
## parameter_big      49   56     11
## parameter_common   12   26      9
```

We can see that `small` and `medium` exchange their `nSlow` and `nSignal` parameters, such that they end up being the same signal. Also `big` has a very short difference between `nFast` and `nSlow`, so it will not react to the market as quickly. This will be why it performs well during the 2020 steep decrease for the training set. We now obtain the signals; we will plot them in the evaluation section.

```r
signal_small <- get_signal(parameter_small, training_index)
signal_med <- get_signal(parameter_med, training_index)
signal_big <- get_signal(parameter_big, training_index)
signal_common <- get_signal(parameter_common, training_index)
pool_signals <- data.frame(signal_small, signal_med, signal_big, signal_common)
colnames(pool_signals) <- c("small", "med", "big", "common")
```

Getting combined signals

We will be creating two methods of obtaining new signals: one, to make the signals vote for the result; two, to create an optimized grammar expression with them. The second approach should behave better than the first, as it contains it.

A majority voting signal

In get_majority_signal we sum the signals across the 4 columns of the pool. If the sum is greater than or equal to 2, we return 1, if it is less than or equal to -2, we return -1, and if it is between -2 and 2, we return 0. This is because we want to buy when the majority of signals are positive, sell when the majority of signals are negative, and hold when the majority of signals are neutral.

```
get_majority_signal <- function(pool) {
    signal_sum <- rowSums(pool)  # sum the pool signals across the 4 columns
    return(ifelse(signal_sum >= 2, 1, ifelse(signal_sum <= -2, -1, 0)))
}
signal_majority <- get_majority_signal(pool_signals)
```

A grammar-evolutioned signal

In get_grammar_expr we create a grammar that evaluates to a combination of our basic signals. We feed it into a grammatical evolution algorithm from the gramEvol library, which will find the best expression among (mod)addition, (mod)subtraction and multiplication that maximizes profit. We then use this expression grammar_expr to create a new signal in get_grammar_signal.

```
add_mod <- function (a,b) ((a+b+4)%%3 -1); sub_mod <- function (a,b) ((a-b+4)%%3 -1)
get_grammar_expr <- function(signals) {
  rules <- list(expr = grule(op(expr, expr),var,-(expr)), op = grule(add_mod,sub_mod,'*'),
              var = grule(signals$small, signals$med, signals$big, signals$common))
  grammar <- CreateGrammar(rules)
  grammar_fitness <- function(expr) { return(-get_profit(eval(expr),training_index)) }
  ge <- GrammaticalEvolution(grammar, grammar_fitness, iterations = 100, max.depth = 5)
  return(ge$best$expressions)
}
get_grammar_signal <- function(signals,expr) { return(eval(expr)) }
```

```
grammar_expr <- get_grammar_expr(pool_signals)
signal_grammar <- get_grammar_signal(pool_signals,grammar_expr)
grammar_expr
```

```
## expression(signals$big * (signals$small * -(sub_mod(signals$med *
##     signals$big, signals$med))))
```

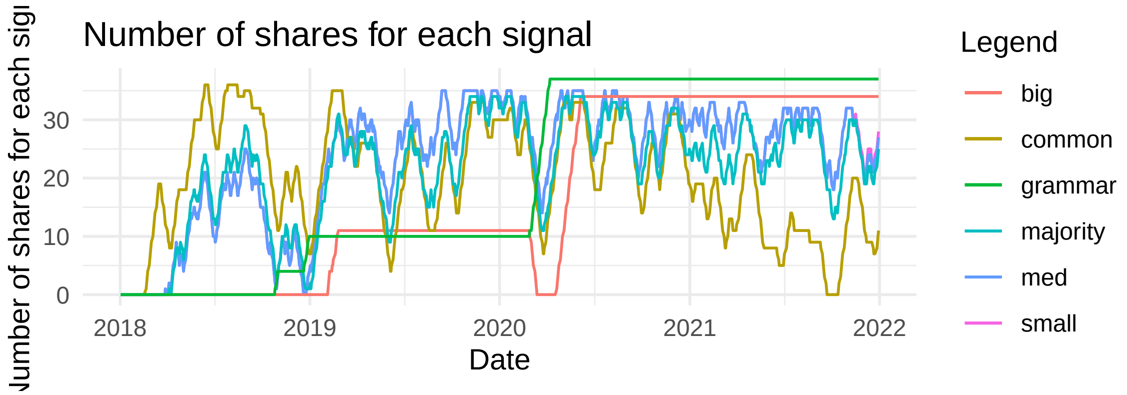
The above is the optimized expression we will be using to calculate our grammar signal.

Performance on training

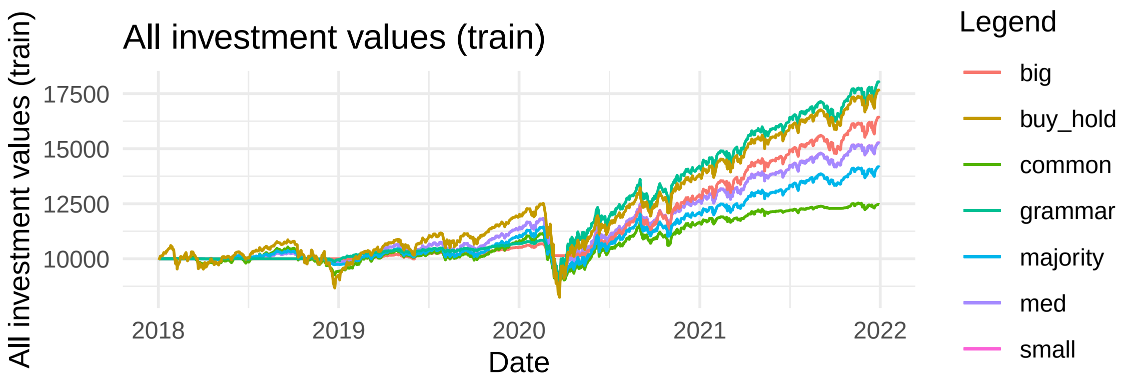
This is the profit obtained for all signals considered until now, calculated with get_profit for training_index.

```
##    small      med      big   common majority  grammar
##  5239.56  5222.74  6389.28  2457.55  4151.22  7994.62
```

We can see how these signals produce different number of shares when investing during the train period, which is calculated from backtest_signal's output dataframe.



Below we can see our total value achieved via each of the strategies, again calculated from `backtest_signal`, as well as a naive strategy which consists of buying all stocks at the beginning and holding afterwards. As the train period had a steep increase of prices at the end, the buy-and-hold approach is very successful then, as well as our `grammar` signal. The `common` signal is doing the worst, as it seems to be the most cautious, although it is interesting to see that `majority` is too. Finally, it is worth noting that `big` does very well during the steep decrease of 2020.



Performance on test

Backtest on future dates

We now obtain the same evaluation but for the `test_year` index (i.e., SPY from 2022 to 2023). We get all signals with `get_signal`, `get_majority_signal` and `get_grammar_signal`.

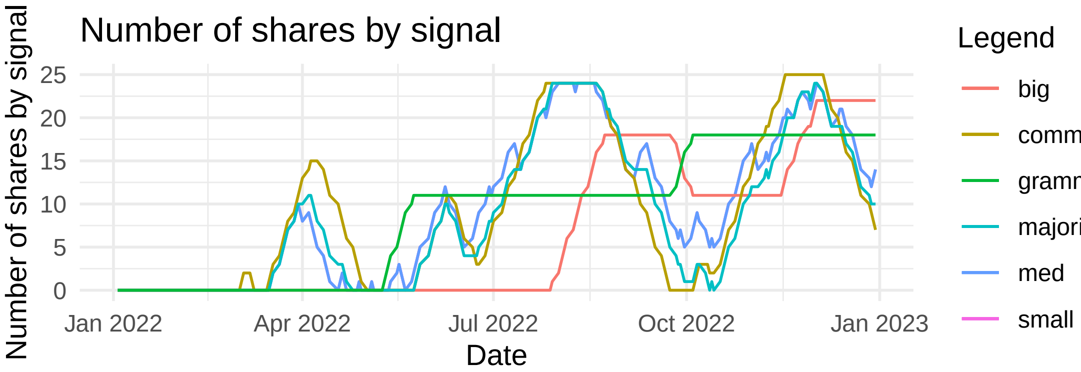
```
test_signal_small <- get_signal(parameter_small, test_year)
test_signal_med <- get_signal(parameter_med, test_year)
test_signal_big <- get_signal(parameter_big, test_year)
test_signal_common <- get_signal(parameter_common, test_year)

test_pool_signals <- data.frame(test_signal_small,test_signal_med,test_signal_big,test_signal_common)
colnames(test_pool_signals) <- c("small", "med", "big", "common")
```

```
test_signal_majority <- get_majority_signal(test_pool_signals)
test_signal_grammar <- get_grammar_signal(test_pool_signals,grammar_expr)
```

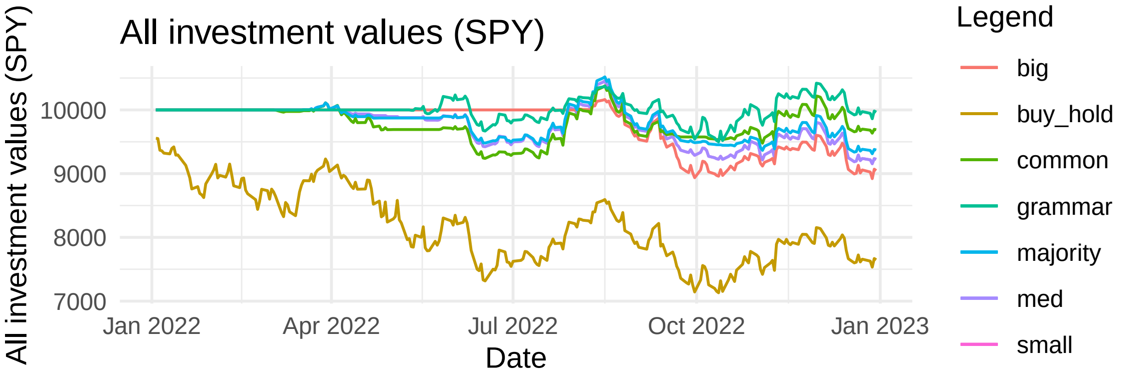
```r
test_all_signals <- data.frame(test_pool_signals, test_signal_majority, test_signal_grammar)
colnames(test_all_signals) <- c("small", "med", "big", "common","majority", "grammar")
```

This is the profit for all of them in this period, and the number of shares hold against time. We can see that it is a period of decline, as we commented earlier.

```
##      small        med        big     common   majority    grammar
## -776.30007 -776.30007 -951.88018 -312.17998 -629.88003  -35.10012
```



In the plot below we see how the total value progresses. The `grammar` signal is the best one to hold up against the decrease, while it is normal for the buy-and-hold to perform very badly. It is interesting to see that the `common` signal also performs well, but `majority` does not improve on the rest.



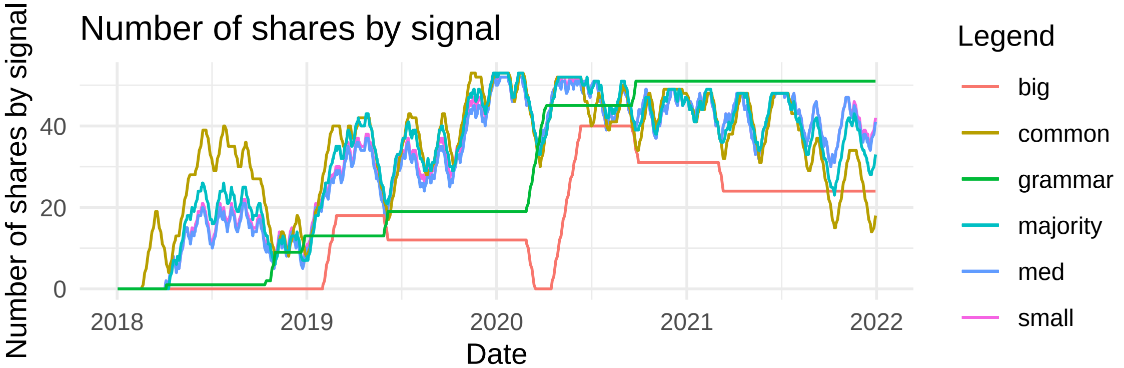
Test on another stock index

Similar to the previous test, we obtain the evaluation for the `test_index` index (i.e., QQQ data from 2018 to 2021). We get all signals with `get_signal`, `get_majority_signal` and `get_grammar_signal` again, combined so that we can plot them all and get their profits, seen below.

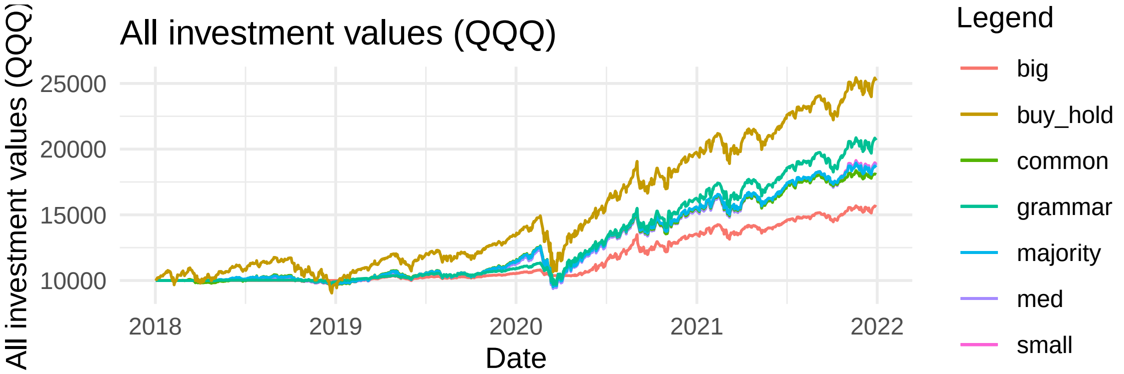
```r
qqq_signal_small <- get_signal(parameter_small, test_index)
qqq_signal_med <- get_signal(parameter_med, test_index)
qqq_signal_big <- get_signal(parameter_big, test_index)
qqq_signal_common <- get_signal(parameter_common, test_index)

qqq_pool_signals <- data.frame(qqq_signal_small, qqq_signal_med, qqq_signal_big, qqq_signal_common)
colnames(qqq_pool_signals) <- c("small", "med", "big", "common")
```

```r
qqq_signal_majority <- get_majority_signal(qqq_pool_signals)
qqq_signal_grammar <- get_grammar_signal(qqq_pool_signals, grammar_expr)

qqq_all_signals <- data.frame(qqq_pool_signals, qqq_signal_majority, qqq_signal_grammar)
colnames(qqq_all_signals) <- c("small", "med", "big", "common","majority", "grammar")
```

```
##     small       med       big    common  majority   grammar
##  8847.431  8745.321  5604.280  8068.300  8618.461 10686.050
```



Above, we calculate the number of shares during the period, and below we plot all investment values again. Following the market increase at the end, the buy-and-hold approach is very profitable, as we know. We also note that the `grammar` signal is also quite successful; having success in both test sets, we can conclude with a positive result from this method. While `big` is too conservative, the rest do not vary much.

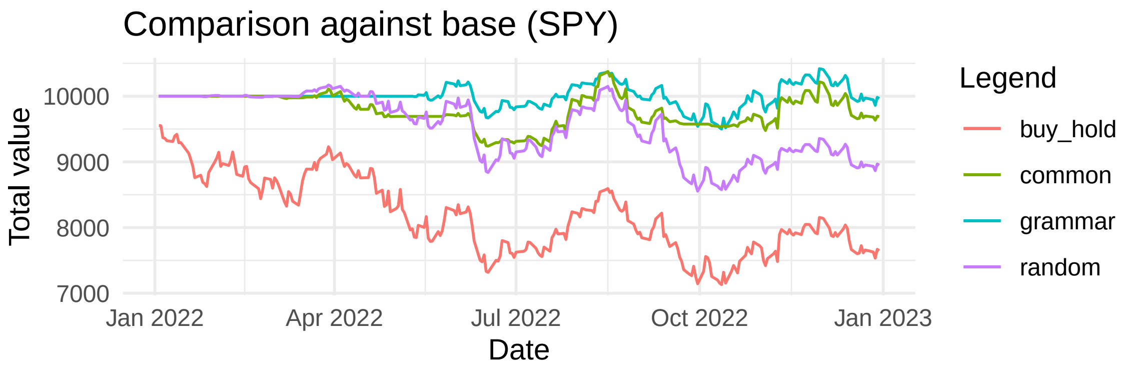


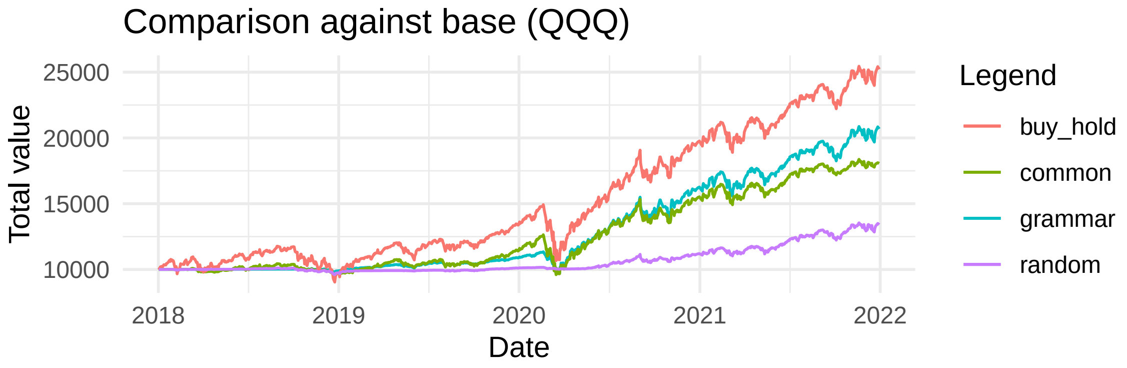
Comparison

Recapitulation of `grammar` against base approaches

We have found out that the `grammar` signal approach behaves well at both the market increase of 2021 for QQQ and at the decrease of 2022 for SPY. Meanwhile, the buy-and-hold approach is not reliable against these market changes, and the original parameters of MACD are too cautious. We are also comparing it against a random signal, which clearly does not perform well.

```
spy_random_signal <- xts(sample(-1:1, length(rownames(test_all_signals)), replace = TRUE),
                         order.by = as.Date(rownames(test_all_signals)))
qqq_random_signal <- xts(sample(-1:1, length(rownames(qqq_all_signals)), replace = TRUE),
                         order.by = as.Date(rownames(qqq_all_signals)))
```





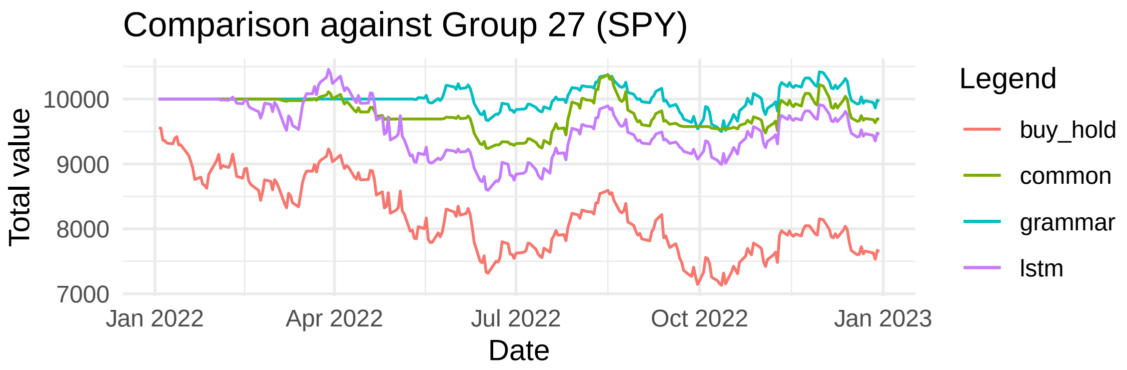
The `grammar` signal against prediction-based signal

Our classmates from Group 27 have produced a model to predict the price on the next day; with that, they have produced a signal that buys when the price is going up in the future, and sells when it will decrease. We want to compare our signals with their approach, so we load their data for both of our test sets and preprocess it to match formats.

```
test_signal_zestin <- read.csv("predicted_days_spy.csv")
# Replace the column values 'buy' with 1, 'sell' with -1 and 'hold' with 0
test_signal_zestin$Signals[test_signal_zestin$Signals == 'Buy'] <- 1
test_signal_zestin$Signals[test_signal_zestin$Signals == 'Sell'] <- -1
test_signal_zestin$Signals[test_signal_zestin$Signals == 'Hold'] <- 0
```

```
# Check length of their signal data and append the `Signals` column to the test_all_signals dataframe
if (length(test_signal_zestin$Signals) == nrow(test_all_signals)) {
    test_all_signals <- cbind(test_all_signals, test_signal_zestin$Signals)
}else{test_all_signals <- cbind(test_all_signals, test_signal_zestin$Signals[1:nrow(test_all_signals)])
colnames(test_all_signals)[ncol(test_all_signals)] <- "lstm"
test_all_signals[is.na(test_all_signals)] <- 0 # replace NA with hold
test_all_signals <- as.data.frame(test_all_signals)
test_all_signals$lstm <- as.numeric(test_all_signals$lstm)
```

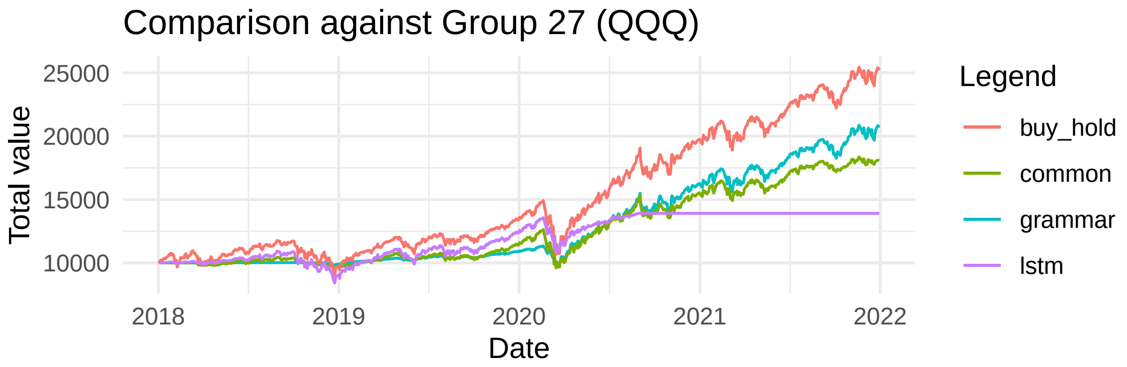
For our first test set (i.e., SPY from 2022 to 2023), we are plotting our best signal `grammar` against the `lstm` signal from Group 27. For this time period, we can observe that, due to the bear market, our `grammar` signal performs better than all the other signals especially, the `buy_hold` signal, as that strategy does work in a bear market. Our signal also does better than the `lstm` signal which has been trained on the SPY data.



```
qqq_signal_zestin <- read.csv("predicted_days_qqq.csv")
qqq_signal_zestin$Signals[qqq_signal_zestin$Signals == 'Buy'] <- 1
qqq_signal_zestin$Signals[qqq_signal_zestin$Signals == 'Sell'] <- -1
qqq_signal_zestin$Signals[qqq_signal_zestin$Signals == 'Hold'] <- 0
```

```
if (length(qqq_signal_zestin$Signals) == nrow(qqq_all_signals)) {
    qqq_all_signals <- cbind(qqq_all_signals, qqq_signal_zestin$Signals)
}else{qqq_all_signals <- cbind(qqq_all_signals, qqq_signal_zestin$Signals[1:nrow(qqq_all_signals)])}
colnames(qqq_all_signals)[ncol(qqq_all_signals)] <- "lstm"
qqq_all_signals[is.na(qqq_all_signals)] <- 0
qqq_all_signals <- as.data.frame(qqq_all_signals)
qqq_all_signals$lstm <- as.numeric(qqq_all_signals$lstm)
```

For our second test set (i.e., QQQ data from 2018 to 2021), we are plotting our best signal `grammar` against the `lstm` signal from Group 27. For this time period, we can observe that, due to the bull market, our `grammar` signal performs better than the `lstm` signals, but not the `buy_hold` signal. This is because, in a bull market, the indicators expect the prices to keep going up and hence, the `buy_hold` signal performs better than all the other strategies.


Comparison against Group 27 (QQQ)
Total value
Date
Legend
buy_hold
common
grammar
lstm