CITF: Contrarian Indicator Trend Following
FNCE30010 - Algorithmic Trading

Elisha Xin Yee Chung (1019581)
Jason F. Suhartanto (1086250)
Stefan Marbun (1046479)

University of Melbourne - Semester 2, 2021.

Abstract

Trend following (TF) is a popular strategy to make trading decisions based on technical analysis. In TF, traders simply jump on the current market sentiment, i.e. buying when the trend is positive and selling when the trend is negative. In this paper, we propose an improved TF strategy: the Contrarian Indicator Trend Following (CITF) strategy. This strategy combines a baseline TF model with popular contrarian indicators and multi-threshold analysis to determine when to open long positions. The baseline model chosen was one that utilised a Linear Regression (LR) crossover indicator published on QuantConnect, which we backtested alongside CITF strategy between 2015 and 2020 with the SPY ETF as our benchmark. The analysis on the backtesting results show that CITF demonstrates a more conservative risk control compared to the baseline, which results in overall better alpha and risk metrics (Sharpe Ratio, drawdown) with stronger returns in bear markets. This paper also acknowledges possible overfitting trade-offs and proposes alternative countermeasures.

Keywords: *Technical analysis, Trend following, Contrarian strategy, Backtesting, Algorithmic trading.*

Contents

1 Introduction

Trend following (TF) is a widely implemented trading strategy due to its simple and extensible algorithm, resulting in its long-lived popularity among trading researchers. According to this strategy, one should long an asset when its price trends positively and short otherwise. TF traders are not interested in forecasting future price movements, but rather just jumping on whatever the current trend is, betting that it will persist.

This paper presents an improved TF strategy: the Contrarian Indicator Trend Following (CITF). The baseline TF model uses a moving linear regression crossover indicator, and the improved CITF strategy uses additional indicators from the contrarian strategy. Section 2 gives a preliminary analysis on the behaviours of TF and contrarian strategies from literature research. Section 3 proposes our improvement approaches to the baseline TF strategy. The backtest results are given in Section 4, and its analyses are given in Section 5.

2 Trend Following

The baseline TF model uses the time series momentum strategy, which is to long assets with recent positive performance and to short assets with recent negative performance (Hurst et al., 2017; Ziemba, 2013). This is the opposite to the popular trading principle to buy low and sell high, which can yield different risks. Instead, trend followers "cut the losses" and "let the profits run", expecting the overall strategy to be profitable.

TF strategies can also be implemented over a wide range of asset classes, whose significance is statistically proven by Lempérière et al. (2014). Another study (Hurst et al., 2017) also shows the ability of TF models to withstand major recessions throughout history, including the 2000 dot-com bubble and the 2008 financial crisis. Consequently, it is widely known that TF is considered a long volatility strategy where it is effective when the volatility is either very low or very high.

However, academics have drawn skepticism over the use of the simple baseline TF strategy to make trading decisions. A study (Hutchinson & O'Brien, 2014) found that the strategy has been performing below its long term average since the 2008 crisis. Another report emphasises the importance of having diverse strategies for our portfolio, as relying completely on a TF strategy may not be a good idea. Additionally, there are many improvements to enhance a TF strategy, such as combining it with additional indicators or optimisation methods for tuning parameters.

This paper will focus on improving a baseline TF model with a popular strategy called the contrarian. Contrarian traders opt to go against the current market sentiment at a given time, i.e. to purchase low performing assets and sell high performing assets. While the indicators of this strategy are in contrast to traditional TF indicators, some papers (Zhao & Si, 2020; Liu et al., 2018) have successfully combined both strategies to improve trading performance.

3 Methodology

This section will describe the baseline TF strategy we use and our approaches to improve it. Section 3.1 covers a baseline TF strategy that will be used as a framework, and Section 3.2 covers our approaches to improve the baseline strategy.

3.1 Baseline Strategy: Trend Following with Linear Regression

Strategy Description

Our baseline model follows a TF strategy presented by Jing Wu in QuantConnect[1]. This strategy uses a moving linear regression (LR) indicator to determine a crossover. The model backtests its performance with an LR indicator lookback period of six months. The main asset to be used for the backtest is the

[1] Jing Wu's ETF Trend Following Strategy, retrieved from https://www.quantconnect.com/forum/discussion/3260/etf-trend-following-algorithm/, accessed on October 15, 2021.

SPDR S&P 500 Trust (SPY) ETF, as the performance of major market indices are popular benchmarks to evaluate trading strategies. The pseudocode of the model is given below:

Algorithm 1 Baseline TF Strategy

Get `actual_open_price_today`
`slope`, `fitted_open_price_today` $\leftarrow$ LinearRegression(Opens of last half year)
$\theta \leftarrow$ Slope Threshold > 0
if `slope` is uptrend **and** `slope` $> \theta$ **then**
  if `actual_open_price_today` $>$ `fitted_open_price_today` **and** asset not invested **then**
    Long asset
  else if `actual_open_price_today` $>$ 95% upper Bollinger band **and** asset is already invested **then**
    Liquidate asset
  end if
else if `slope` is downtrend **and** `slope` $< -\theta$ **then**
  if `actual_open_price_today` $<$ `fitted_open_price_today` **and** asset not invested **then**
    Short asset
  else if `actual_open_price_today` $<$ 95% lower Bollinger band **and** asset is already invested **then**
    Liquidate asset
  end if
else if asset is already invested **then**
  if Position is Long **and** `slope` is downtrend **then**
    Liquidate asset (Stop loss)
  else if Position is Short **and** `slope` is uptrend **then**
    Liquidate asset (Stop loss)
  end if
end if

Disadvantages of Baseline TF

One disadvantage of using TF is its liability to false signals. Prices move greatly compared to the stable linear regression line. Occasionally, a price goes above the LR line but it may be just a period of the asset being overbought. Additionally, TF does not take market mean reversion into consideration. In the mean reversion theory, after extreme price movements, prices will eventually revert back to the long term average once the market has corrected itself. If trend followers go long on an overbought asset right when its price is about to correct itself and decrease back to the mean, they will not be riding the upwards trend as expected.

Improvement Suggestions

In addition to the TF indicators, contrarian trend indicators could be used to time our entry into the trend. Instead of just using a moving average crossover indicator, we will measure if mean reversion has already occurred. We will only go long on an asset after a pullback has occurred so we do not buy in the downturn.

Another alternative for improvement is to measure if an asset is oversold or overbought, which may indicate potential future price movements. For instance, when an asset is oversold, the price of that asset is lower than what it should be. Traders will react to this low price and buy to gain profit, with the market readjusting as the price hikes up from demand. Once the price increases to its expected mean, equilibrium has been reached and demand will eventually be in equilibrium too. Thus, oversold assets have potential for future price increases. This idea suggests similar indications for an overbought asset.

3.2 Improved Strategy: Contrarian Indicator Trend Following (CITF)

For our improved CITF model, we consider two strategies often used by contrarian investors: 1) the detection of mean reversion, and 2) the calculation of Relative Strength Index (RSI) values to determine overbuying and overselling.

1. For mean reversion detection, we check if the price has reverted back to its SMA. If the current price is close enough to its current SMA and the previous period's price was not as close to its SMA, it means a pullback to the mean has occurred and the trader can now ride the trend. A tolerance level θ is set to determine if the price is close enough to its SMA.
2. We also use the Relative Strength Index (RSI) with a period of 14 days to measure if an asset is oversold or overbought. If an asset is oversold, it may imply that the price is undervalued and it might rise in the future due to the market correcting itself. The opposite holds for overbought assets. By consensus, an RSI value of lower than 30 indicates that the asset is oversold, and a value of higher than 70 indicates that the asset is overbought.

The TF indicator that we opted to use is the 200 day Simple Moving Average (SMA) indicator. In some cases, SMA has been evident to perform better trades than an LR indicator. Moreover, to keep a consistent result comparison, we are using the same set of ETFs as the baseline strategy. Below is the pseudocode for our implementation:

Algorithm 2 CITF Strategy

Get $\texttt{price}_t, \texttt{price}_{t-1} \leftarrow$ Open Price
Get $\texttt{SMA}_t, \texttt{SMA}_{t-1} \leftarrow$ 200 Day SMA
Get $\texttt{RSI}_t \leftarrow$ 14 Day RSI
$\theta_{RSI} \leftarrow 30$
$\Delta_{SMA} \leftarrow 0.5$
if $\texttt{price}_t > \texttt{SMA}_t$ **then**
    $\texttt{isUptrend} \leftarrow$ **True**
end if
if $\texttt{price}_t - \texttt{SMA}_t \leq \Delta_{SMA}$ **and** $\texttt{price}_{t-1} - \texttt{SMA}_{t-1} > \Delta_{SMA}$ **then**
    $\texttt{isMeanReverted} \leftarrow$ **True**
end if
if $\texttt{RSI}_t \leq \theta_{RSI}$ **then**
    $\texttt{isOverSold} \leftarrow$ **True**
end if
if `isUptrend` **and** `isMeanReverted` **and** `isOverSold` **then**
    Long asset
else
    Liquidate asset
end if

According to our strategy, we will go long on an asset if the following conditions are satisfied:

1. The current price of the asset is above its 200 day SMA, indicating that an uptrend is occuring.
2. The current price is close enough to its SMA and the previous price is not as close to its SMA. This shows that mean reversion has occurred.
3. 14 Day RSI value is equal to or lower than 30. The asset is oversold and there is potential for the price to increase.

If the above conditions are not satisfied for an asset, we will liquidate its holdings. Furthermore, we decided to only take long positions as this allows us to reduce the risks with short selling, in the scenario that the price does not decrease and we will not be able to buy back at a lower price.

4 Backtesting Results

Backtesting was done on both baseline strategy and our CITF strategy with the following properties:

1. The backtest period is from 1 January 2015 to 1 January 2020. We could not extend the period further due to QuantConnect's limit of 10,000 trades in a single backtest. We chose this period since it has both bull and bear periods which we want to test against our strategy. Based on SPY's performance, the bull period found is 1 January 2017 to 1 January 2018, and the bear period is 1 October 2018 to 24 December 2018.
2. The starting portfolio is $100,000 cash and no equity holdings.
3. The performance benchmark is the SPY ETF.

The figures on the following subsections show the portfolio values using our strategies against the SPY ETF benchmark. The black horizontal line on each plot is to show the starting $100,000 cash of the portfolio.

4.1 Overall Period: 1 January 2015 - 1 January 2020

Figure 1: Returns of baseline TF (red) against SPY benchmark (blue) on overall period.

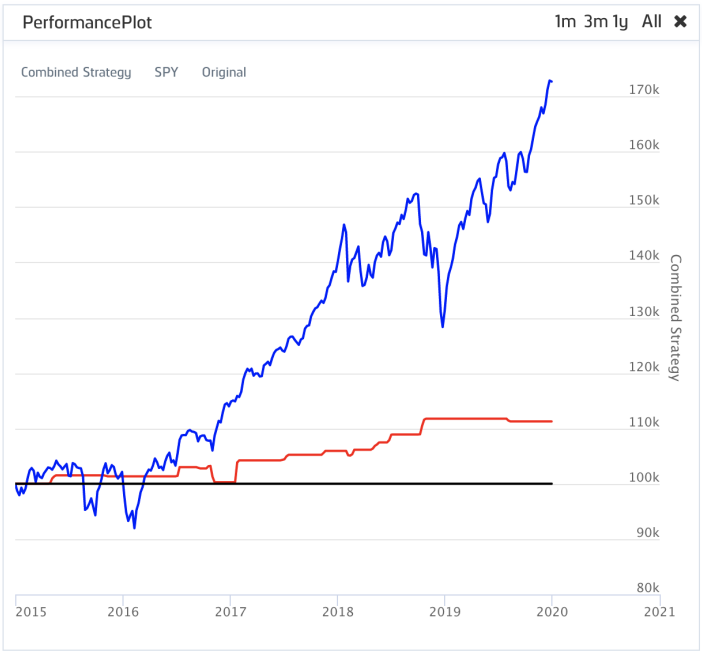

Figure 2: Returns of CITF (red) against SPY benchmark (blue) on overall period.

Statistic	Baseline TF	CITF
Returns (Net profit)	22.4%	11.25%
Alpha	0.005	0.014
Sharpe ratio	0.366	0.855
Average Win	0.11%	0.82%
Average Loss	-0.13%	-0.79%
Drawdown	25.2%	2.9%

Table 1: Backtest statistics of Baseline TF vs CITF on overall period.

Both the baseline TF strategy and CITF give positive alphas. In terms of returns, the baseline TF has higher returns but CITF has a higher average win. However, both strategies had a lower return than the benchmark.

4.2 Bull Period: 1 January 2017 - 1 January 2018

Figure 3: Returns of baseline TF (red) against SPY benchmark (blue) on bull period.

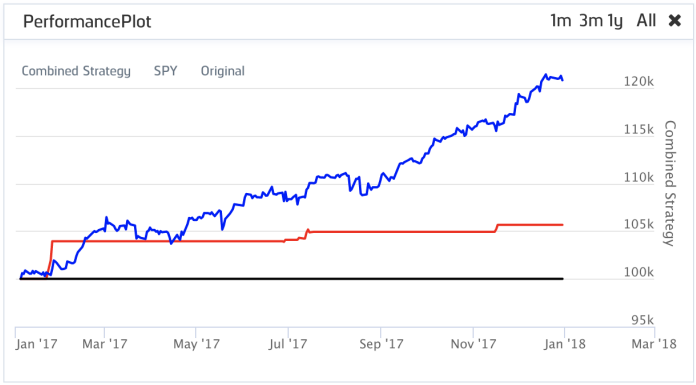

Figure 4: Returns of CITF (red) against SPY benchmark (blue) on bull period.

Statistic	Baseline TF	CITF
Returns (Net profit)	39.017%	5.65%
Alpha	0.093	0.035
Sharpe ratio	3.24	1.624
Average Win	0.35%	1.16%
Average Loss	-0.15%	-0.23%
Drawdown	5.1%	0.3%

Table 2: Backtest statistics of Baseline TF vs CITF on bull period.

In terms of returns, the baseline strategy performs better than both CITF and the benchmark in the bull period. The baseline strategy also has a high Sharpe ratio of 3.24 which is higher than CITF's 1.624 Sharpe ratio.

4.3 Bear Period: 1 October 2018 - 24 December 2018

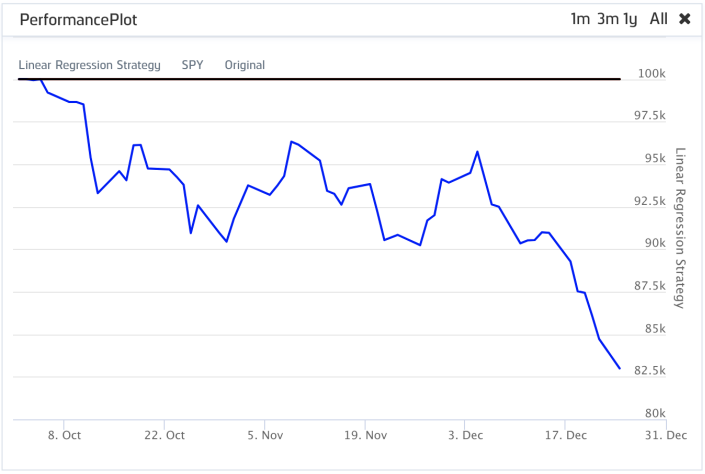

Figure 5: Returns of baseline TF (red) against SPY benchmark (blue) on bear period.

Figure 6: Returns of CITF (red) against SPY benchmark (blue) on bear period.

Statistic	Baseline TF	CITF
Returns (Net profit)	0%	2.62%
Alpha	0	0.101
Sharpe ratio	0	2.653
Average Win	0%	0.87%
Average Loss	0%	0%
Drawdown	0%	0.2%

Table 3: Backtest statistics of Baseline TF vs CITF on bear period.

For the bear period, the baseline TF strategy did not make any trades due to conditions not being met. Conversely, CITF made positive returns during this period when even the SPY benchmark faced losses.

5 Discussion

This section will provide an analysis on the backtesting results of Section 4.

5.1 Backtesting Analysis

In the overall period, despite CITF giving a lower return than the baseline TF strategy, its Sharpe ratio is found to be higher. This is a strong evidence of CITF's risk being lower than the baseline. There is a possibility that this is due to the CITF strategy excluding shorting decisions, whereas the baseline strategy implements shorting. Shorting increases our risk in the case that we predicted the trend as a downtrend wrongly and the price increases infinitely instead.

Another observation is that both the baseline strategy in the bull period and CITF in the bear period have unusually high Sharpe ratios. The drawdown ratio for these periods are also relatively low, with values of 0.093 and 0.101 respectively. Although this is an impressive statistic improvement, there is a possible overfitting issue despite both are backtested on different periods and assets. To resolve this, the backtest period could have been increased a couple of more decades further back to capture a more accurate backtest result.

Examining transaction costs, CITF manages to have a lower transaction cost compared to the baseline TF strategy. The overall period transaction cost for the baseline strategy is $2,139.51, while the overall period transaction cost for CITF is $440.96. This is because our combined strategy had more specific conditions to come to a trade, which resulted in lesser trades. Consequently, this also results in a lower return value for the CITF strategy which takes less advantage of the bull periods on the backtest.

5.2 Handling Overfitting

One of the drawbacks of the model we proposed is the use of fixed constants for threshold values. To perform our backtests, we manually tune our thresholds (e.g. mean reversion tolerance, RSI values) mainly through following general conventions for similar strategies. This may have caused the backtest feedback on QuantConnect to classify our model as overfit. This is a common issue for people in the QuantConnect community, where the parameters of the strategy are fine-tuned to fit the noise of the backtesting data. This can be avoided by running backtests on different data or in crisis conditions.

In practice, there are many ways to handle overfitting when building strategies. One of the methods to find optimal values of parameters is the Particle Swarm Optimisation (PSO) algorithm, which is a population-based optimisation technique to solve non-linear constraints in trading strategies. In this case, it will find the optimal threshold values for the current backtest period to optimise certain metrics, e.g. finding the highest Sharpe Ratio or investment returns.

5.3 Risk Control

Our improved CITF model has also shown to have a rather conservative risk management approach. This is due to the additional conditions in the algorithm as explained in Section 3.2, which causes trading to occur less often than in the baseline strategy.

For instance, we investigate two of our primary conditions to execute a trade: 1) the RSI values, and 2) the 200 day SMA. The said values are given in Figure 7 and 8 below.

Figure 7: Price of SPY (blue) against its 200 day SMA (red) during the overall period.

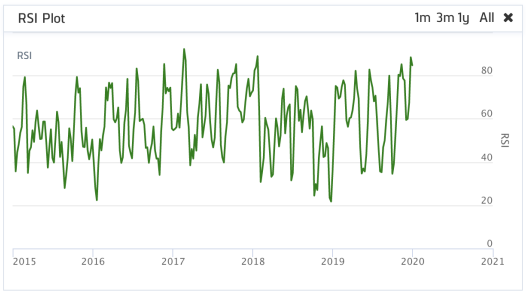

Figure 8: RSI value of SPY during the overall period.

From Figure 7 and 8, we found that when our RSI values for SPY were below 30, i.e. oversold, the current price of SPY was rarely above its 200 day SMA. Hence, our combined strategy does not trade in those moments since both conditions are not met and this is a reason of why CITF executes less trades than the baseline TF. An alternative to resolve this issue is to check for golden crosses instead of checking for 200 day SMA crossovers. A golden cross is a popular indicator to determine if a market is bullish by evaluating the change of its short and long moving averages.

Additionally, as discussed in Section 5.1, our CITF combined strategy only considers taking long positions, unlike the baseline TF that can take both long and short positions as part of the strategy. While reducing risk was one of the upsides of our strategy, we seem to have reduced it to the point of affecting our returns. A suggestion to improve this is on the periods where it is not ideal for us to go long on the asset, instead of liquidating that asset and just holding onto cash, we could use the cash and invest into indices such as SPY. This way, we can still potentially get returns without compromising volatility.

6 Conclusion

In this paper, we propose an improved trend following (TF) trading strategy — the Contrarian Indicator Trend Following (CITF) — which is built on top of a baseline Linear Regression TF strategy. Our improvements focus on implementing a Relative Strength Index (RSI) indicator which is a popular signal for contrarian trading strategies. Our backtesting results show that CITF has lower trading risks compared to the baseline TF, shown by a higher Sharpe ratio despite a lower return. However, overfitting is suspected to have influenced this result. This overfitting issue may be reduced with limiting the number of parameters and opting for optimisation algorithms (e.g. PSO) for selecting threshold values.

For further work, TF strategies can always be extended and combined with other indicators to achieve better returns and risk metrics. Current trading researches are also exploring machine learning methods to identify market situations and optimise trade timings. Furthermore, the backtests can be done for further decades back to capture a more accurate performance evaluation.

References

Auer, B. R. (2021). Have trend-following signals in commodity futures markets become less reliable in recent years? *Financial Markets and Portfolio Management*, 1–21.

Bailey, D. H., Borwein, J., Lopez de Prado, M., & Zhu, Q. J. (2016). The probability of backtest overfitting. *Journal of Computational Finance, forthcoming*.

Burghardt, G. (2010). Measuring the impact of trend following in the cta space. *Opalesque interview*.

Clenow, A. F. (2012). *Following the trend: diversified managed futures trading*. John Wiley & Sons.

Covel, M. (2004). *Trend following: how great traders make millions in up or down markets*. FT Press.

Hurst, B., Ooi, Y. H., & Pedersen, L. H. (2017). A century of evidence on trend-following investing. *The Journal of Portfolio Management*, *44*(1), 15–29.

Hutchinson, M. C., & O'Brien, J. (2014). Is this time different? trend-following and financial crises. *The Journal of Alternative Investments*, *17*(2), 82–102.

Jegadeesh, N., & Titman, S. (2011). Momentum. *Annu. Rev. Financ. Econ.*, *3*(1), 493–509.

Kaabar, S. (2021). *Using contrarian indicators in trend-following systems.* `https://medium.com/geekculture/using-contrarian-indicators-in-trend-following-systems-5f658f4d9546`. ([Online; accessed 13-Oct-2021])

Lempérière, Y., Deremble, C., Seager, P., Potters, M., & Bouchaud, J.-P. (2014). Two centuries of trend following. *arXiv preprint arXiv:1404.3274*.

Liu, J., Si, Y.-W., Zhang, D., & Zhou, L. (2018). Trend following in financial time series with multi-objective optimization. *Applied Soft Computing*, *66*, 149–167.

Mean reversion. (n.d.). `https://www.cmcmarkets.com/en/trading-guides/mean-reversion`. ([Online; accessed 13-Oct-2021])

Wu, J. (2018). *Etf trend following algorithm.* `https://www.quantconnect.com/forum/discussion/3260/etf-trend-following-algorithm/p1`. ([Online; accessed 13-Oct-2021])

Zhao, M., & Si, Y.-W. (2020). Trend following with contrarian strategy and firefly optimization algorithm. In *2020 ieee/wic/acm international joint conference on web intelligence and intelligent agent technology (wi-iat)* (p. 818-824). doi: 10.1109/WIIAT50758.2020.00126

Ziemba, W. T. (2013). Is the 60–40 stock–bond pension fund rule wise? *The Journal of Portfolio Management*, *39*(2), 63–72. Retrieved from `https://jpm.pm-research.com/content/39/2/63` doi: 10.3905/jpm.2013.39.2.063

A QuantConnect Python Code for Baseline TF

The Python file is provided on baseline.py.

```
# The code is from Jing Wu and Shile Wen, posted at the following link
# https://www.quantconnect.com/forum/discussion/3260/etf-trend-following-algorithm/p1
# The strategy is to use Trend Following to make trades for ETFs


from math import ceil,floor
from datetime import datetime
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from System.Drawing import Color


class TrendFollowingAlgorithm(QCAlgorithm):


    def Initialize(self):
        self.SetStartDate(2018, 10, 1)
        self.SetEndDate(2018, 12, 24)
        self.SetCash(100000)
        self.lookback = int(252/2)
        self.profittake = 1.96 # 95% bollinger band
        self.maxlever = 0.9 # always hold 10% Cash
        self.AddEquity("SPY", Resolution.Daily)
        etfs = [
            # Equity
            'DIA', # Dow
            'SPY', # S&P 500
            # Fixed income
            'IEF', # Treasury Bond
            'HYG', # High yield bond
            # Alternatives
            'USO', # Oil
            'GLD', # Gold
            'VNQ', # US Real Estate
            'RWX', # Dow Jones Global ex-U.S. Select Real Estate Securities Index
            'UNG', # Natual gas
            'DBA', # Agriculture
        ]

        for etf in etfs:
            self.AddEquity(etf, Resolution.Daily)

        self.symbol_data = {} # stores the Rolling open and close data, as well as
            ↪ weights and stop prices

        self.PctDailyVolatilityTarget = 0.025 # target daily vol target in %

        # setting benchmark for graph
        self.SetBenchmark("SPY")
        # Variable to hold the last calculated SPY value
        self.lastSPYValue = None
        # Inital benchmark value scaled to be the same as portfolio starting cash
        self.SPYPerformance = self.Portfolio.TotalPortfolioValue
        # Performance plot
```

```
        PerformancePlot = Chart("PerformancePlot")
        PerformancePlot.AddSeries(Series("Linear Regression Strategy", SeriesType.Line,
            ↪ "", Color.Red))
        PerformancePlot.AddSeries(Series("SPY", SeriesType.Line,"", Color.Blue))
        PerformancePlot.AddSeries(Series("Original", SeriesType.Line, "", Color.Black))
        self.AddChart(PerformancePlot)

        # trailing stop
        self.Schedule.On(self.DateRules.EveryDay("SPY"), self.TimeRules.AfterMarketOpen
            ↪ ("SPY", 10), self.trail_stop)

        # perform calculations for asset weightings
        self.Schedule.On(self.DateRules.EveryDay("SPY"), self.TimeRules.AfterMarketOpen
            ↪ ("SPY", 28), self.compute_regression_asset_weightings)

        # rebalance the portfolio according to calculations
        self.Schedule.On(self.DateRules.EveryDay("SPY"), self.TimeRules.AfterMarketOpen
            ↪ ("SPY", 30), self.rebalance)

        self.curr_day = -1

        # update closing price data
        self.Schedule.On(self.DateRules.EveryDay("SPY"), self.TimeRules.
            ↪ BeforeMarketClose("SPY", 1), self.update_closes)

        # plot graph
        self.Schedule.On(self.DateRules.EveryDay("SPY"), self.TimeRules.
            ↪ BeforeMarketClose("SPY", 1), self.plotting_performance)

    def update_closes(self):
        # updates closing price data

        for symbol, sd in self.symbol_data.items():
            if self.CurrentSlice.Bars.ContainsKey(symbol):
                sd.UpdateClose(self.CurrentSlice.Bars[symbol].Close)

    def OnData(self, data):
        # updates the SymbolDatas with daily opening prices

        if self.curr_day == self.Time.day:
            return

        self.curr_day = self.Time.day

        for symbol, sd in self.symbol_data.items():
            if data.Bars.ContainsKey(symbol):
                sd.UpdateOpen(data.Bars[symbol].Open)

    def OnSecuritiesChanged(self, changed):
        # add and remove SymbolDatas to our dict
        for security in changed.AddedSecurities:
            self.symbol_data[security.Symbol] = SymbolData(self, security.Symbol, self.
                ↪ lookback)

        for security in changed.RemovedSecurities:
            self.symbol_data.pop(security.Symbol, None)

    def calc_vol_scalar(self):
```

```
    # calculate the volatility scale factors for each ticker

    processed_sd = {symbol.Value: list(sd.open_data)[::-1] for symbol, sd in self.
        ↪ symbol_data.items()}

    df_price = pd.DataFrame(processed_sd)
    rets = np.log(df_price).diff().dropna()
    lock_value = df_price.iloc[-1]

    # Exponentially-weighted moving std
    price_vol = rets.ewm(halflife=20,ignore_na=True, min_periods=0, adjust=True).
        ↪ std(bias=False).dropna()

    volatility_scalar = self.PctDailyVolatilityTarget / price_vol.iloc[-1]

    return volatility_scalar

def compute_regression_asset_weightings(self):
    # compute asset weightings

    A = range( self.lookback + 1 )
    for symbol, sd in self.symbol_data.items():
        if not sd.IsReady:
            continue

        prices = list(sd.open_data)[::-1] # undo reverse order of RWs

        # volatility
        std = np.std(prices)
        # Price points to run regression
        Y = prices
        # Add column of ones so we get intercept
        X = np.column_stack([np.ones(len(A)), A])
        if len(X) != len(Y):
            length = min(len(X), len(Y))
            X = X[-length:]
            Y = Y[-length:]
            A = A[-length:]
        # Creating Model
        reg = LinearRegression()
        # Fitting training data

        reg = reg.fit(X, Y)
        # run linear regression y = ax + b
        b = reg.intercept_
        a = reg.coef_[1]

        # Normalized slope
        slope = a / b *252.0
        # Currently how far away from regression line
        delta = Y - (np.dot(a, A) + b)
        # Don't trade if the slope is near flat (at least %7 growth per year to
            ↪ trade)
        slope_min = 0.252

        # Long but slope turns down, then exit
        if sd.weight > 0 and slope < 0:
            sd.weight = 0
```

```
        # short but slope turns upward, then exit
        if sd.weight < 0 and slope > 0:
            sd.weight = 0

        # Trend is up
        if slope > slope_min:

            # price crosses the regression line
            if delta[-1] > 0 and delta[-2] < 0 and sd.weight == 0:
                sd.stopprice = None
                sd.weight = slope
            # Profit take, reaches the top of 95% bollinger band
            if delta[-1] > self.profittake * std and sd.weight > 0:
                sd.weight = 0

        # Trend is down
        if slope < -slope_min:

            # price crosses the regression line
            if delta[-1] < 0 and delta[-2] > 0 and sd.weight == 0:
                sd.stopprice = None
                sd.weight = slope
            # profit take, reaches the top of 95% bollinger band
            if delta[-1] < self.profittake * std and sd.weight < 0:
                sd.weight = 0


def rebalance(self):
    # rebalance portfolio

    vol_mult = self.calc_vol_scalar()
    no_positions = len([1 for _, sd in self.symbol_data.items() if sd.weight != 0])

    for symbol, sd in self.symbol_data.items():
        if not sd.IsReady:
            continue
        if sd.weight == 0:
            self.Liquidate(symbol)
        elif sd.weight > 0:
            self.SetHoldings(symbol, (min(sd.weight, self.maxlever)/no_positions)*
                ↪ vol_mult[symbol.Value])
        elif sd.weight < 0:
            self.SetHoldings(symbol, (max(sd.weight, -self.maxlever)/no_positions)*
                ↪ vol_mult[symbol.Value])

def trail_stop(self):
    for symbol, sd in self.symbol_data.items():
        if not sd.IsReady:
            continue
        mean_price = np.mean(list(sd.close_data))
        # Stop loss percentage is the return over the lookback period
        stoploss = abs(sd.weight * self.lookback / 252.0) + 1 # percent change per
            ↪ period
        if sd.weight > 0 and sd.stopprice is not None:
            if sd.stopprice is not None and sd.stopprice < 0:
                sd.stopprice = mean_price / stoploss
            else:
```

```
                sd.stopprice = max(mean_price / stoploss, sd.stopprice)
                if mean_price < sd.stopprice:
                    sd.weight = 0
                    self.Liquidate(symbol)

            elif sd.weight < 0 and sd.stopprice is not None:
                if sd.stopprice is not None and sd.stopprice < 0:
                    sd.stopprice = mean_price * stoploss
                else:
                    sd.stopprice = min(mean_price * stoploss, sd.stopprice)
                    if mean_price > sd.stopprice:
                       sd.weight = 0
                       self.Liquidate(symbol)

            else:
                sd.stopprice = None

    def plotting_performance(self):

        # benchmark = 0

        # for symbol in self.symbols:
        # benchmark += self.Securities[symbol].Close

        # benchmark = benchmark / len(self.symbols)

        # Plot performance graph
        benchmark = self.Securities["SPY"].Close

        # Update SPY performance if it's not the first periof performance
        if self.lastSPYValue is not None and self.lastSPYValue != 0:
           self.SPYPerformance = self.SPYPerformance * (benchmark/self.lastSPYValue)

        # store today's benchmark close price for use tomorrow
        self.lastSPYValue = benchmark

        # make our plots
        self.Plot("PerformancePlot", "Linear Regression Strategy", self.Portfolio.
            ↪ TotalPortfolioValue)
        self.Plot("PerformancePlot", "SPY", self.SPYPerformance)
        self.Plot("PerformancePlot", "Original", 100000)

class SymbolData:
    def __init__(self, algorithm, symbol, lookback):
        self.open_data = RollingWindow[float](lookback)
        self.close_data = RollingWindow[float](3)

        hist = algorithm.History(symbol, lookback, Resolution.Daily).loc[symbol]
        for _, row in hist.iterrows():
            self.open_data.Add(row.open)
            self.close_data.Add(row.close)

        self.stopprice = 0
        self.weight = 0

    def UpdateOpen(self, value):
        self.open_data.Add(value)
```

```
    def UpdateClose(self, value):
        self.close_data.Add(value)

    @property
    def IsReady(self):
        return self.open_data.IsReady and self.close_data.IsReady
```

B QuantConnect Python Code for CITF

The Python file is provided on citf.py.

```
# The following code is our implemented improvement for Trend Following.
# We used the 200 day Simple Moving Average, 14 period RSI and
# current prices of the assets

import numpy as np
from datetime import datetime
from System.Drawing import Color

class BasicTemplateAlgorithm(QCAlgorithm):

    def Initialize(self):

        self.SetStartDate(2015, 1, 1)
        self.SetEndDate(2020, 1, 1)
        self.SetCash(100000)
        self.data = {}
        self.macd = {}
        self.rsi_values = {}

        fastPeriod = 12
        slowPeriod = 26
        signalPeriod = 9
        period = 200

        self.SetWarmUp(period)
        # self.symbols = ["SPY", "EFA", "BND", "VNQ", "GSG"]
        self.symbols = [
            # Equity
            'DIA', # Dow
            'SPY', # S&P 500
            # Fixed income
            'IEF', # Treasury Bond
            'HYG', # High yield bond
            # Alternatives
            'USO', # Oil
            'GLD', # Gold
            'VNQ', # US Real Estate
            'RWX', # Dow Jones Global ex-U.S. Select Real Estate Securities Index
            'UNG', # Natual gas
            'DBA', # Agriculture
        ]

        self.fast_sma = {}
        self.sma_history = {}
        self.price_history = {}
```

```
    for symbol in self.symbols:
        self.AddEquity(symbol, Resolution.Daily)
        self.data[symbol] = self.SMA(symbol, period, Resolution.Daily)
        self.rsi_values[symbol] = self.RSI(symbol, 14, MovingAverageType.Simple,
            ↪ Resolution.Daily)
        self.fast_sma[symbol] = self.SMA(symbol, 50, Resolution.Daily)
        self.sma_history[symbol] = RollingWindow[float](3)
        self.price_history[symbol] = RollingWindow[float](3)
        self.price_history[symbol].Add(-1)

        self.macd[symbol] = self.MACD(symbol, fastPeriod, slowPeriod, signalPeriod,
            ↪  MovingAverageType.Exponential, Resolution.Daily)
        self.__previous = datetime.min

    # charts
    # plot sma, current price to see the crossovers
    # plot rsi to see how it compares with current price
    # plot the 70, 30 rsi lines to see the overbought, solds

    PricePlot = Chart("Price␣Plot")
    PricePlot.AddSeries(Series("200␣Day␣SMA", SeriesType.Line,"", Color.Red))
    PricePlot.AddSeries(Series("Current␣Price", SeriesType.Line,"", Color.Blue))

    RSIPlot = Chart("RSI␣Plot")
    RSIPlot.AddSeries(Series("RSI", SeriesType.Line,"", Color.Green))
    RSIPlot.AddSeries(Series("Overbought", SeriesType.Line,"", Color.Black))
    RSIPlot.AddSeries(Series("Oversold", SeriesType.Line, "", Color.Black))

    # setting benchmark for graph
    self.SetBenchmark("SPY")
    # Variable to hold the last calculated SPY value
    self.lastSPYValue = None
    # Inital benchmark value scaled to be the same as portfolio starting cash
    self.SPYPerformance = self.Portfolio.TotalPortfolioValue
    # Performance plot
    PerformancePlot = Chart("PerformancePlot")
    PerformancePlot.AddSeries(Series("Combined␣Strategy", SeriesType.Line,"", Color
        ↪ .Red))
    PerformancePlot.AddSeries(Series("SPY", SeriesType.Line,"", Color.Blue))
    PerformancePlot.AddSeries(Series("Original", SeriesType.Line, "", Color.Black))

    self.AddChart(PricePlot)
    self.AddChart(RSIPlot)
    self.AddChart(PerformancePlot)


def OnData(self, data):

    if self.IsWarmingUp: return

    tolerance = 0.0015
    isUptrend = []

    for symbol, sma in self.data.items():
        if not self.macd[symbol].IsReady: return

        # only once per day
        if self.__previous.date() == self.Time.date(): return
```

```
        holdings = self.Portfolio[symbol].Quantity

        self.sma_history[symbol].Add(sma.Current.Value)
        self.price_history[symbol].Add(self.Securities[symbol].Price)

        current_sma = sma.Current.Value
        current_price = self.Securities[symbol].Price

        if self.price_history[symbol][1] != -1:
            previous_sma = self.sma_history[symbol][1]
            previous_price = self.price_history[symbol][1]

        else:
            previous_sma = current_sma
            previous_price = current_price

        tolerance = 0.5

        # plot SPY
        if symbol == "SPY":
            self.plotting_info(symbol, sma)

        signalDeltaPercent = (self.macd[symbol].Current.Value - self.macd[symbol].
            ↪ Signal.Current.Value)/self.macd[symbol].Fast.Current.Value


        if (self.Securities[symbol].Price > sma.Current.Value
        and current_price - current_sma <= tolerance
        and previous_price - previous_sma > tolerance
        and self.rsi_values[symbol].Current.Value < 30):
            self.SetHoldings(symbol, 1/len(self.symbols))
        else:
            self.Liquidate(symbol)

        self.__previous = self.Time

    # plot performance
    self.plotting_performance()

def plotting_info(self, symbol, sma):
    self.Plot("Price␣Plot", "Current␣Price", float(self.Securities[symbol].Price))
    self.Plot("Price␣Plot", "200␣Day␣SMA", float(sma.Current.Value))
    self.Plot("RSI␣Plot", "RSI", float(self.rsi_values[symbol].Current.Value))

def plotting_performance(self):

    # Plot performance graph
    benchmark = self.Securities["SPY"].Close

    # Update SPY performance if it's not the first periof performance
    if self.lastSPYValue is not None:
       self.SPYPerformance = self.SPYPerformance * (benchmark/self.lastSPYValue)

    # store today's benchmark close price for use tomorrow
    self.lastSPYValue = benchmark

    # make our plots
```

```
self.Plot("PerformancePlot", "Combined␣Strategy", self.Portfolio.
    ↪ TotalPortfolioValue)
self.Plot("PerformancePlot", "SPY", self.SPYPerformance)
self.Plot("PerformancePlot", "Original", 100000)
```