Multi-Period Analysis of Industry Standard Trade and Custom Strategies Using Quantstrat for Nasdaq ETF

CFRM 522 Final Project

Gustavo Zambrana

June 07, 2021

Abstract

This document provides an analysis evaluating six-different trading strategies utilizing the systematic approach discussed in CFRM 522, plus the author's experience backtesting trading strategies. The primay goal of this artifact is to answer the question, are any of these strategies worth trading with Nasdaq ETF data? In addition, this paper presents and uses a distintive methodology to analyze walk-forward simulation results from the one presented in class.

An indirect result of this project, it is that it provides in-depth insights into performing backtesting with quantstrats and the issues encountered (particularly optimization and walk-forward analysis) and identified work-arounds. Finally, a project developed library provides the initial foundation to conduct similar studies for any set of strategies, time periods, with the ultimate goal to minimize the large amount of boiler-template code and intrisicaties of configuring quantstrat.

Contents

1 Introduction

Trading, it is a topic that evokes passion, superstition, and challenges even the most scientific oriented person to put their emotions aside. Systematic Trading provides a scientific approach to evaluate any trading hypothesis, using the well established principles use in other STEM sciences.

When choosing a topic for research in trading, the options are limitless. This paper focused on the analysis of several industry well-known trade strategies, that the reader can encounter in most financial news, books, among others. Specifically, the following strategies were selected: Simple Moving Average (SMA), Moving Average Convergence Divergence (MCAD), Boiler Bands (BB), and Relative Strength Index (RSI). In addition, two less known, but intriguing strategies were integrated into the project: Average Directional Movement Index (ADX) and David Varadi Oscillator (DVO).

The selected strategies served several purposes:

1. Include standard trading ideas that are commonly encountered
2. Implementation of two trading strategies with different amount of effort to demonstrate the wiring required in quantstrat, and to explore momentum and mean reversion algorithms which provide different methodology than RSI and BBs. Custom Trade Strategies:

   i) ADX is an indicator provided by the R package TTR; however, it requires non-standard configuration, as there are three hyper parameters that must be accounted for with composite rules. ADX indicator is momentum based strategy. Detail documentation for the concepts for this strategy can be further explore online, a good starting point is: **adx-indicator-trading-strategy**

   ii) The second custom strategy (DVO), it is not provided in the TTR R package. The strategy is is a mean-reverting indicator, implementation in quantstrat required custom implementation. Detail documentation for the concepts for this strategy can be read online, a good starting point is: **DV Indicators' review**

   The code for both of these custom strategies is in the project library, see adx_Setup() and dvo_Setup()

3. Compare the relative strength and weakness of each trade strategy versus the well-known Nasdaq ETF (QQQ)

The approach for the evaluation follows the systematic analytic process from CFRM-522, and it is implemented sequentially in this document. In addition, additional steps are include to further stress the selected strategies, including:

1. Multiple-Regime Analysis

   One of the biggest factors, if not the biggest, evaluating a trading strategy, it is to determine the performance across different market regimes. One recent market regime we have observed is the COVID Regime from 2021. For the analysis, two time periods were selected:

i) 2012-2017

This period did not experience dramatic down years, across multiple years, but provides a sufficient reference for analysis with about 1250 trading days

ii) 2005-2017

This market period, includes multiple down years in the market, 2007-2010, encompassing the Financial Crisis. The length of the period is 12-years, and provides a good frame of reference for comparions to the smaller and most recent period.

2. Optimization of n number of hyper parameters per strategy

   Each strategy is optimized with the hyper parameters that it is configured with. In addition, permutations of the optimizations were executed as follows:

   i) Optimized strategies hyper parameters without stop loss or trail loss configuration
   ii) Optimized strategies hyper parameters with Stop Loss defaulted
   iii) Optimized strategies and Stop Loss hyper parameters
   iv) Optimized strategies and Trail-Stop Loss hyper parameters

3. Walk Forward Analysis across multiple training and testing windows

   This step allows identification of best possible training and test windows for each strategy, some strategies are expected to require different training and test windows. The most optimum windows are selected based on the methodology presented in the walk-forward analysis section.

4. Walk Forward Analysis across different Market Regimes

   In order to truly test trading strategies, it is the opinion of the author, that one also needs to test across many different market regimes, and specifically periods of multiple down years, that stressed the financial markets. Evaluating the performance of the strategies against such period should provide insight on how trading strategies may respond during highly volatile and down markets.

5. Monte Carlos Analysis within the bounds explained in CFRM 522 will be utilized mainly for verifying the performance of non-optimized in-sample strategies

   Monte Carlos Analysis as derived and implemented in class consisted of in-sample testing of the strategies. The basic idea is to draw random samples from the data created during backtests and/or optimizations in the in-sample time period. This methodology assumes independent, identically distributed data, something that it is not the case with financial market data. Thus, this research utilized Monte Carlos analysis as implemented in quantstrat for in-sample validation, and relied primarily on walk-forward-analysis to draw overall conclusions due to his out-sample capabilities.

2 Environment Setup

2.1 Required Libraries

```
library(quantmod)
library(quantstrat)
library(rgl)
library(reshape2)
library(dplyr)
library(tibble)
library(parallel)
library(rstudioapi)
library(knitr)
library(readr)
library(magrittr)
```

2.2 Multi-Core

Configuring R for Multi-Core execution is strongly recommended to perform the analysis in a realistic time period, but it must be done with care. The analysis was conducted in a MAC with 32G of RAM, and 16 CPU cores. The number of CPU cores configured is 3, increasing to a higher number consumes the entire memory when executing WFA simulations, each core spawns an R session, which tends to utilize about 800M-1.5GB of RAM depending on the length of the time period within a few minutes, after repeated executions or within one RStudio will certainly abort.

2.3 Project R Library

A higher label library was developed to wrap quantstrat to provide a consistent API to execute all analysis, a type of facade object oriented pattern, but function based given R's non Object Oriented philosophy. The library also has all the utilities developed to process data, summarized data, etc. The complete library is included in this paper as an appendix. The file is sourced at this point, but not included to set up the environment ready for executing the analysis.

2.4 Market Data

Two periods of market data are selected. It is noted that in order to perform the analysis in quantstrat, the data for the period being analyzed must match the data retrieved. Initially, the 12-years of data were retrieved, but quantstrat would error out when setting up the start date to a later time than the downloaded market data. Bypassing this error, required to download the data for the period just before the analysis of such period. The 12-year period of data is only used in the walk-forward-analysis, the wfa analysis is performed with downloaded data for 5-years, then the data it is downloaded for 12-years.

```
stock.st = c("QQQ")   #
currency("USD")
stock(stock.st, currency="USD",multiplier=1)
Sys.setenv(TZ="UTC")

# 5-year window
initDate = '2011-12-31'
startDate = '2012-01-01'
```

```
endDate = '2017-12-31'

# 12-year window
initDate_long_period  = '2004-12-31'
startDate_long_period = '2005-01-01'
endDate_long_period   = '2017-12-31'

getSymbols(stock.st,
           from=startDate,
           to=endDate,
           index.class="POSIXct",
           adjust=T)
```

3 Execution Guidelines

- System with minimum of 16 GBytes of RAM
- 16 CPU Cores
- Fresh-reboot and close all un-necessary applications
- Monitor System Memory
- Optional: Multi-Core CPU
  - Configured up to 3 or 4, R-sessions are spawned in parallel and appear to leak memory
- The WFA analysis conducted takes about 2-hours with the defined settings above
  - It was necessary to cache the results of WFA simulations in CSV files in order to iterate on development in RMarkdown
  - The results are loaded from csv files in the same directory where the project RMD file is executed
  - To re-run the wfa simulation just delete the csv files, only do if you have follow the guidelines above (renamed csv files, don't delete)
- RStudio and/or RMarkdown are not catching R-Chunk results, this behavior was observed and can be easily verify online.

4 Configuration

4.1 Trading Strategies

Six strategies are defined, and the configuration of each strategy is configured via nested R list. The hyper parameters for each strategy are managed through the nested list.

```
initEq = 1000000
supported_strategies = c("ADX",  "DVO", "RSI", "SMA", "BBAND", "MACD" )

# Defaults for stop loss and trailing losses for all strategies

stopLossPercDef     <- 0.02
trailingLossPercDef <- 0.03
stopLossRange <- seq(0.01, 0.05, by=0.01)  # 5 permutations
trailingRange <- seq(0.01, 0.06, by=0.02)   # 3 permutations

# Hyper Parameter defaults for all Strategies, one entry per strategy
```

```
strategies_def_arguments = list(
                          adx   = list( n                    = 41 ,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        nRange               = seq(40, 41, by=1),
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange ),
                          dvo   = list( navg                 = 2,
                                        percentlookback      = 126,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        navgRange            = seq(2, 6, by=2) ,
                                        percentlookbackRange = seq(120, 128, by=4),
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange ),
                          rsi   = list( n                    = 14,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        nRange               = seq(4, 10, by=2),
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange ),
                          sma   = list( nFast                = 10,
                                        nSlow                 = 100 ,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        fastRange            = seq(5, 15, by=5) ,
                                        slowRange            = seq(80, 120, by=20) ,
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange ),
                          bband = list( n                    = 20,
                                        sd                   = 2,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        n_range              = seq(10,50,by=20),
                                        sd_range             = seq(1.5,2.5,by=1),
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange ),
                          macd  = list( nFast                = 10,
                                        nSlow                = 40,
                                        sig                  = 9,
                                        stopLoss             = stopLossPercDef,
                                        trailLoss            = trailingLossPercDef,
                                        fastRange            = seq(12, 18, by=2) ,
                                        slowRange            = seq(24, 36, by=4) ,
                                        sigRange             = seq(8, 9)  ,
                                        stopLossRange        = stopLossRange,
                                        trailLossRange       = trailingRange )
                               )
```

4.2 Walk-Forward-Analysis

Default values for the single WFA period analysis. Multi-Window WFA configurations are defined withing the walkf-forward analysis section.

```
# WFA defaults

periodTrainLength <- 36
periodTestLength  <- 12
period_wfa        = 'months'
opt_metric        = "Net.Trading.PL"
```

5 In-Sample Backtests

All Strategies are executed with the runStrategies() function.

Inputs:

- strategies
  - list of strings identifying each strategy
- strategies-args
  - configuration of hyper parameters for each strategy
- append-portf-name
  - Portfolio Name is built dynamically based on the strategy name. This optional param, if specified, is concatenated to the name of the portfolio, i.e. portfolio_name = strategy_append-portf-name.
  - This option is useful to run permuations of an analysis and save the portfolios for retrieval and further analysis.
- run-type
  - type of quantstrat execution. Valid options ["regular", "paramset", "wfa"]
    - regular = basic back test, calls applyStrategy()
    - paramset = execute optimization of strategies via apply.paramset()
    - wfa = executes walk-forward-analysis
- initEq-var
  - initial portfolio equity
- stock.st-var
  - list of market data symbols
- initDate-var
  - initial date of back testing
- addStopLoss
  - add stop loss configuration rules, and use values configured in strategies_def_arguments per strategy
- addTrailStop
  - add trail stop loss configuration rules, and use values configured in strategies_def_arguments per strategy

Outputs:

- results object
  - results contain a nested list object with different outputs for the execution, depending on the run_type option.
  - Contains:
    - DataFrame summary of the results per strategy executed - all performance metrics from tradeStats are saved, with hyper paramaters for the execution.
    - Each strategy output is saved under its own portfolio name

5.1 Basic

- Execute backtest for all trade strategies

```
suppressWarnings( rm( non_opt_sum_results) )
non_opt_sum_results = runStrategies (strategies            = supported_strategies,
                                     strategies_args       = strategies_def_arguments,
                                     append_portf_name     = "A",
                                     run_type              = "regular",
                                     initEq_var            = initEq,
                                     stock.st_var          = stock.st,
                                     initDate_var          = initDate,
                                     addStopLoss           = FALSE,
                                     addTrailStop          = FALSE)
```

5.2 Basic with Stop Loss

- Execute backtest for all trade strategies with stop loss defaulted

```
suppressWarnings( rm( non_opt_sum_results_stopl) )
non_opt_sum_results_stopl = runStrategies (strategies      = supported_strategies,
                                     strategies_args       = strategies_def_arguments,
                                     append_portf_name     = "B",
                                     run_type              = "regular",
                                     initEq_var            = initEq,
                                     stock.st_var          = stock.st,
                                     initDate_var          = initDate,
                                     addStopLoss           = TRUE,
                                     addTrailStop          = FALSE)
```

5.3 Basic with Trail Stop Loss

- Execute backtest for all trade strategies with trail-stop loss defaulted

```
suppressWarnings( rm( non_opt_sum_results_trailloss) )
non_opt_sum_results_trailloss = runStrategies (strategies        = supported_strategies,
                                       strategies_args       = strategies_def_arguments,
                                       append_portf_name     = "C",
                                       run_type              = "regular",
                                       initEq_var            = initEq,
                                       stock.st_var          = stock.st,
                                       initDate_var          = initDate,
                                       addStopLoss           = FALSE,
                                       addTrailStop          = TRUE)
```

5.4 Basic Stop Loss and Trail Stop Loss

- Execute backtest for all trade strategies with stop and trail-stop loss defaulted

```
suppressWarnings( rm( non_opt_sum_results_bothloss) )
non_opt_sum_results_bothloss = runStrategies (strategies            = supported_strategies,
                                         strategies_args       = strategies_def_arguments,
                                         append_portf_name     = "D",
                                         run_type              = "regular",
                                         initEq_var            = initEq,
                                         stock.st_var          = stock.st,
                                         initDate_var          = initDate,
                                         addStopLoss           = TRUE,
                                         addTrailStop          = TRUE)
```

5.5 Summary Results

- The following summary tables provide the results of all backtests with the key trading statistics. This data is then review to draw conclusions.
- The number of plots is minimized due to the large number of permutations executed, only a few are selected to confirm visually the strategies.

Non-Optimized Trade Strategies Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_A	QQQ	4	2	319842.10	18.9865833	-195701.5	1.6343366
DVO_A	QQQ	477	182	422271.72	0.7247294	-783402.7	0.5390225
RSI_A	QQQ	227	64	912372.65	3.8585912	-499394.9	1.8269561
SMA_A	QQQ	29	14	125389.53	-3.0351425	-630884.4	0.1987520
BBAND_A	QQQ	89	37	777215.03	13.0430728	-221241.0	3.5129796
MACD_A	QQQ	51	25	72622.42	-2.5252943	-675615.2	0.1074908

Non-Optimized Trade Strategies with Stop-Loss hard-coded - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_B	QQQ	10	5	-23964.11	-1.9307088	-140058.88	-0.1711002
DVO_B	QQQ	491	214	354049.01	0.8273919	-606208.26	0.5840386
RSI_B	QQQ	258	105	153540.20	0.5768753	-273225.26	0.5619546
SMA_B	QQQ	29	14	604580.37	2.4532472	-183995.95	3.2858352
BBAND_B	QQQ	92	43	439743.95	5.2457362	-77762.72	5.6549455
MACD_B	QQQ	52	26	56386.89	0.7727170	-258296.95	0.2183026

Non-Optimized Trade Strategies with Trail-Loss hard-coded - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_C	QQQ	10	5	3513.617	0.3052987	-80816.12	0.0434767
DVO_C	QQQ	491	213	288400.805	0.6763507	-523598.35	0.5508054
RSI_C	QQQ	277	115	-206386.855	-0.7924728	-332996.23	-0.6197874
SMA_C	QQQ	30	15	-26577.101	-0.8917341	-176259.72	-0.1507837
BBAND_C	QQQ	93	44	253312.920	3.0486360	-91052.69	2.7820477
MACD_C	QQQ	52	26	31409.431	0.6959389	-140448.57	0.2236365

Non-Optimized Trade Strategies with Stop and Trail Loss hard-coded - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_D	QQQ	10	5	6194.00	0.5443579	-78135.74	0.0792723
DVO_D	QQQ	495	219	276844.31	0.6767032	-547389.46	0.5057538
RSI_D	QQQ	277	118	-21888.84	-0.1615809	-312470.96	-0.0700508
SMA_D	QQQ	30	15	15426.86	0.5659473	-144644.48	0.1066537
BBAND_D	QQQ	95	45	331273.08	3.9580611	-89458.16	3.7031063
MACD_D	QQQ	52	26	28571.98	0.6862876	-107741.91	0.2651891

The following behavior can be observed from the summary metrics above:

- For strategies DVO, RSI, SMA, BBAND, and MACD, using the default stop-loss reduces the Max-Drawdown

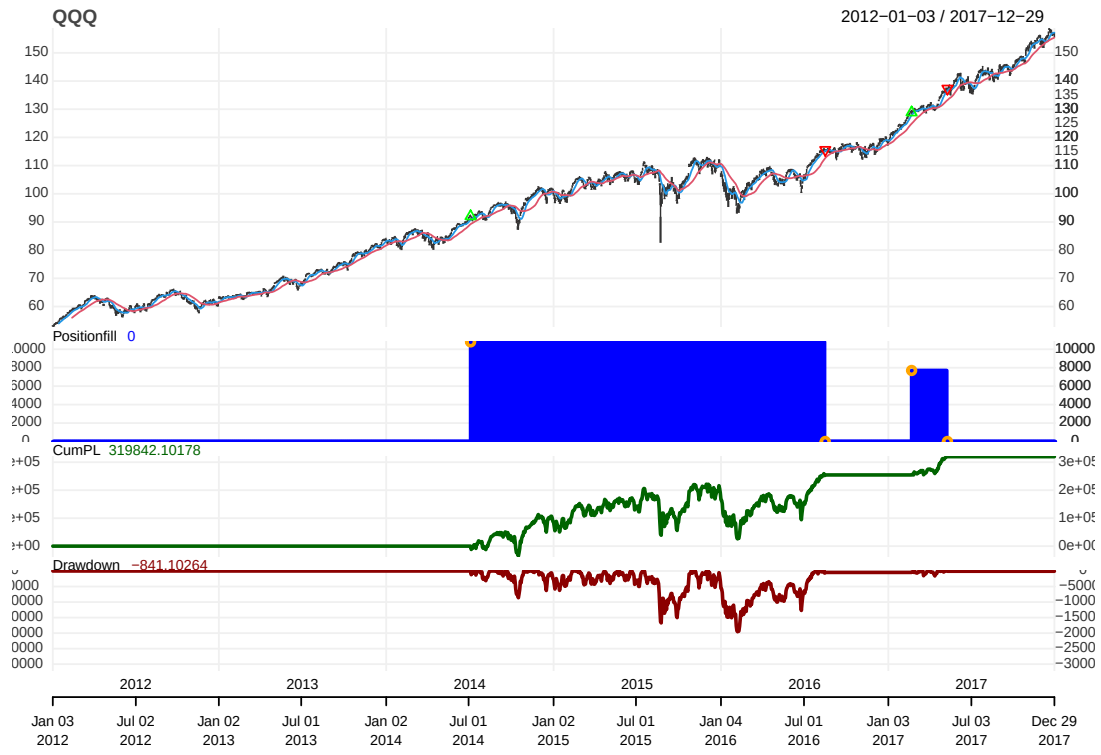
- For SMA, and RSI the reduction in Max-Drawdown indicates a significant reduction using stop loss rule.
- Putting a stop-loss on RSI reduces the large Max-Drawdown, but at the cost of a significant drop in the Net.Trading.PL. Given the initial large Max-Drawdown, it is best to keep this strategy with the stop-loss.
- The ADX strategy is the one that did not benefit from using either stop-loss or trail-stop loss. The strategy appears to be very selective about when it trades, and putting a stop, reduces the initial large Ann.Sharpe ratio to negative or almost zero. It is best to keep this strategy without any kind of stop signal.

5.6 Charts

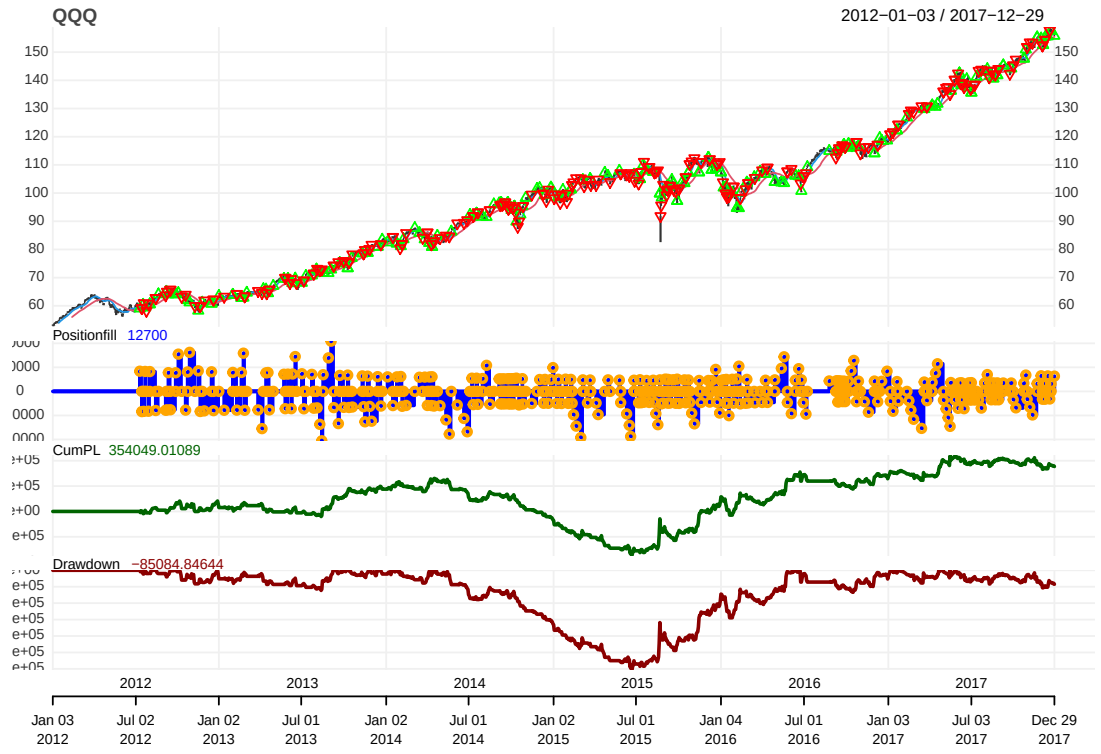
5.6.1 Strategies Chart Positions

The following trading charts represent the best non-optimized strategies, and primarily served as a visual verification that the strategies are functioning on the long and short-side:

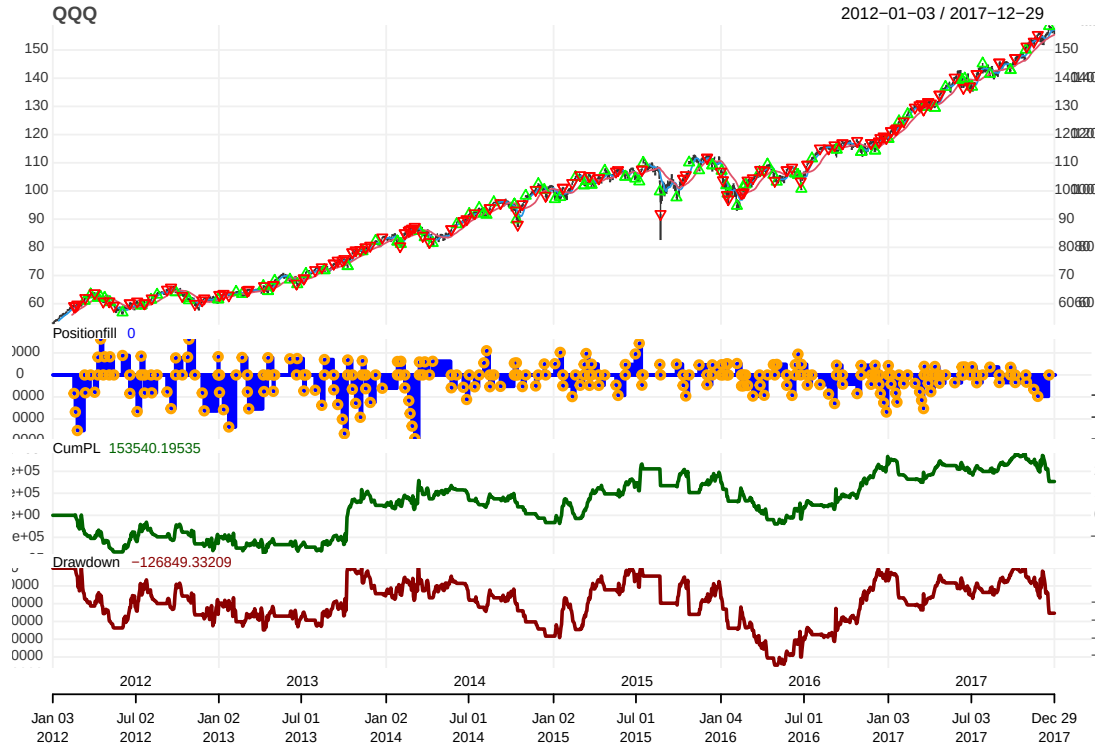
ADX



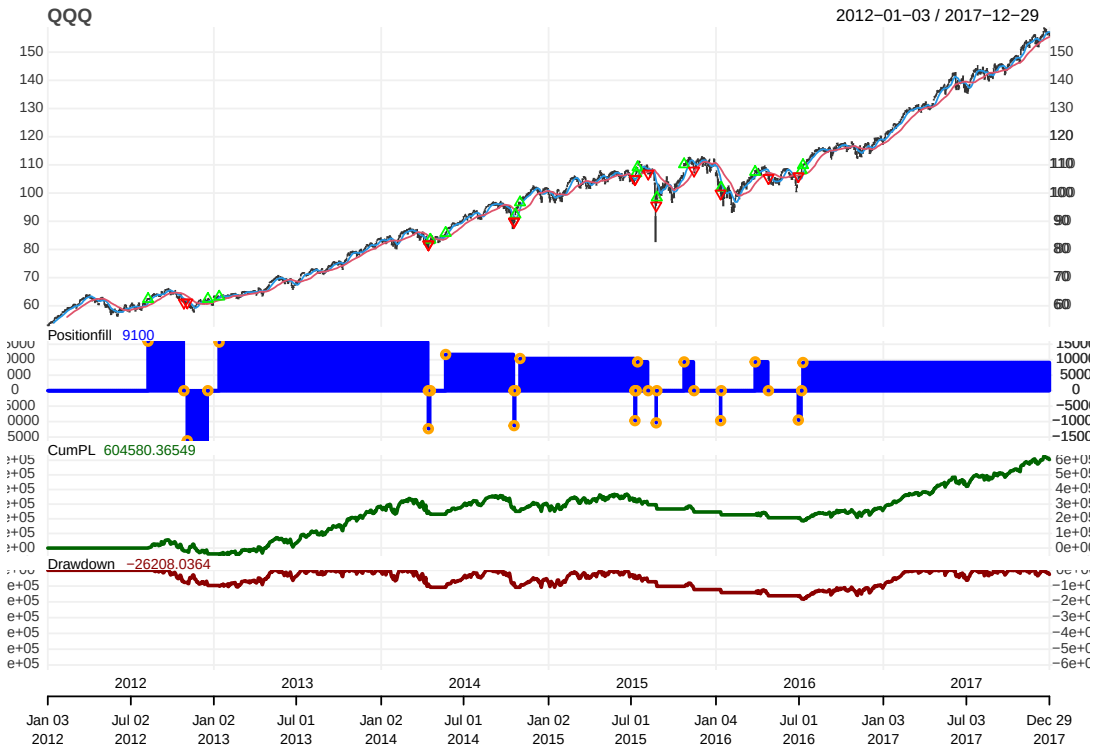
DVO


QQQ
2012−01−03 / 2017−12−29
Positionfill 12700
CumPL 354049.01089
Drawdown −85084.84644
2012
2013
2014
2015
2016
2017
Jan 03 2012
Jul 02 2012
Jan 02 2013
Jul 01 2013
Jan 02 2014
Jul 01 2014
Jan 02 2015
Jul 01 2015
Jan 04 2016
Jul 01 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

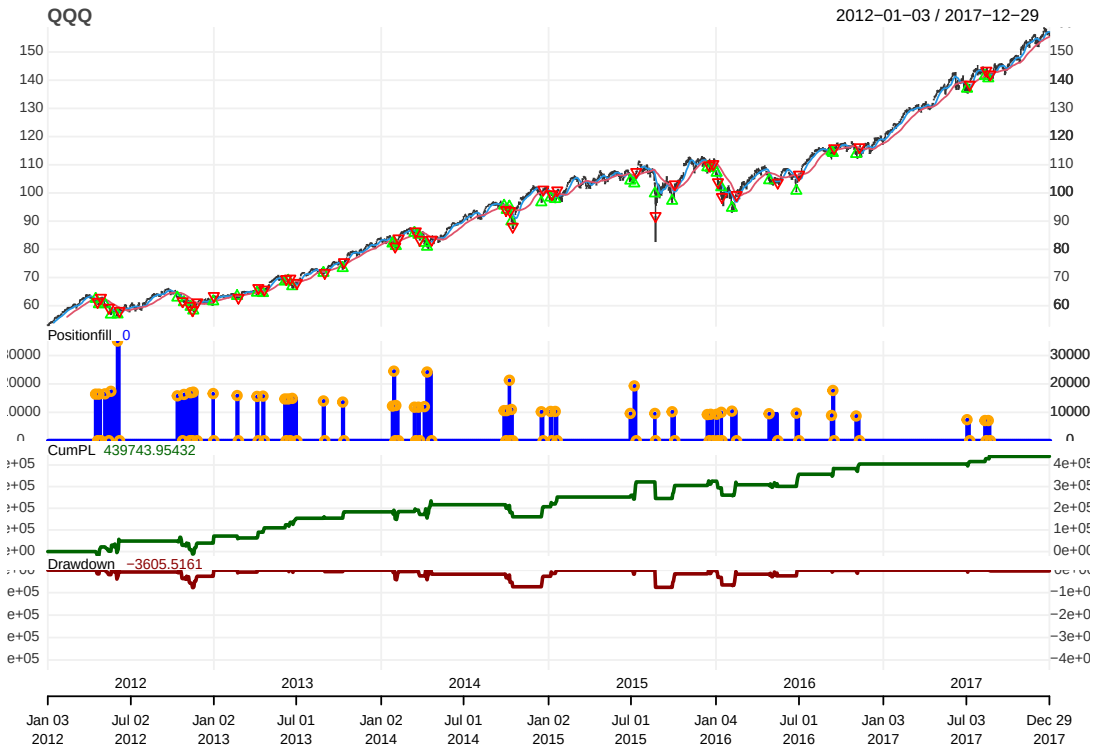
RSI


QQQ
2012−01−03 / 2017−12−29
Positionfill 0
CumPL 153540.19535
Drawdown −126849.33209
2012
2013
2014
2015
2016
2017
Jan 03 2012
Jul 02 2012
Jan 02 2013
Jul 01 2013
Jan 02 2014
Jul 01 2014
Jan 02 2015
Jul 01 2015
Jan 04 2016
Jul 01 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

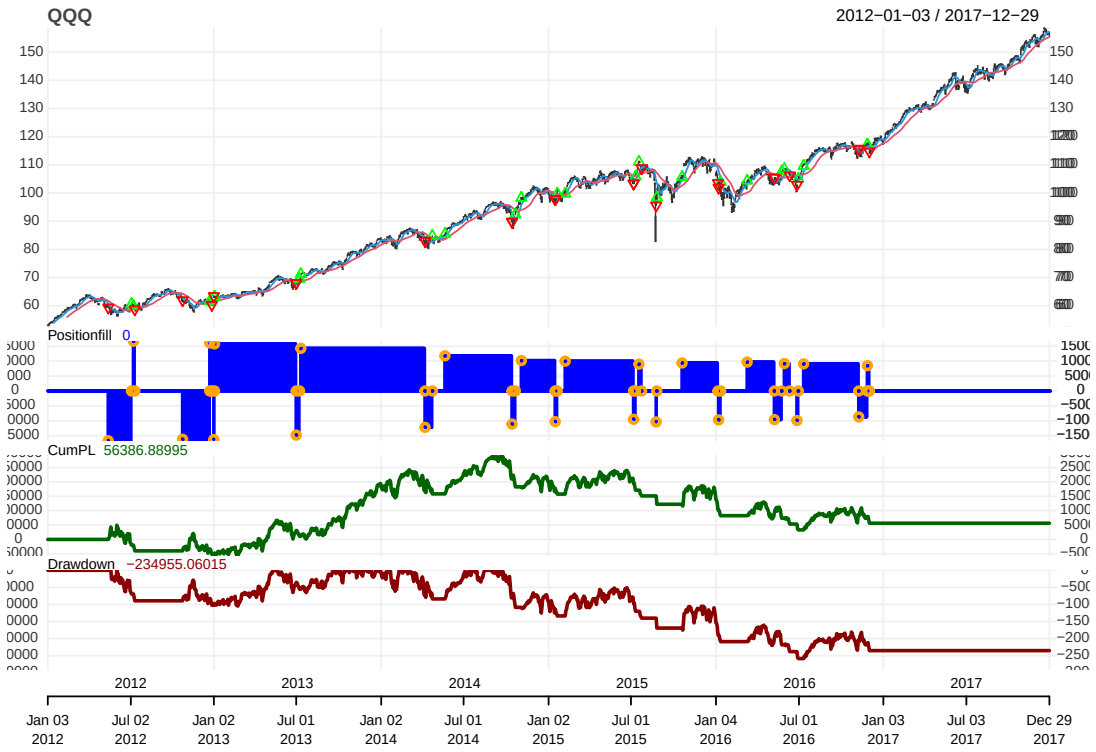
SMA


QQQ
2012−01−03 / 2017−12−29
Positionfill 9100
CumPL 604580.36549
Drawdown −26208.0364
2012
2013
2014
2015
2016
2017
Jan 03 2012
Jul 02 2012
Jan 02 2013
Jul 01 2013
Jan 02 2014
Jul 01 2014
Jan 02 2015
Jul 01 2015
Jan 04 2016
Jul 01 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

BBAND


QQQ
2012−01−03 / 2017−12−29
Positionfill 0
CumPL 439743.95432
Drawdown −3605.5161
2012
2013
2014
2015
2016
2017
Jan 03 2012
Jul 02 2012
Jan 02 2013
Jul 01 2013
Jan 02 2014
Jul 01 2014
Jan 02 2015
Jul 01 2015
Jan 04 2016
Jul 01 2016
Jan 03 2017
Jul 03 2017
Dec 29 2017

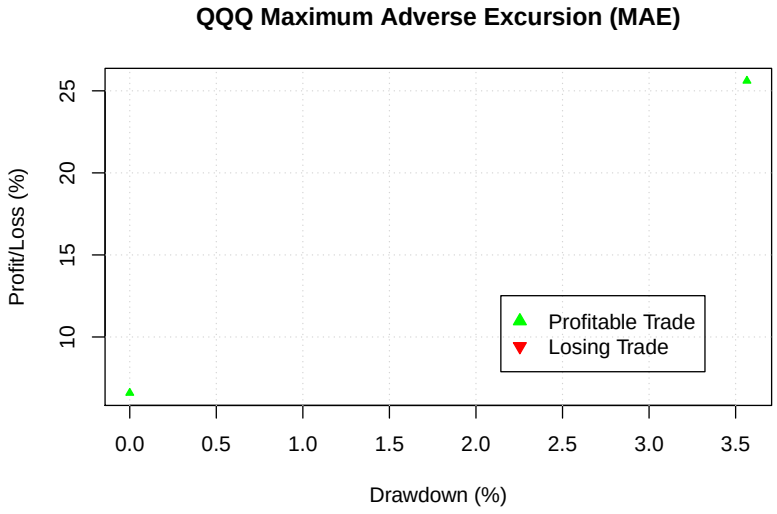
MACD



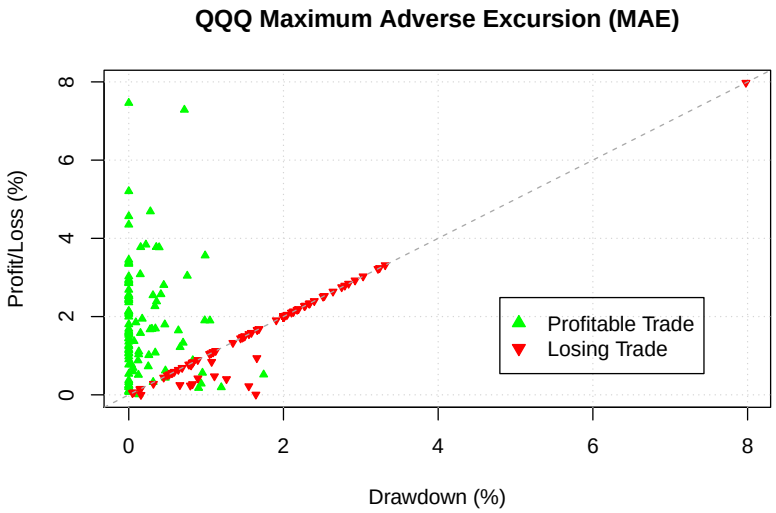
5.6.2 Maximum Adverse Excursion

The following trading charts show the drawdown (%) versus Profit/Loss(%) for the best non-optimized backtests. The results in the charts validate the selected range of stop-loss and trail-stop-loss ranges defined for optimization in the following section.

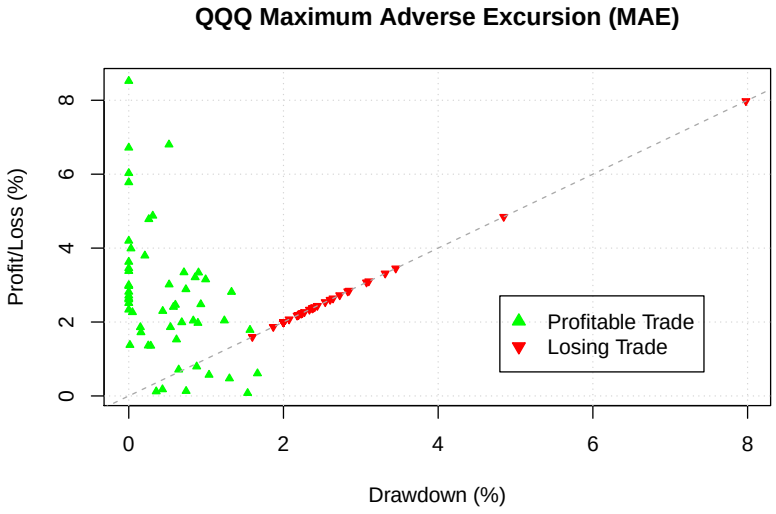
ADX



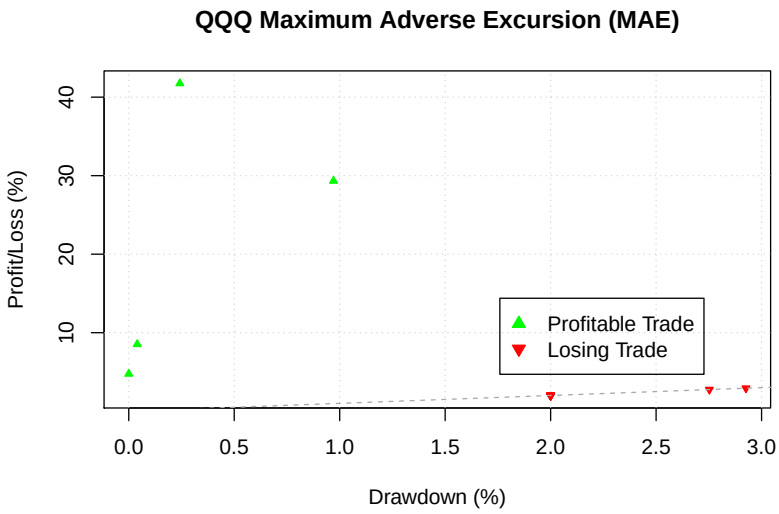
DVO


QQQ Maximum Adverse Excursion (MAE)
Profit/Loss (%)
Drawdown (%)
Profitable Trade
Losing Trade
0
2
4
6
8

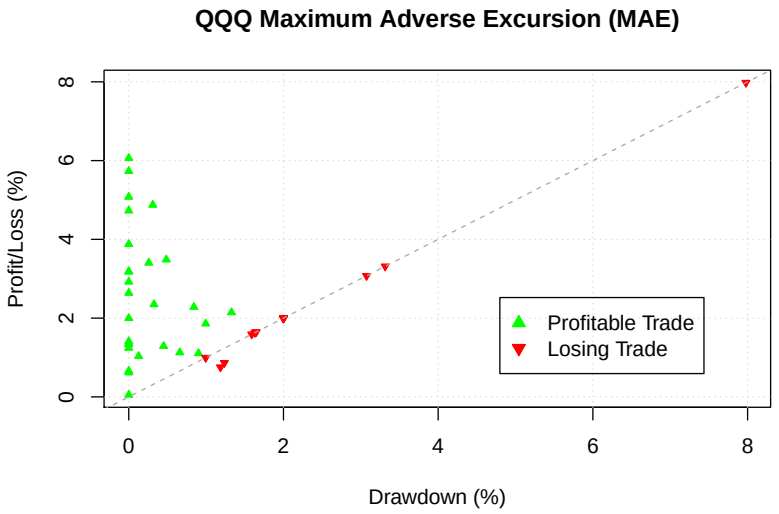
RSI


QQQ Maximum Adverse Excursion (MAE)
Profit/Loss (%)
Drawdown (%)
Profitable Trade
Losing Trade
0
2
4
6
8

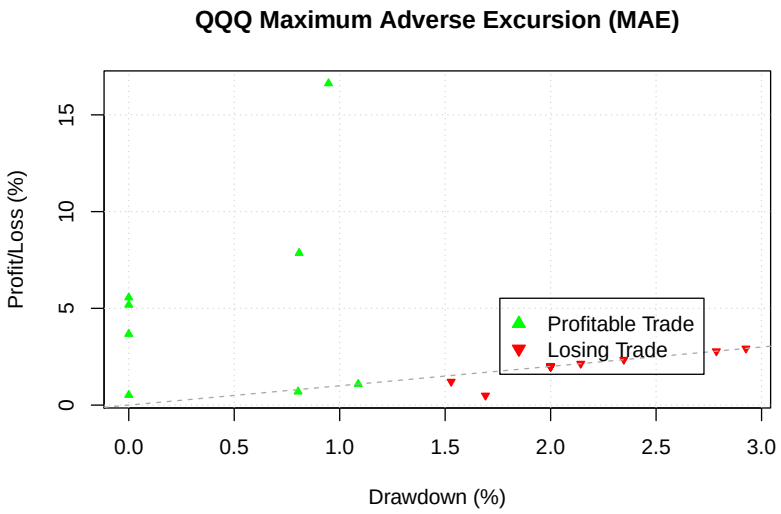
SMA


QQQ Maximum Adverse Excursion (MAE)
Profit/Loss (%)
10
20
30
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Drawdown (%)
Profitable Trade
Losing Trade

BBAND


QQQ Maximum Adverse Excursion (MAE)
Profit/Loss (%)
0
2
4
6
8
0
2
4
6
8
Drawdown (%)
Profitable Trade
Losing Trade

MACD


QQQ Maximum Adverse Excursion (MAE)
Profit/Loss (%)
0
5
10
15
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Drawdown (%)
Profitable Trade
Losing Trade

6 Monte Carlo Analysis

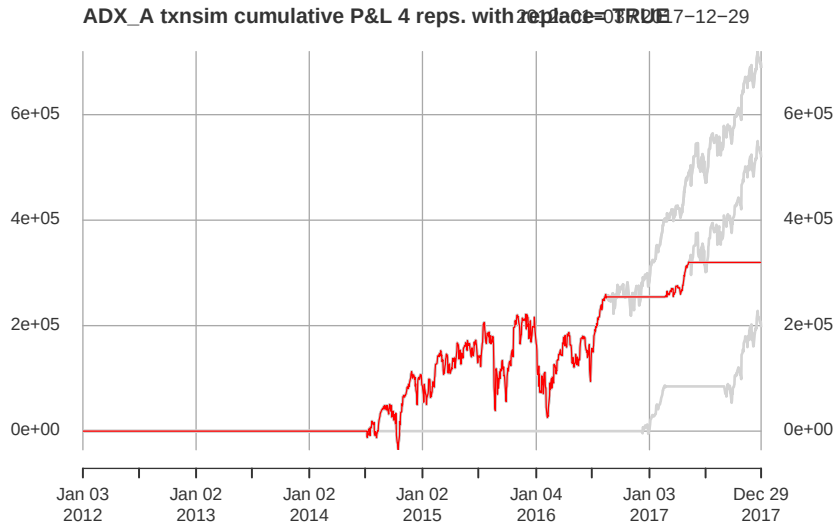
Two Monte Carlo methods were discussed in CFRM 522. This project will focus on **txnsim**. The main reason is that it was presented as most useful for analyzing skill versus luck of a strategy, due to all sampling being from results from random strategies, creating synthetic data that tries to create different versions of reality with the actual trading data.

Monte Carlo analysis is utilized in this project for two purposes:

- Check the performance of the best non-optimized backtest in-sample strategies is at the upper end (best performing) of the Monte Carlo simulations
- Check the performance metrics from the Monte Carlo simulations are comparable to the non-optimized backtests.

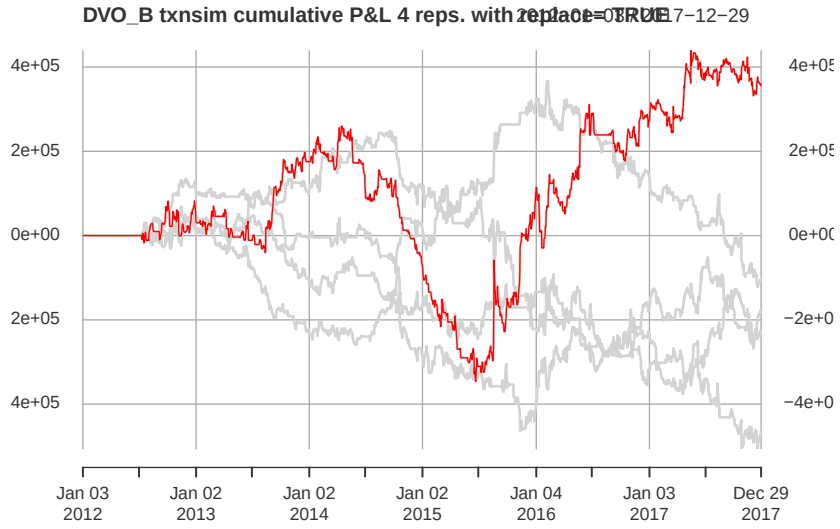
The following set of charts and results are in agreement with the non-optimized results, and all but the MACD strategy appear to show skill. The MACD strategy it is still move forward in the analysis to observe its behavior under out-of-sample validation with walk-forward-analysis.

6.1 ADX



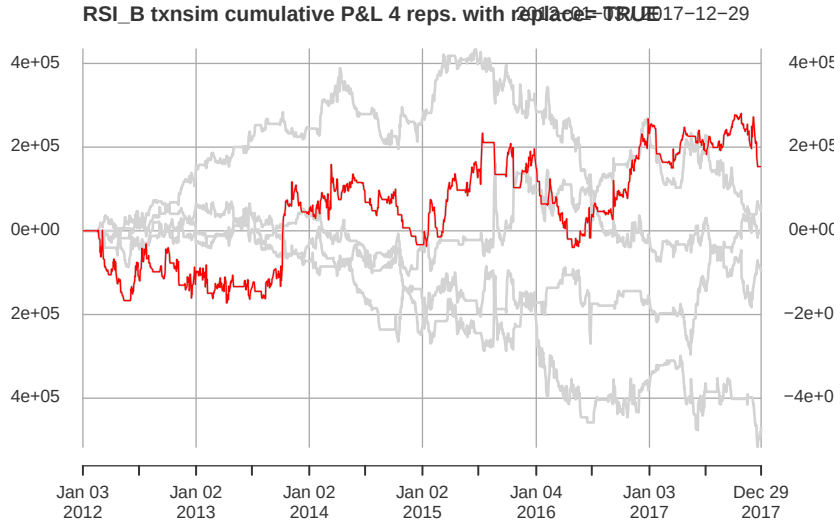
	backtest	Std. Error	Lower CI	Upper CI
mean	2.119563e+02	1.460313e+02	3.356749e+01	6.059998e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	7.129402e+03	2.544807e+03	1.953335e+03	1.192880e+04
maxDD	-1.957015e+05	6.881949e+04	-2.998082e+05	-3.004076e+04
sharpe	2.972990e-02	1.038550e-02	2.601710e-02	6.672730e-02
totalPL	3.198421e+05	2.203613e+05	5.065334e+04	9.144537e+05

6.2 DVO



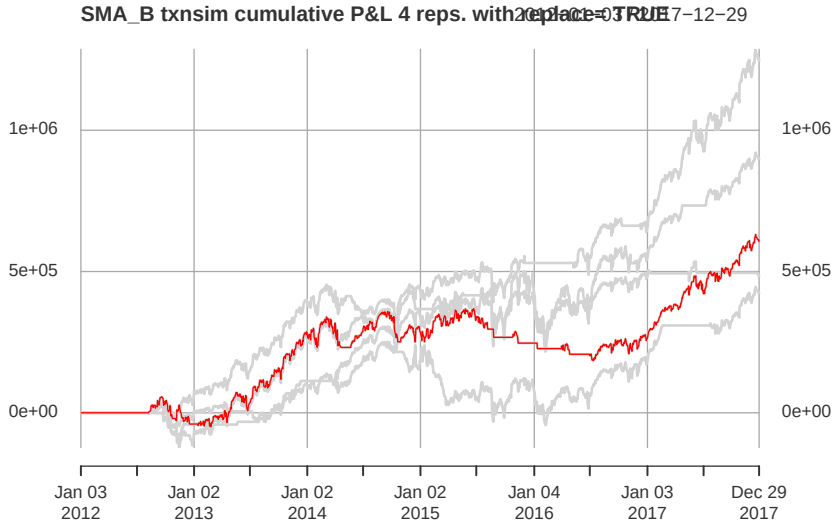
	backtest	Std. Error	Lower CI	Upper CI
mean	2.346249e+02	1.986917e+02	-4.718912e+02	3.069659e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	1.121214e+04	1.489350e+03	5.670936e+03	1.150908e+04
maxDD	-6.062083e+05	1.168269e+05	-7.562285e+05	-2.982755e+05
sharpe	2.092600e-02	2.181830e-02	-5.496990e-02	3.055620e-02
totalPL	3.540490e+05	2.998257e+05	-7.120838e+05	4.632115e+05

6.3 RSI



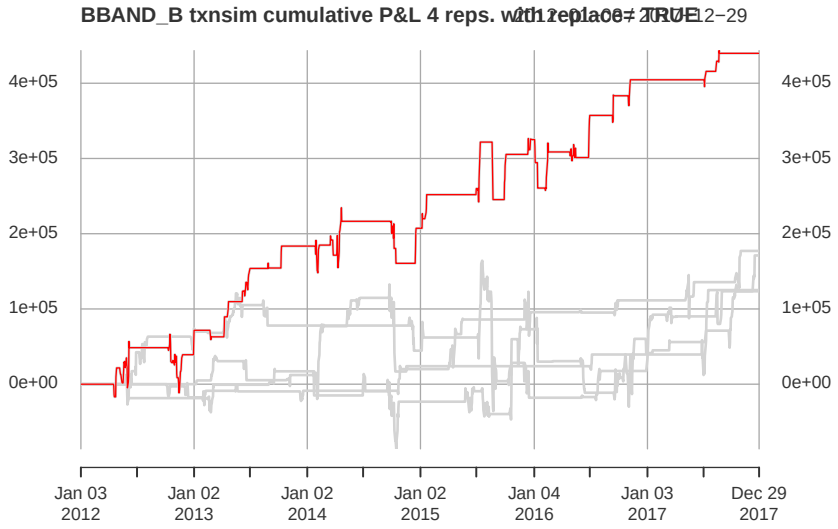
	backtest	Std. Error	Lower CI	Upper CI
mean	1.017496e+02	1.598199e+02	-3.593738e+02	2.671086e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	1.123685e+04	1.190790e+03	7.057811e+03	1.172562e+04
maxDD	-2.732253e+05	1.201686e+05	-6.290184e+05	-1.579662e+05
sharpe	9.055000e-03	1.617840e-02	-3.675690e-02	2.666140e-02
totalPL	1.535402e+05	2.411682e+05	-5.422951e+05	4.030669e+05

6.4 SMA



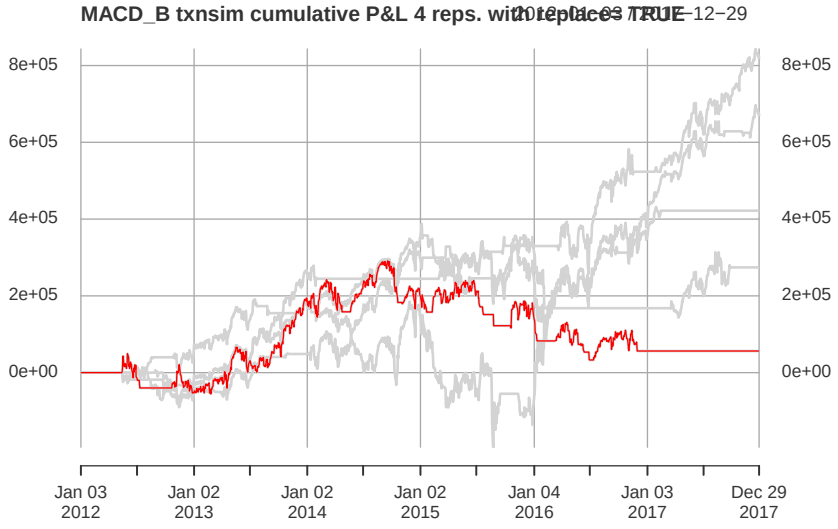
	backtest	Std. Error	Lower CI	Upper CI
mean	4.006497e+02	2.254365e+02	4.068791e+01	9.243827e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	7.653862e+03	1.647264e+03	5.587618e+03	1.204478e+04
maxDD	-1.839959e+05	7.089284e+04	-3.527184e+05	-7.482363e+04
sharpe	5.234610e-02	1.982500e-02	1.479270e-02	9.250520e-02
totalPL	6.045804e+05	3.401837e+05	6.139805e+04	1.394894e+06

6.5 BBAND



	backtest	Std. Error	Lower CI	Upper CI
mean	2.914142e+02	8.768818e+01	-3.448316e+01	3.092482e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	5.083473e+03	1.055796e+03	2.197079e+03	6.335723e+03
maxDD	-7.776272e+04	5.454303e+04	-2.229448e+05	-9.140055e+03
sharpe	5.732580e-02	1.570980e-02	9.271000e-04	6.250840e-02
totalPL	4.397440e+05	1.323215e+05	-5.203509e+04	4.666555e+05

6.6 MACD



	backtest	Std. Error	Lower CI	Upper CI
mean	3.736706e+01	2.011869e+02	-9.780464e+01	6.908334e+02
median	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
stddev	7.022856e+03	1.638261e+03	5.059778e+03	1.148164e+04
maxDD	-2.582970e+05	8.818840e+04	-4.206815e+05	-7.498931e+04
sharpe	5.320800e-03	2.243890e-02	-9.391900e-03	7.856700e-02
totalPL	5.638689e+04	3.035910e+05	-1.475872e+05	1.042468e+06

7 In-Sample Optimization

7.1 Overview

A set of optimizations where conducted within the sample time period of 2012-2017. The optimizations varied a set of hyper parameters for each trade strategy as defined in the *Trading Strategies* section. The purpose of this optimization is to identify the performance of optimum trade strategies with or without using stop and trail-stop loss rules. In addition, simulations were conducted for each strategy that optimized the strategies hyper parameters as well as optimized the stop loss and trail loss hyper parameters.

The goal of the optimization is to identify the optimum configuration within the in-sample period. This allows comparison to the best non-optimized performance of the trade strategies to evaluate the optimization effect. Furthermore, to define the hyper parameters for each strategy to configure for Walk-Forward-Analysis.

Optimizing Metric: Max DrawDown

All Optimizing simulations are executed with the runStrategies() function.

Additional Inputs:

- addDistStopLoss
  - Option to add stop loss optimization when doing an optimization in quantstrat. This is an additional distribution added to the hyper parameter distributions of each strategy
  - Default = FALSE
- addDistTrailLoss
  - Option to add trail stop loss optimization when doing an optimization in quantstrat. This is an additional distribution added to the hyper parameter distributions of each strategy
  - Default = FALSE

7.2 No Stop Loss and No Trailing Loss

```
suppressWarnings( rm( opt_sum_results) )
opt_sum_results   = runStrategies (strategies            = supported_strategies,
                                   strategies_args       = strategies_def_arguments,
                                   append_portf_name     = "A",
                                   run_type              = "paramset",
                                   initEq_var            = initEq,
                                   stock.st_var          = stock.st,
                                   initDate_var          = initDate,
                                   addStopLoss           = FALSE,
                                   addTrailStop          = FALSE)
```

7.3 Stop Loss set - No-Optimized

```
suppressWarnings( rm( opt_sum_results_stop_loss_set) )
opt_sum_results_stop_loss_set = runStrategies (strategies  = supported_strategies,
                                     strategies_args       = strategies_def_arguments,
                                     append_portf_name     = "B",
                                     run_type              = "paramset",
```

```
                                    initEq_var            = initEq,
                                    stock.st_var          = stock.st,
                                    initDate_var          = initDate,
                                    addStopLoss           = TRUE,
                                    addTrailStop          = FALSE)
```

7.4 Stop Loss set and Optimized

```
suppressWarnings( rm( opt_sum_results_stop_loss) )
opt_sum_results_stop_loss  = runStrategies (strategies   = supported_strategies,
                                    strategies_args       = strategies_def_arguments,
                                    append_portf_name     = "C",
                                    run_type              = "paramset",
                                    initEq_var            = initEq,
                                    stock.st_var          = stock.st,
                                    initDate_var          = initDate,
                                    addStopLoss           = TRUE,
                                    addTrailStop          = FALSE,
                                    addDistStopLoss       = TRUE,
                                    addDistTrailLoss      = FALSE )
```

7.5 Trail Stop Loss set and Optimized

```
suppressWarnings( rm( opt_sum_results_trail_stop_loss) )
opt_sum_results_trail_stop_loss  = runStrategies (strategies      = supported_strategies,
                                        strategies_args       = strategies_def_arguments,
                                        append_portf_name     = "D",
                                        run_type              = "paramset",
                                        initEq_var            = initEq,
                                        stock.st_var          = stock.st,
                                        initDate_var          = initDate,
                                        addStopLoss           = FALSE,
                                        addTrailStop          = TRUE,
                                        addDistStopLoss       = FALSE,
                                        addDistTrailLoss      = TRUE )
```

7.6 Summary Results

- The following summary tables provide the results of all optimizations with the key trading statistics. This data is then review to draw conclusions.
- As with the non-optimizing backtests, the number of plots is minimized due to the large number of permutations executed, only a few are selected to confirm visually that the optimization.

Optimized Trade Strategies Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_A.train.1	QQQ	4	2	313867.9	18.4849446	-195701.49	1.6038093
DVO_A.train.1	QQQ	461	175	440528.4	0.7289437	-740877.02	0.5946040
RSI_A.train.1	QQQ	541	199	648183.1	1.4065113	-440271.66	1.4722344
SMA_A.train.7	QQQ	39	19	427196.8	-0.3968163	-424058.29	1.0074010
BBAND_A.train.4	QQQ	85	41	599793.9	12.4293401	-52345.07	11.4584593

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
MACD_A.train.1	QQQ	63	31	137553.0	0.3853143	-497668.50	0.2763947

Optimized Trade Strategies with Stop-Loss Defaulted - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_B.train.10	QQQ	11	5	-66951.91	-14.3501548	-122336.6	-0.5472761
DVO_B.train.9	QQQ	314	132	303855.48	0.9205259	-394202.2	0.7708112
RSI_B.train.4	QQQ	258	105	153540.20	0.5768753	-273225.3	0.5619546
SMA_B.train.5	QQQ	29	14	604580.37	2.4532472	-183995.9	3.2858352
BBAND_B.train.4	QQQ	85	42	334907.79	5.2000766	-76447.3	4.3808978
MACD_B.train.2	QQQ	64	32	66727.45	0.8251617	-168730.1	0.3954685

Optimized Trade Strategies with Stop-Loss optimized - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_C.train.26	QQQ	10	5	23252.75	2.1740263	-42122.09	0.5520322
DVO_C.train.41	QQQ	328	147	290078.18	1.2137054	-173672.82	1.6702566
RSI_C.train.11	QQQ	347	159	379717.68	1.5235312	-210971.54	1.7998526
SMA_C.train.26	QQQ	25	12	379772.67	-1.8110228	-118940.21	3.1929712
BBAND_C.train.17	QQQ	85	42	334907.79	5.2000766	-76447.30	4.3808978
MACD_C.train.1	QQQ	64	32	13949.29	0.2612709	-147099.47	0.0948290

Optimized Trade Strategies with Trail-Stop-Loss optimized - Performance Metrics

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_D.train.19	QQQ	12	6	-34089.49	-15.6655440	-34089.49	-1.0000000
DVO_D.train.7	QQQ	351	171	173915.39	0.9853831	-195833.28	0.8880788
RSI_D.train.7	QQQ	385	188	129164.77	0.6157706	-208518.96	0.6194390
SMA_D.train.7	QQQ	32	16	33723.53	1.6381234	-41282.88	0.8168890
BBAND_D.train.16	QQQ	20	10	41548.78	3.0145505	-56588.52	0.7342264
MACD_D.train.16	QQQ	52	26	22877.31	0.6704077	-63474.35	0.3604183

The following behavior can be observed from the summary metrics above:

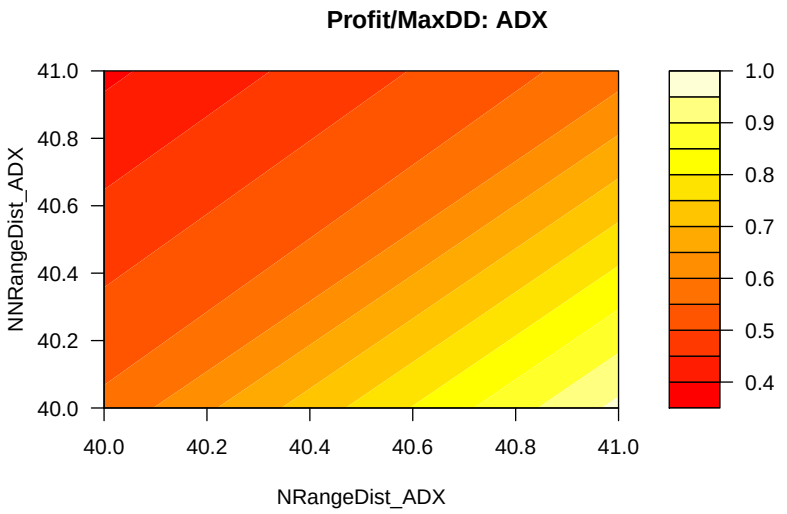
- ADX: Best option is regular hyper parameter optimization without any kind of loss optimization.
- DVO, RSI, SMA, BBand, MACD: Best optimzaton option is hyper parameter optimization with the default configured stop loss.

7.7 Optimum Charts In-Sample

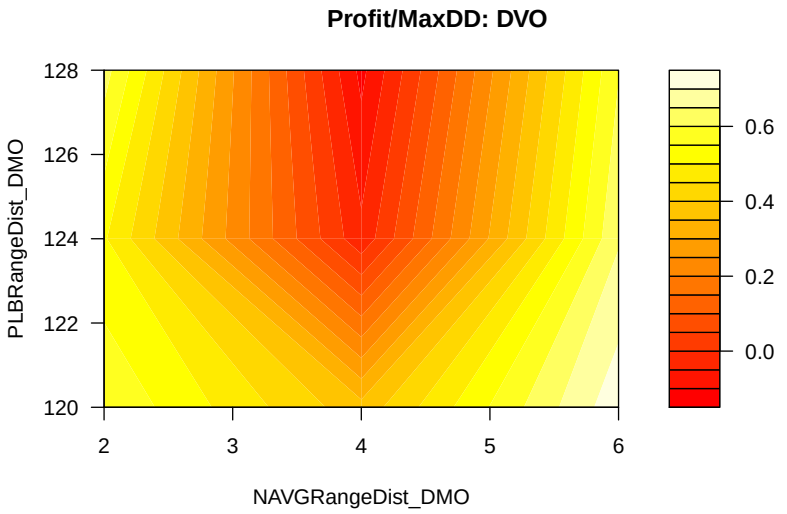
7.7.1 HeatMaps

The following heatmaps proved the optimization was actually conducted, only a small sample is selected for visual validation.

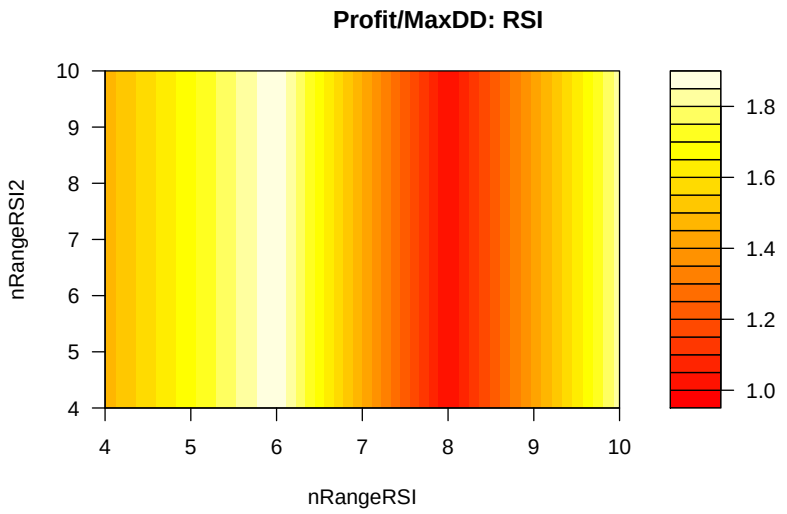
ADX


Profit/MaxDD: ADX
NNRangeDist_ADX
41.0
40.8
40.6
40.4
40.2
40.0
40.0
40.2
40.4
40.6
40.8
41.0
NRangeDist_ADX
1.0
0.9
0.8
0.7
0.6
0.5
0.4

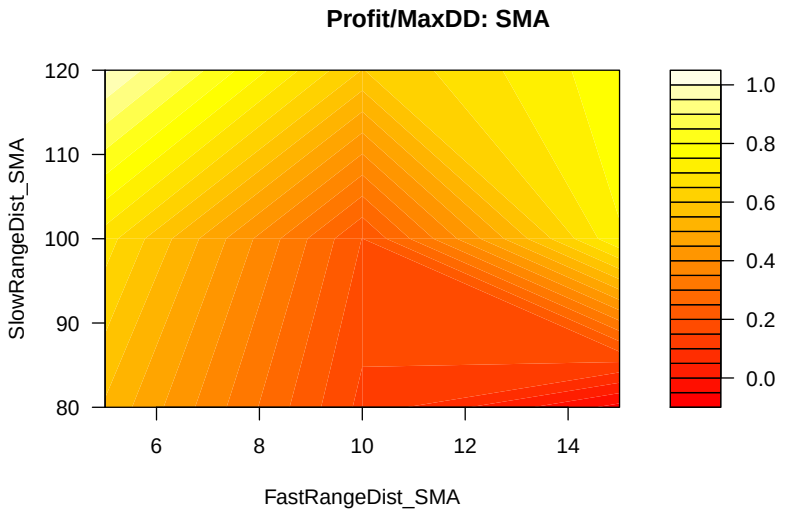
DVO


Profit/MaxDD: DVO
PLBRangeDist_DMO
128
126
124
122
120
2
3
4
5
6
NAVGRangeDist_DMO
0.6
0.4
0.2
0.0

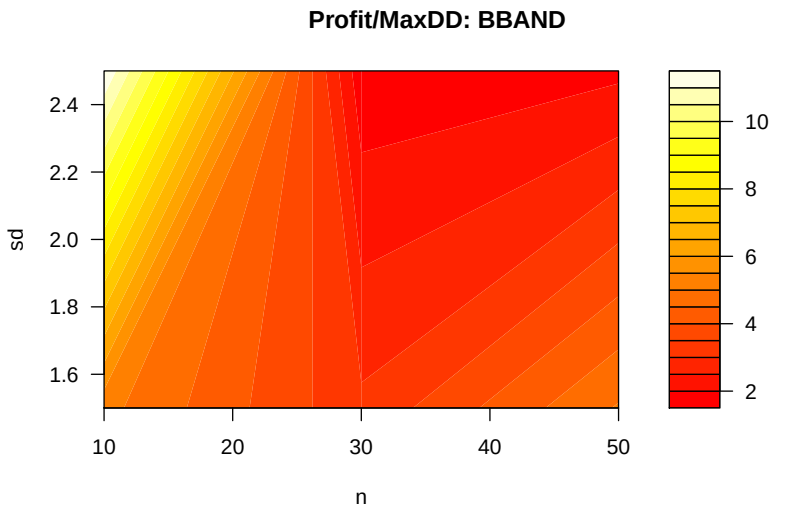
RSI


Profit/MaxDD: RSI
nRangeRSI2
nRangeRSI
10
9
8
7
6
5
4
4
5
6
7
8
9
10
1.8
1.6
1.4
1.2
1.0

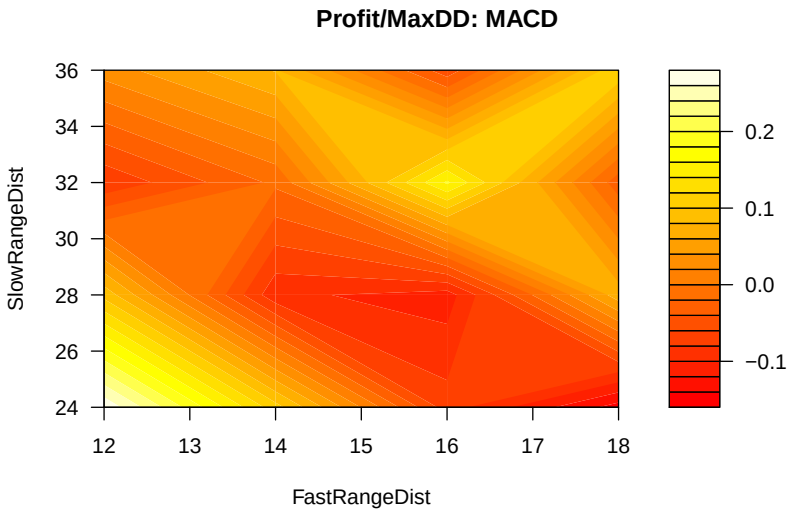
SMA


Profit/MaxDD: SMA
SlowRangeDist_SMA
FastRangeDist_SMA
120
110
100
90
80
6
8
10
12
14
1.0
0.8
0.6
0.4
0.2
0.0

BBAND


Profit/MaxDD: BBAND
sd
2.4
2.2
2.0
1.8
1.6
10
20
30
40
50
n
10
8
6
4
2

MACD


Profit/MaxDD: MACD
SlowRangeDist
36
34
32
30
28
26
24
12
13
14
15
16
17
18
FastRangeDist
0.2
0.1
0.0
−0.1

8 In-Sample Analysis

8.1 Summary

This section compares the results from the in-sample backtests against the best optimized in-sample backtests for each trading strategy. Analyzing the following two tables of key performance metrics for both experiments, the following conclusions can be drawn:

- Optimization with a default stop loss yields a significant reduction in Max-Drawdown compare to the non-optimize set for some strategies, except for ADX.
- There is a reduction in Net.Trading.PL for some strategies due to the optimization, which used Max.Drawdown as the optimizing metric.
- The annualized sharpe-ration slightly improves or stays the same for all strategies
- Given the above findings, walk-forward analysis in the next section will be configured with the defaulted stop-loss configuration, for all strategies, except ADX. In addition, the optimization metric selected for WFA is Net.Trading.PL. The goal will be to try to increased the profitability of the strategies, and used the stop loss configuration to minimize drawdowns, as indicated by the results of the optimization.

8.2 Non-optimize Strategies Summary

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_A	QQQ	4	2	319842.10	18.9865833	-195701.49	1.6343366
DVO_B	QQQ	491	214	354049.01	0.8273919	-606208.26	0.5840386
RSI_B	QQQ	258	105	153540.20	0.5768753	-273225.26	0.5619546
SMA_B	QQQ	29	14	604580.37	2.4532472	-183995.95	3.2858352
BBAND_B	QQQ	92	43	439743.95	5.2457362	-77762.72	5.6549455
MACD_B	QQQ	52	26	56386.89	0.7727170	-258296.95	0.2183026

8.3 Optimized strategies Summary

Portfolio	Symbol	Num.Txns	Num.Trades	Net.Trading.PL	Ann.Sharpe	Max.Drawdown	Profit.To.Max.Draw
ADX_B.train.10	QQQ	11	5	-66951.91	-14.3501548	-122336.6	-0.5472761
DVO_B.train.9	QQQ	314	132	303855.48	0.9205259	-394202.2	0.7708112
RSI_B.train.4	QQQ	258	105	153540.20	0.5768753	-273225.3	0.5619546
SMA_B.train.5	QQQ	29	14	604580.37	2.4532472	-183995.9	3.2858352
BBAND_B.train.4	QQQ	85	42	334907.79	5.2000766	-76447.3	4.3808978
MACD_B.train.2	QQQ	64	32	66727.45	0.8251617	-168730.1	0.3954685

8.4 Metrics for Optimum Trade Strategies

8.4.1 Trade Related

strategy	Win.Percent	Loss.Percent	WL.Ratio
ADX	100.00000	0.00000	Inf
DVO	50.75758	49.24242	1.0307692
RSI	51.42857	48.57143	1.0588235
SMA	21.42857	78.57143	0.2727273
BBAND	73.80952	26.19048	2.8181818
MACD	34.37500	65.62500	0.5238095

8.4.2 Profit Related

strategy	Gross.Profits	Gross.Losses	Profit.Factor
ADX	313867.9	0.0	NA
DVO	2177199.5	-1878720.4	1.158874
RSI	1852839.1	-1699298.9	1.090355
SMA	424719.4	-237629.9	1.787314
BBAND	582162.2	-247254.4	2.354507
MACD	470838.0	-404110.6	1.165122

8.4.3 Averages

strategy	Avg.Win.Trade	Avg.Losing.Trade	Avg.WinLoss.Ratio
ADX	156933.93	NaN	NA
DVO	32495.52	-28903.39	1.1242804
RSI	34311.83	-33319.59	1.0297797
SMA	141573.12	-21602.72	6.5534861
BBAND	18779.43	-22477.68	0.8354701
MACD	42803.45	-19243.36	2.2243234

9 Walk-Forward-Analysis

9.1 Overview

Walk-forward analysis is a powerful approach to test trading strategies using an in-sample and out-sample approach. The concept is very common in testing Machine Learning and Artificial Intelligence models. The basic idea is to divide your data into multiple training and testing periods. In each training period the model is optimized (in-sample), and then based on the hyper parameters optimized for that training cycle, the corresponding testing period is evaluated. The latter process is repeated many times, this process is known as Cross-Validation. Statistics are derived from the set of cross-validation periods. In Finance, unlike other domains, it is important that the data is not shuffle, due to the nature of the data, for instance, we cannot have Jan-3 2013 market data, following a 2015 data point, the dynamics and the market regimes are different, no to mention the calculation of daily returns will not make sense.

Walk-forward-analysis is the author's opinion the real true test for a trading strategy, as it evaluates models out-of-sample, data that the model should never have seen during optimization. In addition, in order to make comprehensive evaluation process; the data should incorporate a large variety of market regimes, in particularly periods of large volatility and down times, such as the financial crisis from 2007-2008. Looking at model behaviors during extreme market regimes will provide some insight into how trading models may perform under stress.

The periods selected for analysis in this research document are 2012-2017 and 2005-2017. The first period is five years and should provide an indication of how the model is performing in the near past, this is a subjective decision, but selected mainly as there are about 1250 trading days, and results in three cross-validation periods or more depending on the length of the training and testing windows. The second period selected, includes the financial crisis, which introduces (apart from COVID) some of the largest volatility and down market periods in the past twenty years.

The following is the detail process follow for walk-forward-analysis:

- Analysis per time period
  - Perform WFA for one training and testing window for each trading strategy
  - Perform WFA for multiple training and testing windows for each trading strategy
  - Saved all the statistics derived in each WFA cross-validation period (each training and test permutation)
- Defined a methodology to select optimum training and test windows
- Select optimum training and test windows per strategy
- Compare results between the two time periods

The walk-forward-analysis was executed with the optimum stop loss was determined by the in-sample optimization. In addition, due to the large number or permutations executed, the walk forward charts were not generated, these charts even so pretty to look do not aid on analysis across hundreds of simulations, that is six-strategies times two time periods times twenty training and test periods for 2005-2017, and then additional ones for 2012-2017. In order to analyzed such a large set of simulations all statistical trading metrics of each WFA execution are retrieved and stored in tibbles() during each run. Tibbles provide SQL-like capabilities to analyze the results, and this capability was employed to mine the results and extract the optimum performance windows per strategy using the methodology defined.

9.2 Analysis Time Period : 2012-2017

9.2.1 Single WFA Window

```
suppressWarnings( rm( wfa_sum_all) )
wfa_sum_all = runStrategies (strategies            = supported_strategies,
                             strategies_args       = strategies_def_arguments,
                             run_type              = "wfa",
                             initEq_var            = initEq,
                             stock.st_var          = stock.st,
                             initDate_var          = initDate,
                             periodTrainLength_var = periodTrainLength,
                             periodTestLength_var  = periodTestLength,
                             period_wfa_var        =  period_wfa,
                             opt_metric_var        = opt_metric,
                             cleanup_var           = TRUE,
                             output_fil_var        = "all_strategies_one_wfa_run_5year_.csv",
                             addStopLoss           = TRUE,
                             addTrailStop          = FALSE)
```

strategy	period_train_len	period_test_len	period	wfe_Net.Trading.PL	ave_train_Net.Trading.PL	ave_test_Net.Trading.PL
ADX	36	12	months	-3.0156215	-5525.312	16662.25
DVO	36	12	months	2.0535770	124772.254	256229.43
RSI	36	12	months	0.5390354	43984.938	23709.44
SMA	36	12	months	2.2745786	47439.425	107904.70
BBAND	36	12	months	1.9820887	46039.087	91253.56
MACD	36	12	months	-1.6190202	22724.340	-36791.17

strategy	period_train_len	period_test_len	period	wfe_Max.Drawdown	ave_train_Max.Drawdown	ave_test_Max.Drawdown
ADX	36	12	months	3.291964	-11358.63	-37392.20
DVO	36	12	months	1.462992	-43812.89	-64097.89
RSI	36	12	months	2.646366	-43286.91	-114552.99
SMA	36	12	months	2.041092	-23720.37	-48415.47
BBAND	36	12	months	2.589555	-12168.15	-31510.10
MACD	36	12	months	3.441515	-29498.75	-101520.38

strategy	period_train_len	period_test_len	period	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
ADX	36	12	months	7.616248	-1.5950264	-12.1481160
DVO	36	12	months	4.291666	0.2363668	1.0144075
RSI	36	12	months	3.404871	0.0353069	0.1202156
SMA	36	12	months	610.575493	-0.1285342	-78.4798598
BBAND	36	12	months	11.643872	0.1400829	1.6311071
MACD	36	12	months	2.416753	-0.5385831	-1.3016222

9.2.2 Multiple WFA Windows

```
suppressWarnings( rm( multiple_wfa_sum_df) )
multiple_wfa_sum_df = runStrategiesMultipleWFA (
                            strategies            = c("DVO", "RSI", "SMA", "BBAND", "MACD"),
                            strategies_args       = strategies_def_arguments,
                            initEq_var            = initEq,
                            stock.st_var          = stock.st,
                            initDate_var          = initDate,
```

```
                                periodTrainLengthRange = seq(12, 36, by=12),
                                periodTestLengthRange  = seq(6, 12, by=6),
                                period_wfa_var         =  period_wfa,
                                opt_metric_var         = opt_metric,
                                output_fil_var         = "all_strategies_multwin_wfa_run_5year_.csv",
                                addStopLoss            = TRUE,
                                addTrailStop           = FALSE )
```

strategy	period_train_len	period_test_len	period	wfe_Net.Trading.PL	ave_train_Net.Trading.PL	ave_test_Net.Trading.PL
DVO	12	6	months	3.1178638	39676.289	123705.266
RSI	12	6	months	2.0584706	29932.304	61614.769
SMA	12	6	months	0.9259273	40650.076	37639.016
BBAND	12	6	months	3.5444464	19456.148	68961.274
MACD	12	6	months	-1.2925544	37247.355	-48144.233
DVO	12	12	months	0.9646773	186226.247	179648.239
RSI	12	12	months	0.7513310	91642.835	68854.107
SMA	12	12	months	1.9088604	72982.297	139313.017
BBAND	12	12	months	0.5745773	77591.271	44582.186
MACD	12	12	months	0.0408215	66329.544	2707.669
DVO	24	6	months	0.0895910	56203.974	5035.368
RSI	24	6	months	4.0790423	10247.444	41799.756
SMA	24	6	months	-1.4333243	19748.969	-28306.678
BBAND	24	6	months	3.0252288	15504.196	46903.739
MACD	24	6	months	-3.0108640	14503.426	-43667.843
DVO	24	12	months	0.5482981	183183.444	100439.132
RSI	24	12	months	0.6442390	78928.676	50848.929
SMA	24	12	months	2.3540639	39497.939	92980.672
BBAND	24	12	months	0.5225945	57012.040	29794.178
MACD	24	12	months	-1.4862303	27229.061	-40468.655
DVO	36	6	months	3.6409552	44824.807	163205.111
RSI	36	6	months	3.0052672	9906.367	29771.281
SMA	36	6	months	-2.2678366	21348.747	-48415.471
BBAND	36	6	months	6.5623202	13905.685	91253.556
MACD	36	6	months	-8.7797584	9832.667	-86328.444
DVO	36	12	months	2.0535770	124772.254	256229.433
RSI	36	12	months	0.5390354	43984.938	23709.439
SMA	36	12	months	2.2745786	47439.425	107904.703
BBAND	36	12	months	1.9820887	46039.087	91253.556
MACD	36	12	months	-1.6190202	22724.340	-36791.167

strategy	period_train_len	period_test_len	period	wfe_Max.Drawdown	ave_train_Max.Drawdown	ave_test_Max.Drawdown
DVO	12	6	months	12.764780	-6648.711	-84869.33
RSI	12	6	months	6.968250	-9863.566	-68731.79
SMA	12	6	months	11.870649	-5916.102	-70227.97
BBAND	12	6	months	5.007844	-3878.178	-19421.31
MACD	12	6	months	10.835356	-5930.063	-64254.34
DVO	12	12	months	3.393787	-26423.912	-89677.13
RSI	12	12	months	3.009592	-22837.576	-68731.79
SMA	12	12	months	2.276908	-22601.884	-51462.41
BBAND	12	12	months	1.258611	-17500.204	-22025.94
MACD	12	12	months	2.781337	-21900.341	-60912.23
DVO	24	6	months	13.078506	-10603.580	-138678.98
RSI	24	6	months	4.738007	-18133.097	-85914.74
SMA	24	6	months	9.903384	-6495.559	-64328.01
BBAND	24	6	months	3.546838	-6844.585	-24276.64
MACD	24	6	months	8.719431	-8180.216	-71326.83
DVO	24	12	months	7.246742	-19962.113	-144660.28
RSI	24	12	months	2.369004	-36266.194	-85914.74
SMA	24	12	months	4.951692	-12991.118	-64328.01
BBAND	24	12	months	2.628692	-13689.171	-35984.62
MACD	24	12	months	3.110453	-21902.802	-68127.64
DVO	36	6	months	4.700042	-13637.727	-64097.89

strategy	period_train_len	period_test_len	period	wfe_Max.Drawdown	ave_train_Max.Drawdown	ave_test_Max.Drawdown
RSI	36	6	months	10.479710	-10930.931	-114552.99
SMA	36	6	months	4.082185	-11860.186	-48415.47
BBAND	36	6	months	5.179110	-6084.076	-31510.10
MACD	36	6	months	8.560610	-11310.905	-96828.25
DVO	36	12	months	1.462992	-43812.893	-64097.89
RSI	36	12	months	2.646366	-43286.908	-114552.99
SMA	36	12	months	2.041092	-23720.371	-48415.47
BBAND	36	12	months	2.589555	-12168.152	-31510.10
MACD	36	12	months	3.441515	-29498.746	-101520.38

strategy	period_train_len	period_test_len	period	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
DVO	12	6	months	0.4366830	0.8511918	0.3717010
RSI	12	6	months	0.5936539	0.3118151	0.1851102
SMA	12	6	months	-0.6745382	0.9144562	-0.6168356
BBAND	12	6	months	1.3901622	0.5606444	0.7793867
MACD	12	6	months	-3.5563393	0.5245461	-1.8654641
DVO	12	12	months	0.5215360	0.8759738	0.4568519
RSI	12	12	months	0.5353286	0.3885945	0.2080257
SMA	12	12	months	-0.0178945	-39.4843921	0.7065539
BBAND	12	12	months	1.3505501	0.5395364	0.7286710
MACD	12	12	months	-0.5331402	-0.0867801	0.0462660
DVO	24	6	months	0.1264254	0.2152005	0.0272068
RSI	24	6	months	11.8623608	0.0129925	0.1541212
SMA	24	6	months	0.0273463	-45.3199962	-1.2393330
BBAND	24	6	months	5.8029128	0.1313803	0.7623882
MACD	24	6	months	3.6455361	-0.3720006	-1.3561414
DVO	24	12	months	0.4940111	0.5872333	0.2900998
RSI	24	12	months	0.7174669	0.2623549	0.1882310
SMA	24	12	months	0.0114794	-90.6399924	-1.0404971
BBAND	24	12	months	1.7101244	0.3096197	0.5294882
MACD	24	12	months	1.5526356	-0.7638822	-1.1860308
DVO	36	6	months	5.0424447	0.1532070	0.7725378
RSI	36	6	months	2.4840585	0.0603264	0.1498544
SMA	36	6	months	500.0242180	-0.1569521	-78.4798598
BBAND	36	6	months	17.8381920	0.0914390	1.6311071
MACD	36	6	months	11.5314383	-0.4168346	-4.8067024
DVO	36	12	months	4.2916656	0.2363668	1.0144075
RSI	36	12	months	3.4048714	0.0353069	0.1202156
SMA	36	12	months	610.5754927	-0.1285342	-78.4798598
BBAND	36	12	months	11.6438720	0.1400829	1.6311071
MACD	36	12	months	2.4167528	-0.5385831	-1.3016222

9.3 Analysis Time Period : 2005-2017

9.3.1 Single WFA Window

```
suppressWarnings( rm( wfa_sum_all_12year) )
wfa_sum_all_12year = runStrategies (strategies      = supported_strategies,
                                    strategies_args       = strategies_def_arguments,
                                    run_type              = "wfa",
                                    initEq_var            = initEq,
                                    stock.st_var          = stock.st,
                                    initDate_var          = initDate_long_period,
                                    periodTrainLength_var = periodTrainLength,
                                    periodTestLength_var  = periodTestLength,
                                    period_wfa_var        =  period_wfa,
                                    opt_metric_var        = opt_metric,
                                    cleanup_var           = TRUE,
```

```
                                output_fil_var        = "all_strategies_one_wfa_run_12year_.csv",
                                addStopLoss           = TRUE,
                                addTrailStop          = FALSE)
```

strategy	period_train_len	period_test_len	period	wfe_Net.Trading.PL	ave_train_Net.Trading.PL	ave_test_Net.Trading.PL
ADX	36	12	months	-3.0156215	-5525.312	16662.25
DVO	36	12	months	2.0535770	124772.254	256229.43
RSI	36	12	months	0.5390354	43984.938	23709.44
SMA	36	12	months	2.2745786	47439.425	107904.70
BBAND	36	12	months	1.9820887	46039.087	91253.56
MACD	36	12	months	-1.6190202	22724.340	-36791.17

strategy	period_train_len	period_test_len	period	wfe_Max.Drawdown	ave_train_Max.Drawdown	ave_test_Max.Drawdown
ADX	36	12	months	3.291964	-11358.63	-37392.20
DVO	36	12	months	1.462992	-43812.89	-64097.89
RSI	36	12	months	2.646366	-43286.91	-114552.99
SMA	36	12	months	2.041092	-23720.37	-48415.47
BBAND	36	12	months	2.589555	-12168.15	-31510.10
MACD	36	12	months	3.441515	-29498.75	-101520.38

strategy	period_train_len	period_test_len	period	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
ADX	36	12	months	7.616248	-1.5950264	-12.1481160
DVO	36	12	months	4.291666	0.2363668	1.0144075
RSI	36	12	months	3.404871	0.0353069	0.1202156
SMA	36	12	months	610.575493	-0.1285342	-78.4798598
BBAND	36	12	months	11.643872	0.1400829	1.6311071
MACD	36	12	months	2.416753	-0.5385831	-1.3016222

9.3.2 Multiple WFA Windows

```
suppressWarnings( rm( multiple_wfa_sum_df_12year) )
multiple_wfa_sum_df_12year = runStrategiesMultipleWFA (
                                strategies             = c("DVO", "RSI", "SMA", "BBAND", "MACD"),
                                strategies_args        = strategies_def_arguments,
                                initEq_var             = initEq,
                                stock.st_var           = stock.st,
                                initDate_var           = initDate_long_period,
                                periodTrainLengthRange = seq(12, 36, by=12),
                                periodTestLengthRange  = seq(6, 12, by=6),
                                period_wfa_var         =  period_wfa,
                                opt_metric_var         = opt_metric,
                                output_fil_var        = "all_strategies_multwin_wfa_run_12year_.csv")
```

strategy	period_train_len	period_test_len	period	wfe_Net.Trading.PL	ave_train_Net.Trading.PL	ave_test_Net.Trading.PL
DVO	12	6	months	-2.9278213	21100.0616	-61777.209
RSI	12	6	months	9.3024539	17670.9629	164383.318
SMA	12	6	months	2.2356000	16937.5317	37865.545
BBAND	12	6	months	5.7752667	15313.8532	88441.586
MACD	12	6	months	-1.9717786	16218.9439	-31980.167
DVO	12	12	months	-0.3303535	81342.7764	-26871.869
RSI	12	12	months	4.1290969	31579.2311	130393.707
SMA	12	12	months	0.2286337	29588.0355	6764.821
BBAND	12	12	months	2.2819290	36441.7880	83157.572

strategy	period_train_len	period_test_len	period	wfe_Net.Trading.PL	ave_train_Net.Trading.PL	ave_test_Net.Trading.PL
MACD	12	12	months	-2.3087964	21379.1057	-49360.003
DVO	24	6	months	4.1899960	20237.5457	84795.235
RSI	24	6	months	9.4942418	10681.5918	101413.615
SMA	24	6	months	11.8327024	2722.4462	32213.895
BBAND	24	6	months	4.6007898	12490.2749	57465.129
MACD	24	6	months	73.6617148	-804.0805	-59229.946
DVO	24	12	months	2.6483216	65097.1631	172398.225
RSI	24	12	months	4.0260356	26898.7203	108295.205
SMA	24	12	months	3.4357161	5444.8923	18707.104
BBAND	24	12	months	2.2900254	29086.9125	66609.768
MACD	24	12	months	16.3145803	-2038.3078	-33254.137
DVO	36	6	months	-5.1659748	16453.4907	-84998.318
RSI	36	6	months	3.6524115	14652.1566	53515.706
SMA	36	6	months	12.1954657	2785.0568	33965.065
BBAND	36	6	months	2.6838772	10475.8991	28116.027
MACD	36	6	months	29.4564987	-1533.5298	-45172.418
DVO	36	12	months	-1.8368806	28543.3893	-52430.797
RSI	36	12	months	3.8380118	21346.7765	81929.180
SMA	36	12	months	0.8105314	8520.1415	6905.842
BBAND	36	12	months	0.8486781	25638.1079	21758.502
MACD	36	12	months	58.7237222	-887.0985	-52093.724

strategy	period_train_len	period_test_len	period	wfe_Max.Drawdown	ave_train_Max.Drawdown	ave_test_Max.Drawdown
DVO	12	6	months	40.926635	-5217.244	-213524.25
RSI	12	6	months	34.790908	-5668.126	-197199.24
SMA	12	6	months	22.911846	-2465.043	-56478.68
BBAND	12	6	months	47.882698	-2355.401	-112782.95
MACD	12	6	months	42.133106	-2465.043	-103859.90
DVO	12	12	months	11.151234	-16354.792	-182376.12
RSI	12	12	months	7.701761	-25604.435	-197199.24
SMA	12	12	months	4.053367	-17857.398	-72382.59
BBAND	12	12	months	11.982688	-9468.839	-113462.14
MACD	12	12	months	6.519628	-16709.411	-108939.15
DVO	24	6	months	32.697904	-4745.915	-155181.48
RSI	24	6	months	29.612206	-9058.599	-268245.11
SMA	24	6	months	5.554438	-9178.838	-50983.29
BBAND	24	6	months	37.633913	-3269.284	-123035.94
MACD	24	6	months	13.872388	-9205.488	-127702.10
DVO	24	12	months	3.890190	-18895.653	-73507.68
RSI	24	12	months	8.627013	-20012.348	-172646.80
SMA	24	12	months	3.065330	-18357.675	-56272.33
BBAND	24	12	months	18.930275	-6538.568	-123776.88
MACD	24	12	months	5.284969	-19022.334	-100532.45
DVO	36	6	months	27.213686	-6928.406	-188547.47
RSI	36	6	months	35.451255	-8323.249	-295069.62
SMA	36	6	months	9.230829	-6731.148	-62134.07
BBAND	36	6	months	59.854860	-2397.475	-143500.52
MACD	36	6	months	12.089248	-8294.475	-100273.96
DVO	36	12	months	7.361712	-26092.345	-192084.34
RSI	36	12	months	12.940520	-14675.722	-189911.48
SMA	36	12	months	4.656066	-19156.366	-89193.29
BBAND	36	12	months	29.927430	-4794.950	-143500.52
MACD	36	12	months	6.180179	-16588.950	-102522.67

strategy	period_train_len	period_test_len	period	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
DVO	12	6	months	-0.1008873	0.3245393	-0.0327419
RSI	12	6	months	0.9317036	0.1306274	0.1217060
SMA	12	6	months	0.0249297	0.3810234	0.0094988
BBAND	12	6	months	0.1983140	0.8957405	0.1776379
MACD	12	6	months	-0.4192882	0.4143843	-0.1737464

strategy	period_train_len	period_test_len	period	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
DVO	12	12	months	-0.0333800	0.4429505	-0.0147857
RSI	12	12	months	1.9435391	0.0530804	0.1031639
SMA	12	12	months	0.0365298	-2.4150587	-0.0882216
BBAND	12	12	months	0.4701303	0.3996102	0.1878689
MACD	12	12	months	0.4269590	-0.3653135	-0.1559739
DVO	24	6	months	0.6284597	0.0758656	0.0476785
RSI	24	6	months	3.2617390	0.0165339	0.0539292
SMA	24	6	months	0.0481855	-0.3856887	-0.0185846
BBAND	24	6	months	0.7324562	0.1611503	0.1180355
MACD	24	6	months	1.0489974	-0.1787869	-0.1875470
DVO	24	12	months	0.6623418	0.1840350	0.1218941
RSI	24	12	months	0.9681612	0.0706828	0.0684324
SMA	24	12	months	0.0893579	-0.7713774	-0.0689287
BBAND	24	12	months	0.4575513	0.3354231	0.1534733
MACD	24	12	months	0.3899884	-0.3100946	-0.1209333
DVO	36	6	months	-1.2015668	0.0458289	-0.0550665
RSI	36	6	months	0.5576656	0.0680568	0.0379529
SMA	36	6	months	0.3064780	-0.1210539	-0.0371004
BBAND	36	6	months	0.5148833	0.1111129	0.0572102
MACD	36	6	months	1.4423620	-0.1197655	-0.1727452
DVO	36	12	months	-1.3734575	0.0231782	-0.0318343
RSI	36	12	months	1.4253396	0.0405127	0.0577444
SMA	36	12	months	0.9758702	-0.1316582	-0.1284813
BBAND	36	12	months	0.1750480	0.2499843	0.0437593
MACD	36	12	months	1.2329076	-0.1559895	-0.1923206

9.4 Results Summary

9.4.1 Methodology to select optimum training and test windows

Walk-forward-efficiency (WFE) metric was presented in class as a way to analyze trading strategies evaluated with WFA. It is defined as the annualized mean ratio for a given metric, that is: (annual mean of testing metric)/(annual mean of training metric). It is the author's opinion, that using this metric for selection has many issues, and can lead to selection of poorly trading strategies.

Issues with WFE:

- The metrics does not account for the sign of the underlying metric.
  - A net.trading.PL for the test period that is positive, and a training period that is negative, results in a negative WFE, which indicates that the test metric under perform the training metric.
  - Training and Test net.trading.pl in which both have negative amounts, and the test metric has a smaller value (more negative) will result in a very large positive WFE, which gives the appaerance that it is a great performing strategy from a net.trading.pl point of view (this example can be observed with real data resulting from the analysis below)
- The metric does not consider the magnitude of the underlying data.
  - Consider two trading strategies, one has a WFE of 1.0, the other a WFE of 0.9; everything else is comparable in terms of risk. By selecting the strategy with a WFE of 1, one may select a strategy that makes significant less money than the one with WFE of 0.9

– The strategy with WFE of 1.0, could be based on $10,000 (mean test net.trading.pl)/$10,000 (mean training net.trading.pl) = 1; the other strategy is $90K (mean test net.trading.pl)/$100K (mean training net.trading.pl); with everything else being similar, the most profitable strategy is the one with the lower WFE. The underlying issue is not considering the magnitude of the metrics

Sample data points illustrating some of the issues above, based on the conducted analysis:

- The SMA and MACD have misleading WFE metrics, selection based on WFE > 1 will yield a poorly selected strategy.

strategy	period_train_len	period_test_len	wfe_Ann.Sharpe	ave_train_Ann.Sharpe	ave_test_Ann.Sharpe
SMA	36	6	500.02422	-0.1569521	-78.4798598
SMA	36	12	610.57549	-0.1285342	-78.4798598
RSI	24	6	11.86236	0.0129925	0.1541212
MACD	36	6	11.53144	-0.4168346	-4.8067024
BBAND	36	6	17.83819	0.0914390	1.6311071
BBAND	36	12	11.64387	0.1400829	1.6311071

Methodology designed to select optimum training and test windows per market period tested:

1. Calculate the difference for each of the following metrics:

   net_trade_pl_diff = ave_test_Net.Trading.PL - ave_train_Net.Trading.PL

   net_max_dd_diff = ave_test_Max.Drawdown - ave_train_Max.Drawdown

   net_ann_sr_diff = ave_test_Ann.Sharpe - ave_train_Ann.Sharpe

2. Sort in descending order by: 1) strategy 2) net_trade_pl_diff
3. Narrow results for which the annualized test Trading.PL > 0 (no point on trading on strategies losing money)
4. Select the top performing strategy, the one with the highest net_trade_pl_diff
5. Further narrow to only strategies with a test annualized sharpe-ratio > user-defined-value (optional)

9.4.2 Findings for 2012-2017

strategy	period_train_len	period_test_len	net_trade_pl_diff	net_max_dd_diff	net_ann_sr_diff	ave_test_Net.Trading.PL
BBAND	36	6	77347.87	-25426.02	1.5396680	91253.56
DVO	36	12	131457.18	-20285.00	0.7780406	256229.43
SMA	12	12	66330.72	-28860.53	40.1909460	139313.02

strategy	ave_test_Max.Drawdown	ave_test_Ann.Sharpe	ave_train_Ann.Sharpe
BBAND	-31510.10	1.6311071	0.0914390

strategy	ave_test_Max.Drawdown	ave_test_Ann.Sharpe	ave_train_Ann.Sharpe
DVO	-64097.89	1.0144075	0.2363668
SMA	-51462.41	0.7065539	-39.4843921

9.4.3 Findings for 2005-2017

strategy	period_train_len	period_test_len	net_trade_pl_diff	net_max_dd_diff	net_ann_sr_diff	ave_test_Net.Trading.PL
BBAND	12	6	73127.73	-110427.55	-0.7181026	88441.59
DVO	24	12	107301.06	-54612.03	-0.0621409	172398.22
RSI	12	6	146712.36	-191531.12	-0.0089214	164383.32
SMA	36	6	31180.01	-55402.92	0.0839535	33965.07

strategy	ave_test_Max.Drawdown	ave_test_Ann.Sharpe	ave_train_Ann.Sharpe
BBAND	-112782.95	0.1776379	0.8957405
DVO	-73507.68	0.1218941	0.1840350
RSI	-197199.24	0.1217060	0.1306274
SMA	-62134.07	-0.0371004	-0.1210539

9.4.4 Result Analysis for both Time Periods

- For 2012-2017, three trading strategies identified as optimum per selection methodology defined.
  - BB, DVO, and SMA
- For 2005-2017, four trading strategies identified as optimum per criteria defined
  - BB, DVO, and SMA, RSI
  - Do note the large max. drawdowns in this period
  - Do note the small sharpe ratios
- The strategies selected for 2012-2017 have the following characteristics:
  - Ratios of Net.Trading.PL/Max-Drawdown between 2 to 4 in the test years
  - Ann. test sharpe-ratios of all but one > 1.0
  - Ann. Max-Drawdown between $30k to $60k
  - These strategies benefited from a general up market during the period without the extreme volatility observed during the financial crisis
- The strategies selected for 2005-2017 have the following characteristics:
  - Net.Trading.PLs larger than the 2012-2017, about twice as large
  - Ann. Max-Drawdowns between $62k and $197k, with RSI being the outlier on the higher end
  - Low sharpe ratios, this is probably mainly due to the extreme market down years between 2007-2008, which also can be attributed to the larger annualized Max-Drawdowns observed

- In general the strategies do appear to be profitable during extreme market periods, but subject to high volatility, specially Max-Drawdown
- It can be observed that is necessary to execute WFA across a variety of training and test window sizes, as the optimum windows varies per each strategy in both time periods

10 Conclusions

- Three of the six strategies analyzed showed potential to be evaluated with paper-trading with the NAasdaq ETF
  - SMA, DVO, and BBO
  - These three strategies have been selected based on their performance in both time periods analyzed
- The three selected strategies exhibited large losses during the financial crisis of 2007-2008
  - The behavior might indicate that these strategies do not adapt to extreme market shift and down years, additional research in other large down periods is strongly recommended (1987, 2001, and 2020)
  - This observed behavior means additional performance analysis should be conducted to understand the three, six, and twelve months moving averages of key performance metrics since 2012.
  - Boundaries for the statistical performance ranges of the key metrics should be established and closely monitored during paper trading, and production.
  - Deviation from the above established boundaries might indicate a different market regime that the strategies cannot adapt too
- The strategies hyper parameters should be optimized per the time windows defined when deploy for production
- The stop-loss rules assisted to protect in the down side and minimize Max-Drawdowns
- Explore the selected trade strategies with other market data to try to derive additional understanding of the performance.
- One aspect that needs additional exploring, it is comparison of the performance of each strategy versus the buy-hold strategy of the Nasdaq ETF normalized for risk.

11 Appendixes

11.1 Quantstrat Issues

- WFA aborts if a test period has not transactions
  - work around: avoid test periods for strategies with no transactions
- WFA may abort if using multi-core CPUs
  - R sessions are spawned per core, each takes from 800M to 1.5GB of RAM, depends on the size of the data
  - Memory appears to be leaking after repeated simulations, growing even when limiting the number of cores
  - work around: limit cores to 2 or 3, but it is highly depending on a memory on the system
- WFA does not work unless two distributions are added to a paramset
  - work around: configured an extra indicator and use the same paramset.
- WFA does not work if one of the distributions is a range of string options, such as maType in SMA
  - work around: do not use hyper params of type strings
- WFA sometimes abort when calling chart.forward(), error has to do with parsing of dates, mysterious.
  - Work around, avoid calling chart.forward(), un-predictable when the error occurs, appears to be dependent on the amount of data in the train/test period
- WFA may abort with a random error
  - work around: remove all quantstrat cache files, and try again

11.2 Source Code

```
library(quantstrat)
library(dplyr)
library(parallel)

# *** Setup Multi-Core

if( Sys.info()['sysname'] == "Windows") {
  library(doParallel)
  registerDoParallel(cores=detectCores())
} else {
  print("Mac")
  library(doMC)
  registerDoMC(cores=detectCores())
  registerDoMC(cores=3)
}

# ================================================================================
# ************************** Supporting Functions  ************************
# ================================================================================

# **** Function to determine investment, assumes initEq globally defined

osFixedDollar <- function(timestamp, orderqty, portfolio, symbol, ruletype, ...)
{
  ClosePrice <- as.numeric(Cl(mktdata[timestamp,]))
  orderqty <- round(initEq/ClosePrice,-2)
  orderside = list(...)$orderside

  cat("position sizing = ", orderqty, " ", ruletype, " ", orderside, "\n")

  if ( orderside  == "long" ) {
    return(orderqty)
  } else {
    return(-1 * orderqty)
  }
}

# Func to set signal cross/comparison

setSignalType <- function( crossOverSig ) {
  if ( crossOverSig == TRUE ) {
    return ("sigCrossover")
  } else {
    return ("sigComparison")
  }
}

addLongRules <- function(strat.st,  sigcol_enter, sigcol_exit, numShares,
                         ordertype, threshold ,  osFUN , orderset,
                         label_enter, label_exit ) {

  strat.st = add.rule(strategy = strat.st, name='ruleSignal',
                      arguments=list(sigcol    = sigcol_enter,
                                     sigval    = TRUE,
                                     orderqty  = numShares,
                                     ordertype = ordertype,
                                     orderside = 'long',
                                     threshold = threshold,
                                     osFUN     = osFUN,
                                     replace   = FALSE,
                                     orderset  = orderset ),
                      type  ='enter',
                      label = label_enter )
```

```
  strat.st = add.rule(strategy = strat.st, name='ruleSignal',
                      arguments=list(sigcol    = sigcol_exit,
                                     sigval    = TRUE,
                                     orderqty  = 'all',
                                     ordertype = ordertype,
                                     orderside = 'long',
                                     replace   = TRUE,
                                     orderset  = orderset),
                      type  ='exit',
                      label = label_exit )

  return ( strat.st )
}

addShortRules <- function(strat.st,  sigcol_enter, sigcol_exit, numShares,
                          ordertype, threshold ,  osFUN , orderset,
                          label_enter, label_exit ) {

  strat.st = add.rule(strategy = strat.st, name='ruleSignal',
                      arguments=list(sigcol    = sigcol_enter,
                                     sigval    = TRUE,
                                     orderqty  = -numShares,
                                     ordertype = ordertype,
                                     orderside = 'short',
                                     threshold = threshold,
                                     osFUN     = osFUN,
                                     replace   = FALSE,
                                     orderset  = orderset ),
                      type  ='enter',
                      label = label_enter )

  strat.st = add.rule(strategy = strat.st, name='ruleSignal',
                      arguments=list(sigcol    = sigcol_exit,
                                     sigval    = TRUE,
                                     orderqty  = 'all',
                                     ordertype = ordertype,
                                     orderside = 'short',
                                     replace   = TRUE,
                                     orderset  = orderset),
                      type  ='exit',
                      label = label_exit )

  return ( strat.st )
}

addStopLossRules <- function(strategy_name,
                             parent_long,
                             parent_short,
                             sigcol_long_ent,
                             sigcol_short_ent,
                             stop_loss,
                             add_dist         = FALSE,
                             param_set        = NULL,  #  defined if add_dist = TRUE
                             stopLossRange    = NULL,  #  defined if add_dist = TRUE
                             dist_label_long  = "StopLossLONG",
                             dist_label_short = "StopLossSHORT",
                             orderset_long    = "ocolong",
                             orderset_short   = "ocoshort" ) {

    if ( is.null(parent_long) == FALSE ) {
        strategy_name = add.rule(strategy_name,
                                 name = "ruleSignal",
                                 arguments = list(sigcol = sigcol_long_ent ,
                                                  sigval = TRUE,
                                                  replace = FALSE,
                                                  orderside = "long",
                                                  ordertype = "stoplimit",
```

```
                                            tmult = TRUE,
                                            threshold = stop_loss,
                                            orderqty = "all",
                                            orderset = orderset_long),
                                type = "chain",
                                parent = parent_long,
                                label = "StopLossLONG",
                                enabled = TRUE)
    }

    if ( is.null(parent_short) == FALSE ) {
      strategy_name = add.rule(strategy_name,
                    name = "ruleSignal",
                    arguments = list(sigcol = sigcol_short_ent,
                                     sigval = TRUE,
                                     replace = FALSE,
                                     orderside = "short",
                                     ordertype = "stoplimit",
                                     tmult = TRUE,
                                     threshold = stop_loss,
                                     orderqty = "all",
                                     orderset = orderset_short ),
                    type = "chain",
                    parent = parent_short,
                    label = "StopLossSHORT",
                    enabled = TRUE)
    }

    if ( add_dist == TRUE ) {

        if ( is.null(parent_long) == FALSE ) {
            strategy_name = add.distribution(strategy_name,
                                             paramset.label = param_set,
                                             component.type = "chain",
                                             component.label = "StopLossLONG",
                                             variable = list(threshold = stopLossRange ),
                                             label = dist_label_long )
        }

        if ( is.null(parent_short) == FALSE ) {
            strategy_name = add.distribution(strategy_name,
                                             paramset.label = param_set,
                                             component.type = "chain",
                                             component.label = "StopLossSHORT",
                                             variable = list(threshold = stopLossRange ),
                                             label = dist_label_short )
        }

        if ( is.null(parent_long) == FALSE & is.null(parent_short) == FALSE ) {
            strategy_name = add.distribution.constraint(strategy_name,
                                                        paramset.label = param_set,
                                                        distribution.label.1 = dist_label_long,
                                                        distribution.label.2 = dist_label_short,
                                                        operator = "==",
                                                        label = "StopLoss")
        }

    }

    return (strategy_name)
}


addTrailingStopRules <- function(strategy_name,
                                 parent_long,
                                 parent_short,
                                 sigcol_long_ent,
```

```
                           sigcol_short_ent,
                           trail_loss,
                           add_dist         = FALSE,
                           param_set        = NULL,  #  defined if add_dist = TRUE
                           trailLossRange   = NULL,  #  defined if add_dist = TRUE
                           dist_label_long  = "TrailLossLONG",
                           dist_label_short = "TrailLossSHORT",
                           orderset_long    = "ocolong",
                           orderset_short   = "ocoshort" ) {

if ( is.null(parent_long) == FALSE ) {
  strategy_name = add.rule(strategy_name,
                           name = "ruleSignal",
                           arguments = list(sigcol = sigcol_long_ent ,
                                            sigval = TRUE,
                                            replace = FALSE,
                                            orderside = "long",
                                            ordertype = "stoptrailing",
                                            tmult = TRUE,
                                            threshold = trail_loss,
                                            orderqty = "all",
                                            orderset = orderset_long),
                           type = "chain",
                           parent = parent_long,
                           label = "StopTrailingLong",
                           enabled = TRUE)
}

if ( is.null(parent_short) == FALSE ) {
  strategy_name = add.rule(strategy_name,
                           name = "ruleSignal",
                           arguments = list(sigcol = sigcol_short_ent,
                                            sigval = TRUE,
                                            replace = FALSE,
                                            orderside = "short",
                                            ordertype = "stoptrailing",
                                            tmult = TRUE,
                                            threshold = trail_loss,
                                            orderqty = "all",
                                            orderset = orderset_short ),
                           type = "chain",
                           parent = parent_short,
                           label = "StopTrailingShort",
                           enabled = TRUE)
}

if ( add_dist == TRUE ) {

  if ( is.null(parent_long) == FALSE ) {
    strategy_name = add.distribution(strategy_name,
                                     paramset.label = param_set,
                                     component.type = "chain",
                                     component.label = "StopTrailingLong",
                                     variable = list(threshold = trailLossRange ),
                                     label = dist_label_long )
  }

  if ( is.null(parent_short) == FALSE ) {
    strategy_name = add.distribution(strategy_name,
                                     paramset.label = param_set,
                                     component.type = "chain",
                                     component.label = "StopTrailingShort",
                                     variable = list(threshold = trailLossRange ),
                                     label = dist_label_short )
  }

  if ( is.null(parent_long) == FALSE & is.null(parent_short) == FALSE ) {
```

```
      strategy_name = add.distribution.constraint(strategy_name,
                                                  paramset.label = param_set,
                                                  distribution.label.1 = dist_label_long,
                                                  distribution.label.2 = dist_label_short,
                                                  operator = "==",
                                                  label = "TrailLoss")
    }

  }

  return (strategy_name)
}


# ********************************************* Custom ADX Indicator  ***************************

# Create your own signal for entry for ADX

adxsigBuyLong <- function(data, n) {
  print("1")
  # first condition:
  sig <- data[, "DIp.ADX"] > data[, "DIn.ADX"] & data[, "ADX.ADX"] > n
  colnames(sig) <- "buyLongSig"
  sig
}

adxsigSellLong <- function(data, n) {
  print("2")
  # first condition:
  sig <- data[, "DIp.ADX"] < data[, "DIn.ADX"] | data[, "ADX.ADX"] < n
  colnames(sig) <- "exitLongSig"
  sig
}

adxsigBuyShort <- function(data, n) {
  print("3")
  # first condition:
  sig <- data[, "DIp.ADX"] < data[, "DIn.ADX"] & data[, "ADX.ADX"] > n
  colnames(sig) <- "buyShortSig"
  sig
}

adxsigSellShort <- function(data, n) {
  print("4")
  # first condition:
  sig <- data[, "DIp.ADX"] > data[, "DIn.ADX"] | data[, "ADX.ADX"] < n
  colnames(sig) <- "exitShortSig"
  sig
}

adx_setup <- function ( strategy_name,
                        paramset_label,
                        numShares,
                        n                = 40,
                        maType           = "EMA",
                        nRange           = seq(30, 45, by=5),
                        crossOverSig     = TRUE ,
                        addRules         = TRUE,
                        addDist          = TRUE ,
                        addStopLoss      = FALSE,
                        addTrailStop     = FALSE,
                        addDistStopLoss  = FALSE,
                        stopLossRange    = NULL,
                        stop_loss        = NULL,
                        trail_stop       = NULL,
                        addDistTrailLoss = NULL,
                        trailLossRange   = NULL)  {
```

```
orderset_long  = "ocolong"
orderset_short = "ocoshort"
signalType     = setSignalType( crossOverSig )


strategy_name = add.indicator(strategy_name, name="ADX",
              arguments=list( HLC=quote(HLC(mktdata)) ),
              label="ADX")

strategy_name = add.indicator(strategy_name, name="ADX",
              arguments=list(HLC=quote(HLC(mktdata)) ),
              label="ADX2")

# *** Signal

strategy_name = add.signal(strategy_name, name="adxsigBuyLong",
            arguments = list(data = quote(mktdata) , n = n),
            label="ADX.enter.long_"
 )

strategy_name = add.signal(strategy_name, name = "sigThreshold",
                           arguments = list(column = "buyLongSig.ADX.enter.long_",
                                            threshold = 1,
                                            relationship = "eq",
                                            cross = crossOverSig),
                           label = "ADX.enter.long")

strategy_name = add.signal(strategy_name, name="adxsigSellLong",
            arguments = list(data = quote(mktdata), n=n),
            label="ADX.exit.long_"
)

strategy_name = add.signal(strategy_name, name = "sigThreshold",
                           arguments = list(column = "exitLongSig.ADX.exit.long_",
                                            threshold = 0,
                                            relationship = "eq",
                                            cross = crossOverSig),
                           label = "ADX.exit.long")

strategy_name = add.signal(strategy_name, name="adxsigBuyShort",
           arguments = list(data = quote(mktdata), n = n),
           label="ADX.enter.short_"
)

strategy_name = add.signal(strategy_name, name = "sigThreshold",
                           arguments = list(column = "buyShortSig.ADX.enter.short_",
                                            threshold = 0.5,
                                            relationship = "gt",
                                            cross = crossOverSig),
                           label = "ADX.enter.short")

strategy_name = add.signal(strategy_name, name="adxsigSellShort",
           arguments = list(data = quote(mktdata), n = n),
           label="ADX.exit.short_"
)

strategy_name = add.signal(strategy_name, name = "sigThreshold",
                           arguments = list(column = "exitShortSig.ADX.exit.short_",
                                            threshold = 0.5,
                                            relationship = "lt",
                                            cross = crossOverSig),
                           label = "ADX.exit.short")

# short side trading Rules

if ( addRules == TRUE ) {
```

```
    strategy_name = addLongRules(strategy_name,
                                 sigcol_enter = "ADX.enter.long",
                                 sigcol_exit  = "ADX.exit.long",
                                 numShares    = numShares,
                                 ordertype    = 'market',
                                 threshold    = NULL,
                                 osFUN        = "osFixedDollar",
                                 orderset     = orderset_long,
                                 label_enter  = "Enter.Long.ADX",
                                 label_exit   = "Exit.Long.ADX" )

    strategy_name = addShortRules(strategy_name,
                                  sigcol_enter = "ADX.enter.short",
                                  sigcol_exit  = "ADX.exit.short",
                                  numShares    = numShares,
                                  ordertype    = 'market',
                                  threshold    = NULL,
                                  osFUN        = "osFixedDollar",
                                  orderset     = orderset_short,
                                  label_enter  = "Enter.Short.ADX",
                                  label_exit   = "Exit.Short.ADX" )
}

if ( addStopLoss == TRUE ) {
  strategy_name = addStopLossRules(strategy_name,
                                   parent_long      = "Enter.Long.ADX",
                                   parent_short     = "Enter.Short.ADX",
                                   sigcol_long_ent  = "ADX.enter.long",
                                   sigcol_short_ent = "ADX.enter.short",
                                   stop_loss        = stop_loss,
                                   add_dist         = addDistStopLoss,
                                   param_set        = paramset_label,
                                   stopLossRange    = stopLossRange,
                                   dist_label_long  = "StopLossLONG.ADX",
                                   dist_label_short = "StopLossSHORT.ADX",
                                   orderset_long    = orderset_long,
                                   orderset_short   = orderset_short )
}

if ( addTrailStop == TRUE ) {
  strategy_name = addTrailingStopRules(strategy_name,
                                       parent_long      = "Enter.Long.ADX",
                                       parent_short     = "Enter.Short.ADX",
                                       sigcol_long_ent  = "ADX.enter.long",
                                       sigcol_short_ent = "ADX.enter.short",
                                       trail_loss       = trail_stop,
                                       add_dist         = addDistTrailLoss,
                                       param_set        = paramset_label,
                                       trailLossRange   = trailLossRange,
                                       dist_label_long  = "TrailLossLONG.ADX",
                                       dist_label_short = "TrailLossSHORT.ADX",
                                       orderset_long    = orderset_long,
                                       orderset_short   = orderset_short )
}

if ( addDist == TRUE ) {
    # Distribution setup:
    strategy_name = add.distribution(strategy = strategy_name,
                     paramset.label = paramset_label,
                     component.type = "signal",
                     component.label = "ADX.enter.long_",
                     variable = list(n = nRange),
                     label = "NRangeDist_ADX"
    )

    # Distribution setup:
    strategy_name = add.distribution(strategy = strategy_name,
```

```
                      paramset.label = paramset_label,
                      component.type = "signal",
                      component.label = "ADX.exit.long_",
                      variable = list(n = nRange),
                      label = "NNRangeDist_ADX"
      )

      # Distribution setup:
      strategy_name = add.distribution(strategy = strategy_name,
                                       paramset.label = paramset_label,
                                       component.type = "signal",
                                       component.label = "ADX.enter.short_",
                                       variable = list(n = nRange),
                                       label = "NRangeDistShEnter_ADX"
      )

      # Distribution setup:
      strategy_name = add.distribution(strategy = strategy_name,
                                       paramset.label = paramset_label,
                                       component.type = "signal",
                                       component.label = "ADX.exit.short_",
                                       variable = list(n = nRange),
                                       label = "NNRangeDistShExit_ADX"
      )
  }

  return ( list( sigList = list( enterLongSig  = "ADX.enter.long",
                                 enterShortSig = "ADX.enter.short",
                                 exitLongSig   = "ADX.exit.long",
                                 exitShortSig  = "ADX.exit.short" ),
                 strategyObj        = strategy_name,
                 strategyDistLabels = tibble(
                       param     = c("n", "n", "n", "n",
                                     "threshold", "threshold", "threshold", "threshold" ),
                       distLabel = c("NRangeDist_ADX", "NNRangeDist_ADX", "NRangeDistShEnter_ADX", "NNRangeDistShExit_ADX",
                                     "StopLossLONG.ADX", "StopLossSHORT.ADX", "TrailLossLONG.ADX", "TrailLossSHORT.ADX" )  )
           )    )

}

# ********************************************* Custom DVO Indicator  ***************************

# Custom Indicator Function - DVO function

DVO <- function(HLC, navg = 2, percentlookback = 126) {

  # Compute the ratio between closing prices to the average of high and low
  ratio <- Cl(HLC)/(Hi(HLC) + Lo(HLC))/2

  # Smooth out the ratio outputs using a moving average
  avgratio <- SMA(ratio, n = navg)

  # Convert ratio into a 0-100 value using runPercentRank()
  out <- runPercentRank(avgratio, n = percentlookback, exact.multiplier = 1) * 100
  colnames(out) <- "DVO"
  return(out)
}

dvo_Setup <- function ( strategy_name,
                        paramset_label,
                        numShares,
                        navg                = 2,
                        percentlookback     = 126,
                        navgRange           = seq(2, 6, by=2) ,
                        percentlookbackRange = seq(120, 130, by=2) ,
                        crossOverSig        = TRUE ,
                        addRules            = TRUE ,
```

```
                    addDist              = TRUE ,
                    addStopLoss          = FALSE,
                    addTrailStop         = FALSE,
                    addDistStopLoss      = FALSE,
                    stopLossRange        = NULL,
                    stop_loss            = NULL,
                    trail_stop           = NULL,
                    addDistTrailLoss     = NULL,
                    trailLossRange       = NULL)  {

  orderset_long  = "ocolong"
  orderset_short = "ocoshort"
  signalType     = setSignalType( crossOverSig )

  # Add the DVO indicator to your strategy
  strategy_name = add.indicator(strategy = strategy_name, name = "DVO",
                arguments = list(HLC = quote(HLC(mktdata)), navg = navg,
                                 percentlookback = percentlookback),
                label = "DVO_2_126")

  # Implement a sigThreshold which specifies that DVO_2_126 must be less than 20,

  strategy_name = add.signal(strategy_name, name = "sigThreshold",
              arguments = list(column = "DVO_2_126",
                               threshold = 20,
                               relationship = "lt",
                               cross = crossOverSig),
              label = "long_enter_dvo")

  # Add a sigThreshold signal to your strategy that specifies that DVO_2_126 must
  # cross above 80 and label it thresholdexit

  strategy_name = add.signal(strategy_name, name = "sigThreshold",
              arguments = list(column = "DVO_2_126",
                               threshold = 80,
                               relationship = "gt",
                               cross = crossOverSig),
              label = "long_exit_dvo")

  if ( addRules == TRUE ) {

      strategy_name = addLongRules(strategy_name,
                                   sigcol_enter = "long_enter_dvo",
                                   sigcol_exit  = "long_exit_dvo",
                                   numShares    = numShares,
                                   ordertype    = 'market',
                                   threshold    = NULL,
                                   osFUN        = "osFixedDollar",
                                   orderset     = orderset_long,
                                   label_enter  = "Enter.Long.DVO",
                                   label_exit   = "Exit.Long.DVO" )

      strategy_name = addShortRules(strategy_name,
                                    sigcol_enter = "long_exit_dvo",
                                    sigcol_exit  = "long_enter_dvo",
                                    numShares    = numShares,
                                    ordertype    = 'market',
                                    threshold    = NULL,
                                    osFUN        = "osFixedDollar",
                                    orderset     = orderset_short,
                                    label_enter  = "Enter.Short.DVO",
                                    label_exit   = "Exit.Short.DVO" )
  }

  if ( addStopLoss == TRUE ) {
    strategy_name = addStopLossRules(strategy_name,
                                     parent_long      = "Enter.Long.DVO",
```

```
                                    parent_short     = "Enter.Short.DVO",
                                    sigcol_long_ent  = "long_enter_dvo",
                                    sigcol_short_ent = "long_exit_dvo",
                                    stop_loss        = stop_loss,
                                    add_dist         = addDistStopLoss,
                                    param_set        = paramset_label,
                                    stopLossRange    = stopLossRange,
                                    dist_label_long  = "StopLossLONG.DVO",
                                    dist_label_short = "StopLossSHORT.DVO",
                                    orderset_long    = orderset_long,
                                    orderset_short   = orderset_short )
    }

    if ( addTrailStop == TRUE ) {
      strategy_name = addTrailingStopRules(strategy_name,
                                        parent_long      = "Enter.Long.DVO",
                                        parent_short     = "Enter.Short.DVO",
                                        sigcol_long_ent  = "long_enter_dvo",
                                        sigcol_short_ent = "long_exit_dvo",
                                        trail_loss       = trail_stop,
                                        add_dist         = addDistTrailLoss,
                                        param_set        = paramset_label,
                                        trailLossRange   = trailLossRange,
                                        dist_label_long  = "TrailLossLONG.DVO",
                                        dist_label_short = "TrailLossSHORT.DVO",
                                        orderset_long    = orderset_long,
                                        orderset_short   = orderset_short )
    }

    if ( addDist == TRUE ) {
        # Distribution setup:
      strategy_name = add.distribution(strategy = strategy_name,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "DVO_2_126",
                        variable = list(navg = navgRange),
                        label = "NAVGRangeDist_DMO"
        )

      strategy_name = add.distribution(strategy = strategy_name,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "DVO_2_126",
                        variable = list(percentlookback = percentlookbackRange),
                        label = "PLBRangeDist_DMO"
        )
    }

    return ( list( sigList = list( enterLongSig  = "long_enter_dvo",
                                   enterShortSig = "long_exit_dvo",
                                   exitLongSig   = "long_exit_dvo",
                                   exitShortSig  = "long_enter_dvo" ),
                   strategyObj = strategy_name,
                   strategyDistLabels = tibble(
                        param      = c("navg",
                                       "percentlookback", "threshold", "threshold", "threshold", "threshold"),
                        distLabel  = c("NAVGRangeDist_DMO", "PLBRangeDist_DMO",
                                       "StopLossLONG.DVO", "StopLossSHORT.DVO", "TrailLossLONG.DVO", "TrailLossSHORT.DVO")  )
           )   )
}
```

```
# ********************************************* RSI Indicator  ***************************

rsi_Setup <- function ( strategy_name,
                        paramset_label,
                        numShares,
                        n               = 10,
```

```
                      nRange           = seq(2, 10, by=2),
                      crossOverSig     = TRUE ,
                      addRules         = TRUE ,
                      addDist          = TRUE ,
                      addStopLoss      = FALSE,
                      addTrailStop     = FALSE,
                      addDistStopLoss  = FALSE,
                      stopLossRange    = NULL,
                      stop_loss        = NULL,
                      trail_stop       = NULL,
                      addDistTrailLoss = NULL,
                      trailLossRange   = NULL)  {

orderset_long  = "ocolong"
orderset_short = "ocoshort"
signalType    = setSignalType( crossOverSig )

# Add an indicator
strategy_name = add.indicator(strategy = strategy_name, name = "RSI",
              arguments = list(price = quote(getPrice(mktdata)), n=10, maType='EMA'),
              label="RSI_1")

# Add an indicator
strategy_name = add.indicator(strategy = strategy_name, name = "RSI",
              arguments = list(price = quote(getPrice(mktdata)), n=10, maType='EMA'),
              label="RSI_2")

# There are two signals:
# The first is when RSI is greater than 70

strategy_name = add.signal(strategy = strategy_name, name="sigThreshold",
           arguments = list(threshold=70, column="RSI_1",relationship="gt", cross=crossOverSig),
           label="RSI.gt.70")

# The second is when RSI is less than 30
strategy_name = add.signal(strategy = strategy_name, name="sigThreshold",
           arguments = list(threshold=30, column="RSI_1",relationship="lt",cross=crossOverSig),
           label="RSI.lt.30")

  if ( addRules == TRUE ) {

      strategy_name = addLongRules(strategy_name,
                                   sigcol_enter = "RSI.lt.30",
                                   sigcol_exit  = "RSI.gt.70",
                                   numShares    = numShares,
                                   ordertype    = 'market',
                                   threshold    = NULL,
                                   osFUN        = "osFixedDollar",
                                   orderset     = orderset_long,
                                   label_enter  = "Enter.Long.RSI",
                                   label_exit   = "Exit.Long.RSI" )

      strategy_name = addShortRules(strategy_name,
                                    sigcol_enter = "RSI.gt.70",
                                    sigcol_exit  = "RSI.lt.30",
                                    numShares    = numShares,
                                    ordertype    = 'market',
                                    threshold    = NULL,
                                    osFUN        = "osFixedDollar",
                                    orderset     = orderset_short,
                                    label_enter  = "Enter.Short.RSI",
                                    label_exit   = "Exit.Short.RSI" )
  }

  if ( addStopLoss == TRUE ) {
      strategy_name = addStopLossRules(strategy_name,
                                     parent_long      = "Enter.Long.RSI",
```

```
                                    parent_short     = "Enter.Short.RSI",
                                    sigcol_long_ent  = "RSI.lt.30",
                                    sigcol_short_ent = "RSI.gt.70",
                                    stop_loss        = stop_loss,
                                    add_dist         = addDistStopLoss,
                                    param_set        = paramset_label,
                                    stopLossRange    = stopLossRange,
                                    dist_label_long  = "StopLossLONG.RSI",
                                    dist_label_short = "StopLossSHORT.RSI",
                                    orderset_long    = orderset_long,
                                    orderset_short   = orderset_short )
    }

    if ( addTrailStop == TRUE ) {
        strategy_name = addTrailingStopRules(strategy_name,
                                        parent_long      = "Enter.Long.RSI",
                                        parent_short     = "Enter.Short.RSI",
                                        sigcol_long_ent  = "RSI.lt.30",
                                        sigcol_short_ent = "RSI.gt.70",
                                        trail_loss       = trail_stop,
                                        add_dist         = addDistTrailLoss,
                                        param_set        = paramset_label,
                                        trailLossRange   = trailLossRange,
                                        dist_label_long  = "TrailLossLONG.RSI",
                                        dist_label_short = "TrailLossSHORT.RSI",
                                        orderset_long    = orderset_long,
                                        orderset_short   = orderset_short )
    }

    if ( addDist == TRUE ) {
      # Add Distributions

      strategy_name = add.distribution(strategy = strategy_name,
                       paramset.label = paramset_label,
                       component.type = 'indicator',
                       component.label = 'RSI_1',
                       variable = list(n = nRange),
                       label = 'nRangeRSI')

      strategy_name = add.distribution(strategy = strategy_name,
                       paramset.label = paramset_label,
                       component.type = 'indicator',
                       component.label = 'RSI_2',
                       variable = list(n = nRange),
                       label = 'nRangeRSI2')
    }

    return ( list( sigList = list( enterLongSig  = "RSI.lt.30",
                                   enterShortSig = "RSI.gt.70",
                                   exitLongSig   = "RSI.gt.70",
                                   exitShortSig  = "RSI.lt.30" ),
                   strategyObj = strategy_name,
                   strategyDistLabels = tibble(
                     param     = c("n", "threshold", "threshold", "threshold", "threshold"),
                     distLabel = c("nRangeRSI", "StopLossLONG.RSI", "StopLossSHORT.RSI", "TrailLossLONG.RSI", "TrailLossSHORT.RS
    )   )

}

# ********************************************* EMA Indicator  ****************************

ema_Setup <- function ( strategy_name,
                        paramset_label,
                        numShares,
                        nFast            = 10,
                        nSlow            = 40,
                        fastRange        = seq(5, 20, by=5) ,
```

```
                      slowRange        = seq(40, 120, by=20),
                      crossOverSig     = TRUE ,
                      addRules         = TRUE ,
                      addDist          = TRUE ,
                      addStopLoss      = FALSE,
                      addTrailStop     = FALSE,
                      addDistStopLoss  = FALSE,
                      stopLossRange    = NULL,
                      stop_loss        = NULL,
                      trail_stop       = NULL,
                      addDistTrailLoss = NULL,
                      trailLossRange   = NULL)  {

orderset_long  = "ocolong"
orderset_short = "ocoshort"
signalType    = setSignalType( crossOverSig )

# *** EMA Indicators

my_strategy = strategy_name
n_sma_slow  = nSlow
n_sma_fa    = nFast

my_strategy <- add.indicator(strategy = my_strategy, name = "EMA",
                             arguments = list(x=quote(Cl(mktdata)), n=n_sma_fa), label= "ema10" )
my_strategy <- add.indicator(strategy = my_strategy, name = "EMA",
                             arguments = list(x=quote(Cl(mktdata)[,1]), n=n_sma_slow), label= "ema100")

# *** Signals

my_strategy <- add.signal(strategy = my_strategy, name=signalType,
                          arguments = list(columns=c("ema10","ema100"), relationship="gt"),
                          label="ema10.gt.ema100")
my_strategy <- add.signal(strategy = my_strategy, name=signalType,
                          arguments = list(column=c("ema10","ema100"),relationship="lt"),
                          label="ema10.lt.ema100")


if ( addRules == TRUE ) {

    # short side trading Rules

    my_strategy <- add.rule(strategy = my_strategy, name='ruleSignal',  label="EMA_Enter.Short",
                            arguments = list(sigcol="ema10.lt.ema100",sigval=TRUE, replace = FALSE,
                                             orderside='short', ordertype='market', orderqty = -numShares,
                                             TxnFees = 0, osFUN = "osFixedDollar" ),
                            type='enter' )

    my_strategy <- add.rule(strategy = my_strategy, name='ruleSignal',  label="EMA_Exit.Short",
                            arguments = list(sigcol="ema10.gt.ema100",sigval=TRUE, replace = FALSE,
                                             orderqty='all', ordertype='market',  orderside='short',
                                             TxnFees = 0 ),
                            type='exit' )

    # long side trading rules

    my_strategy <- add.rule(strategy = my_strategy, name='ruleSignal',  label="EMA_Enter.Long",
                            arguments = list(sigcol="ema10.gt.ema100", sigval=TRUE, replace = FALSE,
                                             ordertype='market', orderside='long', orderqty = numShares,
                                             TxnFees = 0, osFUN = "osFixedDollar" ),
                            type='enter' )

    my_strategy <- add.rule(strategy = my_strategy, name='ruleSignal',  label="EMA_Exit.Long",
                            arguments = list(sigcol="ema10.lt.ema100", sigval=TRUE, replace = FALSE,
                                             orderqty='all', ordertype='market', orderside='long',
                                             TxnFees = 0),
                            type='exit' )
```

```
  }

  if ( addDist == TRUE ) {
      # Distribution setup:
      my_strategy = add.distribution(strategy = my_strategy,
                      paramset.label = paramset_label,          # set label
                      component.type = "indicator",
                      component.label = "ema10",                # indicator label
                      variable = list(n = fastRange),          # Hyper Param to change in indicator,  fast param
                      label = "FastRangeDist_EMA"              # Instance of set label, associate w/ hyper param varied
      )

      my_strategy = add.distribution(strategy = my_strategy,
                      paramset.label = paramset_label,
                      component.type = "indicator",
                      component.label = "ema100",
                      variable = list(n = slowRange),
                      label = "SlowRangeDist_EMA"
      )

      my_strategy = add.distribution.constraint(strategy = my_strategy,
                                  paramset.label = paramset_label,          # Set Label
                                  distribution.label.1 = 'FastRangeDist_EMA',   # instance of set, ref. to by label
                                  distribution.label.2 = 'SlowRangeDist_EMA',
                                  operator = '<',
                                  label = 'LessThan_EMA'                    # constraint name
      )
  }

  return ( list( sigList = list( enterLongSig  = "ema10.gt.ema100",
                                 enterShortSig = "ema10.lt.ema100",
                                 exitLongSig   = "ema10.lt.ema100",
                                 exitShortSig  = "ema10.gt.ema100" ),
                 strategyObj = my_strategy,
                 strategyDistLabels = tibble( param     = c("n", "n"),
                                              distLabel = c("FastRangeDist_EMA", "SlowRangeDist_EMA")  )
  )   )

}

# ****************************************** SMA Indicator  ***************************

sma_Setup <- function ( strategy_name,
                        paramset_label,
                        numShares,
                        nFast            = 10,
                        nSlow            = 40,
                        fastRange        = seq(5, 20, by=5) ,
                        slowRange        = seq(40, 120, by=20),
                        crossOverSig     = TRUE ,
                        addRules         = TRUE ,
                        addDist          = TRUE ,
                        addStopLoss      = FALSE,
                        addTrailStop     = FALSE,
                        addDistStopLoss  = FALSE,
                        stopLossRange    = NULL,
                        stop_loss        = NULL,
                        trail_stop       = NULL,
                        addDistTrailLoss = NULL,
                        trailLossRange   = NULL)  {

    orderset_long  = "ocolong"
    orderset_short = "ocoshort"
    signalType    = setSignalType( crossOverSig )

    signalType = setSignalType( crossOverSig )
```

```
# *** SMA Indicators

my_strategy = strategy_name
n_sma_slow  = nSlow
n_sma_fa    = nFast

my_strategy <- add.indicator(strategy = my_strategy, name = "SMA",
                             arguments = list(x=quote(Cl(mktdata)), n=n_sma_fa), label= "ma10" )
my_strategy <- add.indicator(strategy = my_strategy, name = "SMA",
                             arguments = list(x=quote(Cl(mktdata)[,1]), n=n_sma_slow), label= "ma100")

# *** Signals

my_strategy <- add.signal(strategy = my_strategy, name=signalType,
                          arguments = list(columns=c("ma10","ma100"), relationship="gt"),
                          label="ma10.gt.ma100")
my_strategy <- add.signal(strategy = my_strategy, name=signalType,
                          arguments = list(column=c("ma10","ma100"),relationship="lt"),
                          label="ma10.lt.ma100")

if ( addRules == TRUE )
{

  my_strategy = addLongRules(my_strategy,
                           sigcol_enter = "ma10.gt.ma100",
                           sigcol_exit  = "ma10.lt.ma100",
                           numShares    = numShares,
                           ordertype    = 'market',
                           threshold    = NULL,
                           osFUN        = "osFixedDollar",
                           orderset     = orderset_long,
                           label_enter  = "Enter.Long.SMA",
                           label_exit   = "Exit.Long.SMA" )

  my_strategy = addShortRules(my_strategy,
                            sigcol_enter = "ma10.lt.ma100",
                            sigcol_exit  = "ma10.gt.ma100",
                            numShares    = numShares,
                            ordertype    = 'market',
                            threshold    = NULL,
                            osFUN        = "osFixedDollar",
                            orderset     = orderset_short,
                            label_enter  = "Enter.Short.SMA",
                            label_exit   = "Exit.Short.SMA" )
}

if ( addStopLoss == TRUE ) {
  my_strategy = addStopLossRules(my_strategy,
                              parent_long      = "Enter.Long.SMA",
                              parent_short     = "Enter.Short.SMA",
                              sigcol_long_ent  = "ma10.gt.ma100",
                              sigcol_short_ent = "ma10.lt.ma100",
                              stop_loss        = stop_loss,
                              add_dist         = addDistStopLoss,
                              param_set        = paramset_label,
                              stopLossRange    = stopLossRange,
                              dist_label_long  = "StopLossLONG.SMA",
                              dist_label_short = "StopLossSHORT.SMA",
                              orderset_long    = orderset_long,
                              orderset_short   = orderset_short )
}

if ( addTrailStop == TRUE ) {
  my_strategy = addTrailingStopRules(my_strategy,
                                  parent_long      = "Enter.Long.SMA",
                                  parent_short     = "Enter.Short.SMA",
                                  sigcol_long_ent  = "ma10.gt.ma100",
```

```
                                    sigcol_short_ent = "ma10.lt.ma100",
                                    trail_loss       = trail_stop,
                                    add_dist         = addDistTrailLoss,
                                    param_set        = paramset_label,
                                    trailLossRange   = trailLossRange,
                                    dist_label_long  = "TrailLossLONG.SMA",
                                    dist_label_short = "TrailLossSHORT.SMA",
                                    orderset_long    = orderset_long,
                                    orderset_short   = orderset_short )
    }

    if ( addDist == TRUE ) {
        # Distribution setup:
        my_strategy = add.distribution(strategy = my_strategy,
                        paramset.label = paramset_label,          # set label
                        component.type = "indicator",
                        component.label = "ma10",                # indicator label
                        variable = list(n = fastRange),          # Hyper Param to change in indicator,  fast param
                        label = "FastRangeDist_SMA"              # Instance of set label, associate w/ hyper param varied
        )

        my_strategy = add.distribution(strategy = my_strategy,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "ma100",
                        variable = list(n = slowRange),
                        label = "SlowRangeDist_SMA"
        )

        my_strategy = add.distribution.constraint(strategy = my_strategy,
                                    paramset.label = paramset_label,           # Set Label
                                    distribution.label.1 = 'FastRangeDist_SMA',    # instance of set, ref. to by label
                                    distribution.label.2 = 'SlowRangeDist_SMA',
                                    operator = '<',
                                    label = 'LessThan_SMA'                     # constraint name
        )
    }

    return ( list( sigList = list( enterLongSig  = "ma10.gt.ma100",
                                   enterShortSig = "ma10.lt.ma100",
                                   exitLongSig   = "ma10.lt.ma100",
                                   exitShortSig  = "ma10.gt.ma100" ),
                   strategyObj        = my_strategy ,
                   strategyDistLabels = tibble(
                      param     = c("n", "n",  "threshold", "threshold", "threshold", "threshold" ),
                      distLabel = c("FastRangeDist_SMA", "SlowRangeDist_SMA",
                                    "StopLossLONG.SMA", "StopLossSHORT.SMA", "TrailLossLONG.SMA", "TrailLossSHORT.SMA")  )
    )   )

}

# ********************************************* MACD Indicator  ***************************

macd_Setup <- function(strategy_name,
                       paramset_label,
                       numShares,
                       nFast = 10,
                       nSlow = 40,
                       sig   = 9,
                       fastRange = seq(10, 18, by=4) ,
                       slowRange = seq(16, 40, by=8) ,
                       sigRange  = seq(7, 10),
                       crossOverSig = TRUE ,
                       addRules     = TRUE ,
                       addDist      = TRUE ,
                       addStopLoss     = FALSE,
                       addTrailStop    = FALSE,
```

```
                    addDistStopLoss = FALSE,
                    stopLossRange   = NULL,
                    stop_loss       = NULL,
                    trail_stop      = NULL,
                    addDistTrailLoss = NULL,
                    trailLossRange   = NULL)  {

orderset_long  = "ocolong"
orderset_short = "ocoshort"
signalType    = setSignalType( crossOverSig )

stratName = strategy_name
stratName = add.indicator(strategy = stratName, name = "MACD",
              arguments = list(x=quote(Cl(mktdata)), nFast = nFast, nSlow = nSlow, sig = sig),
              label='osc')

stratName = add.signal(strategy = stratName, name="sigThreshold",
           arguments=list(column="signal.osc",relationship="gt",threshold=0,
                          cross=crossOverSig),
           label="signal.gt.zero")

stratName = add.signal(strategy = stratName, name="sigThreshold",
           arguments=list(column="signal.osc",relationship="lt",threshold=0,
                          cross=crossOverSig),
           label="signal.lt.zero")

if ( addRules == TRUE) {

  stratName = addLongRules(stratName,
                           sigcol_enter = "signal.gt.zero",
                           sigcol_exit  = "signal.lt.zero",
                           numShares    = numShares,
                           ordertype    = 'market',
                           threshold    = NULL,
                           osFUN        = "osFixedDollar",
                           orderset     = orderset_long,
                           label_enter  = "Enter.Long.MACD",
                           label_exit   = "Exit.Long.MACD" )

  stratName = addShortRules(stratName,
                            sigcol_enter = "signal.lt.zero",
                            sigcol_exit  = "signal.gt.zero",
                            numShares    = numShares,
                            ordertype    = 'market',
                            threshold    = NULL,
                            osFUN        = "osFixedDollar",
                            orderset     = orderset_short,
                            label_enter  = "Enter.Short.MACD",
                            label_exit   = "Exit.Short.MACD" )
}

if ( addStopLoss == TRUE ) {
  stratName = addStopLossRules(stratName,
                               parent_long      = "Enter.Long.MACD",
                               parent_short     = "Enter.Short.MACD",
                               sigcol_long_ent  = "signal.gt.zero",
                               sigcol_short_ent = "signal.lt.zero",
                               stop_loss        = stop_loss,
                               add_dist         = addDistStopLoss,
                               param_set        = paramset_label,
                               stopLossRange    = stopLossRange,
                               dist_label_long  = "StopLossLONG.MACD",
                               dist_label_short = "StopLossSHORT.MACD",
                               orderset_long    = orderset_long,
                               orderset_short   = orderset_short )
}
```

```
    if ( addTrailStop == TRUE ) {
      stratName = addTrailingStopRules(stratName,
                                       parent_long      = "Enter.Long.MACD",
                                       parent_short     = "Enter.Short.MACD",
                                       sigcol_long_ent  = "signal.gt.zero",
                                       sigcol_short_ent = "signal.lt.zero",
                                       trail_loss       = trail_stop,
                                       add_dist         = addDistTrailLoss,
                                       param_set        = paramset_label,
                                       trailLossRange   = trailLossRange,
                                       dist_label_long  = "TrailLossLONG.MACD",
                                       dist_label_short = "TrailLossSHORT.MACD",
                                       orderset_long    = orderset_long,
                                       orderset_short   = orderset_short )
    }

    if ( addDist == TRUE ) {

        # Distribution setup:
       stratName = add.distribution(strategy = stratName,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "osc",
                        variable = list(nFast = fastRange),
                        label = "FastRangeDist"
        )

        stratName = add.distribution(strategy = stratName,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "osc",
                        variable = list(nSlow = slowRange),
                        label = "SlowRangeDist"
        )

        stratName = add.distribution(strategy = stratName,
                        paramset.label = paramset_label,
                        component.type = "indicator",
                        component.label = "osc",
                        variable = list(sig = sigRange),
                        label = "SigRangeDist"
        )

        stratName = add.distribution.constraint(strategy = stratName,
                                    paramset.label = paramset_label,
                                    distribution.label.1 = 'FastRangeDist',
                                    distribution.label.2 = 'SlowRangeDist',
                                    operator = '<',
                                    label = 'LessThan'
        )
    }

    return ( list( sigList = list( enterLongSig  = "signal.gt.zero",
                                   exitLongSig   = "signal.lt.zero" ),
                   strategyObj = stratName ,
                   strategyDistLabels = tibble(
                     param     = c("nFast", "nSlow", "sig",  "threshold", "threshold", "threshold", "threshold"),
                     distLabel = c("FastRangeDist", "SlowRangeDist", "SigRangeDist",
                                   "StopLossLONG.MACD", "StopLossSHORT.MACD", "TrailLossLONG.MACD", "TrailLossSHORT.MACD")  )
    )   )

}

# ********************************************* BBANDs Indicator  ***************************

bband_Setup <- function(strategy_name,
                        paramset_label,
```

```
                      numShares,
                      n               = 20,
                      sd              = 2,
                      n_range         = seq(10,50,by=20),
                      sd_range        = seq(1.5,2.5,by=1),
                      crossOverSig    = TRUE ,
                      addRules        = TRUE ,
                      addDist         = TRUE ,
                      addStopLoss     = FALSE,
                      addTrailStop    = FALSE,
                      addDistStopLoss = FALSE,
                      stopLossRange   = NULL,
                      stop_loss       = NULL,
                      trail_stop      = NULL,
                      addDistTrailLoss = NULL,
                      trailLossRange   = NULL)  {

orderset_long  = "ocolong"
orderset_short = "ocoshort"
signalType     = setSignalType( crossOverSig )

strat.st = strategy_name

strat.st = add.indicator(strat.st, name = "BBands",
              arguments = list(HLC = quote(HLC(mktdata)), n=n, sd=sd, maType='SMA'),
              label='BBands')

strat.st = add.signal(strat.st, name=signalType,
           arguments=list(columns=c("Close","dn"),relationship="lt"),
           label="Cl.lt.LowerBand")

strat.st = add.signal(strategy = strat.st, name=signalType,
           arguments=list(columns=c("Close","up"),relationship="gt"),
           label="Cl.gt.UpperBand")

strat.st = add.signal(strategy = strat.st, name=signalType,
           arguments=list(columns=c("High","mavg"),relationship="gt"),
           label="Hi.Cross.Mid")

strat.st = add.signal(strategy = strat.st, name=signalType,
           arguments=list(columns=c("Low","mavg"),relationship="lt"),
           label="Lo.Cross.Mid")

if ( addRules == TRUE ) {    # only needed stand alone, but no if used as part of an
                             # ensemble indicator
  strat.st = addLongRules(strat.st,
                          sigcol_enter = "Cl.lt.LowerBand",
                          sigcol_exit  = "Hi.Cross.Mid",
                          numShares    = numShares,
                          ordertype    = 'market',
                          threshold    = NULL,
                          osFUN        = "osFixedDollar",
                          orderset     = orderset_long,
                          label_enter  = "Enter.Long",
                          label_exit   = "Exit.All_Long" )

  # strat.st = add.rule(strategy = strat.st, name='ruleSignal', enable = FALSE,
  #           arguments=list(sigcol="Cl.gt.UpperBand",sigval=TRUE, orderqty=-numShares,
  #                          ordertype='market', orderside='short', osFUN = "osFixedDollar" ),
  #           type='enter',
  #           label = "Enter.Short")

   # strat.st = add.rule(strategy = strat.st, name='ruleSignal', enable = FALSE,
  #           arguments=list(sigcol="Lo.Cross.Mid",sigval=TRUE, orderqty= 'all',
  #                          ordertype='market', orderside=NULL),type='exit',
  #         label = "Exit.All_Short")
```

```
  }

  if ( addStopLoss == TRUE ) {
        strat.st = addStopLossRules(strat.st,
                                    parent_long      = "Enter.Long",
                                    parent_short     = NULL,
                                    sigcol_long_ent  = "Cl.lt.LowerBand",
                                    sigcol_short_ent = NULL,
                                    stop_loss        = stop_loss,
                                    add_dist         = addDistStopLoss,
                                    param_set        = paramset_label,
                                    stopLossRange    = stopLossRange,
                                    dist_label_long  = "StopLossLONG",
                                    dist_label_short = "StopLossSHORT",
                                    orderset_long    = orderset_long,
                                    orderset_short   = orderset_short )
  }

  if ( addTrailStop == TRUE ) {
     strat.st = addTrailingStopRules(strat.st,
                                     parent_long      = "Enter.Long",
                                     parent_short     = NULL,
                                     sigcol_long_ent  = "Cl.lt.LowerBand",
                                     sigcol_short_ent = NULL,
                                     trail_loss       = trail_stop,
                                     add_dist         = addDistTrailLoss,
                                     param_set        = paramset_label,
                                     trailLossRange   = trailLossRange,
                                     dist_label_long  = "TrailLossLONG",
                                     dist_label_short = "TrailLossSHORT",
                                     orderset_long    = orderset_long,
                                     orderset_short   = orderset_short )
  }

  if ( addDist == TRUE ) {
      strat.st = add.distribution(strat.st,
                       paramset.label = paramset_label,
                       component.type = 'indicator',
                       component.label = 'BBands',
                       variable = list(n = n_range ),
                       label = 'n'
      )

      strat.st = add.distribution(strat.st,
                       paramset.label = paramset_label,
                       component.type = 'indicator',
                       component.label = 'BBands',
                       variable = list(sd = sd_range ),
                       label = 'sd'
      )
  }

  return ( list( sigList = list( enterLongSig  = "Cl.lt.LowerBand",
                                 enterShortSig = "Cl.gt.UpperBand",
                                 exitLongSig   = "Hi.Cross.Mid",
                                 exitShortSig  = "Lo.Cross.Mid" ),
                 strategyObj = strat.st ,
                 strategyDistLabels = tibble(
                      param     = c("n", "sd" , "threshold", "threshold", "threshold", "threshold" ),
                      distLabel = c("n", "sd" , "StopLossLONG", "StopLossSHORT", "TrailLossLONG", "TrailLossSHORT")
                  )
  )   )

}


# ************************************ Voting Ensemble Custom Indicator  **********************
```

```
createDynSigFormula <- function( mean_long_enter_signals, long_enter_signals ) {

  formula_voting_long_enter = "( sum("
  for ( n in 1:length(long_enter_signals) )  {
    if ( n == 1) {
      formula_voting_long_enter = paste(formula_voting_long_enter, " ", long_enter_signals[[n]], sep="" )
    } else {
      formula_voting_long_enter = paste(formula_voting_long_enter, " + ", long_enter_signals[[n]],  sep="" )
    }
  }

  formula_voting_long_enter = paste(formula_voting_long_enter, ", na.rm = TRUE ) >= ", mean_long_enter_signals, " )", sep="")
  print(formula_voting_long_enter)
  return (formula_voting_long_enter )

}

voting_setup <- function(strategy_name,
                         numShares,
                         signals_vec,
                         crossOverSig = TRUE ) {

    signalType = setSignalType( crossOverSig )

    long_enter_signals = list()
    short_enter_signals = list()
    long_exit_signals = list()
    short_exit_signals = list()

    # Aggregate Signals by type
    for( i in 1:length(signals_vec) )  {

        if ( is.null( signals_vec[i][[ 'enterLongSig' ]] ) == FALSE ) {
          long_enter_signals = append( long_enter_signals, signals_vec[i][[ 'enterLongSig' ]] )

        } else if ( is.null( signals_vec[i][[ 'enterShortSig' ]] ) == FALSE ) {
          short_enter_signals = append( short_enter_signals, signals_vec[i][[ 'enterShortSig' ]] )

        } else if ( is.null( signals_vec[i][[ 'exitLongSig' ]] ) == FALSE ) {
          long_exit_signals = append( long_exit_signals, signals_vec[i][[ 'exitLongSig' ]] )

        } else if ( is.null( signals_vec[i][[ 'exitShortSig' ]] ) == FALSE ) {
          short_exit_signals = append( short_exit_signals, signals_vec[i][[ 'exitShortSig' ]] )

        } else {
          print("Unsupported signal type")
          return (-1)
        }
    }

    # Create Custom Formulas for Voting Signals
    # sum(sig1 + sig2 + .. sign) > length(sig1 + sig2 + .. sign)/2

    mean_long_enter_signals  = length( long_enter_signals )/2
    mean_short_enter_signals = length( short_enter_signals )/2
    mean_long_exit_signals   = length( long_exit_signals )/2
    mean_short_exit_signals  = length( short_exit_signals )/2

    formula_voting_long_enter  = createDynSigFormula( mean_long_enter_signals, long_enter_signals)
    formula_voting_short_enter = createDynSigFormula( mean_short_enter_signals, short_enter_signals)
    formula_voting_long_exit   = createDynSigFormula( mean_long_exit_signals, long_exit_signals)
    formula_voting_short_exit  = createDynSigFormula( mean_short_exit_signals, short_exit_signals)

    # create signals

    strategy_name = add.signal(strategy_name, name = "sigFormula",
               arguments = list(formula = formula_voting_long_enter,
```

```
                                cross = crossOverSig),
              label = "long_enter_voting")

    strategy_name = add.signal(strategy_name, name = "sigFormula",
              arguments = list(formula = formula_voting_long_exit,
                                cross = crossOverSig),
              label = "long_exit_voting")

    strategy_name = add.signal(strategy_name, name = "sigFormula",
              arguments = list(formula = formula_voting_short_enter,
                                cross = crossOverSig),
              label = "short_enter_voting")

    strategy_name = add.signal(strategy_name, name = "sigFormula",
              arguments = list(formula = formula_voting_short_exit,
                                cross = crossOverSig),
              label = "short_exit_voting")

    # create long and short trading rules

    strategy_name = add.rule(strategy = strategy_name, name='ruleSignal',
             arguments=list(sigcol="short_enter_voting",sigval=TRUE, orderqty=-numShares,
                            ordertype='market', orderside='short', osFUN = "osFixedDollar" ),
             type='enter',
             label = "Enter.Short_Voting")

    strategy_name = add.rule(strategy = strategy_name, name='ruleSignal',
             arguments=list(sigcol="short_exit_voting",sigval=TRUE, orderqty= 'all',
                            ordertype='market', orderside=NULL),type='exit',
             label = "Exit.All_Short_Voting")


    strategy_name = add.rule(strategy = strategy_name, name='ruleSignal',
             arguments=list(sigcol="long_enter_voting",sigval=TRUE, orderqty=numShares,
                            ordertype='market', orderside='long', threshold=NULL, osFUN = "osFixedDollar"),
             type='enter',
             label = "Enter.Long_Voting")

    strategy_name =  add.rule(strategy = strategy_name, name='ruleSignal',
             arguments=list(sigcol="long_exit_voting",sigval=TRUE, orderqty= 'all',
                            ordertype='market', orderside=NULL), type='exit',
             label = "Exit.All_Long_Voting")

    return (strategy_name)

}

# ************************************************************************************************
# ********************Functions to execute WFA and save results in tibbles *********************
# ************************************************************************************************

runStrategiesMultipleWFA <- function (strategies,
                                      strategies_args,
                                      initEq_var,
                                      stock.st_var,
                                      initDate_var,
                                      periodTrainLengthRange,
                                      periodTestLengthRange ,
                                      period_wfa_var,
                                      opt_metric_var,
                                      output_fil_var = NULL,
                                      addStopLoss    = FALSE,
                                      addTrailStop   = FALSE ) {
    df_sum_wfa_all_tbl = tibble()

    for( i in 1:length(periodTrainLengthRange) )  {
```

```
      for( j in 1:length(periodTestLengthRange) )  {

        cat("******** Executing WFA n Strategies,  Traning Period = ", periodTrainLengthRange[i],
            " Testing Period = ", periodTestLengthRange[j], " Period = ", period_wfa, "\n")

        wfa_sum_all = runStrategies (strategies            = strategies,
                                     strategies_args       = strategies_args,
                                     run_type              = "wfa",
                                     initEq_var            = initEq_var,
                                     stock.st_var          = stock.st_var,
                                     initDate_var          = initDate_var,
                                     periodTrainLength_var = periodTrainLengthRange[i],
                                     periodTestLength_var  = periodTestLengthRange[j],
                                     period_wfa_var        =  period_wfa_var,
                                     opt_metric_var        = opt_metric_var,
                                     addStopLoss           = addStopLoss,
                                     addTrailStop          = addTrailStop )

        df_sum_wfa_all_tbl = bind_rows(df_sum_wfa_all_tbl, wfa_sum_all[[1]] )
      }
    }

    if ( is.null(output_fil_var) == FALSE ) {
       write.csv(df_sum_wfa_all_tbl, file=output_fil_var)
    }

    return(df_sum_wfa_all_tbl)
}

runStrategies <- function(strategies,
                          strategies_args,
                          run_type,
                          initEq_var,
                          stock.st_var,
                          initDate_var,
                          append_portf_name     = NULL,
                          output_fil_var        = NULL,
                          periodTrainLength_var = 36,
                          periodTestLength_var  = 12,
                          period_wfa_var        = "months",
                          opt_metric_var        = "Net.Trading.PL",
                          cleanup_var           = TRUE,
                          addStopLoss           = FALSE,
                          addTrailStop          = FALSE,
                          addDistStopLoss       = FALSE,  # only applicable in paramset runs
                          addDistTrailLoss      = FALSE ) # only applicable in paramset runs
{
    crossOverSig = TRUE # override for voting, need signals non NA

    if ( run_type == "wfa" ) {
      add_rules = TRUE
      add_dist  = TRUE
      addDistStopLoss = FALSE
      addDistTrailLoss = FALSE
    } else if ( run_type == "paramset") {
      add_rules = TRUE
      add_dist  = TRUE
    } else if ( run_type == "regular" ) {
      add_rules = TRUE
      add_dist  = FALSE
      addDistStopLoss  = FALSE
      addDistTrailLoss = FALSE
    } else {
      print("Unsupported run_type")
      return (-1)
    }
```

```
res_sum_wfa_all_df = tibble()
res_sum_wfa_all_results = list()
res_sum_nonopt_all_df = tibble()
opt_results_all_tbl = tibble()
opt_results_sum_stats_tbl = tibble()

for ( n in 1:length(strategies)) {

  strat.st = strategies[n]

  # Check if VOTING Indicator

  if ( strat.st == "VOTING" ) {
     add_rules = FALSE
     add_dist  = FALSE
     crossOverSig = FALSE
  }

  # Set Portfolio, Account, Order objects...

  portfolio_name = strat.st
  if ( is.null(append_portf_name) == FALSE ) {
      portfolio_name = paste(portfolio_name, "_", append_portf_name, sep="")
  }

  suppressWarnings(rm.strat(  portfolio_name ))

  initPortf(name     = portfolio_name,
            symbols  = stock.st_var,
            initDate = initDate_var,
            currency = 'USD')
  initAcct(name       = portfolio_name,
           portfolios = portfolio_name,
           initDate   = initDate_var,
           initEq     = initEq_var)
  initOrders(portfolio = portfolio_name,
             initDate  = initDate_var)

  # **** Set up Strategy

  paramset_label <- 'test_set'
  my_strategy = strategy(name = strat.st, store=TRUE)
  numShares = 100
  signals_vec = vector()
  strategyDistLabels = NULL

  if ( strat.st == "ADX" | strat.st == "VOTING" ) {
      resultsObj = adx_setup( strategy_name = my_strategy,
                          paramset_label,
                          numShares,
                          n                  =  strategies_args$adx$n ,
                          maType             = "EMA",
                          nRange             = strategies_args$adx$nRange,
                          crossOverSig       = crossOverSig,
                          addRules           = add_rules,
                          addDist            = add_dist,
                          addStopLoss        = addStopLoss,
                          addTrailStop       = addTrailStop,
                          addDistStopLoss    = addDistStopLoss,
                          addDistTrailLoss   = addDistTrailLoss,
                          stopLossRange      = strategies_args$adx$stopLossRange,
                          trailLossRange     = strategies_args$adx$trailLossRange,
                          stop_loss          = strategies_args$adx$stopLoss,
                          trail_stop         = strategies_args$adx$trailLoss )
      signals_vec = append(signals_vec, resultsObj$sigList)
      my_strategy = resultsObj$strategyObj
      strategyDistLabels = resultsObj$strategyDistLabels
```

```
    }

    if ( strat.st == "DVO" | strat.st == "VOTING" ) {
        resultsObj = dvo_Setup( strategy_name = my_strategy,
                                paramset_label,
                                numShares,
                                navg                = strategies_args$dvo$navg,
                                percentlookback     = strategies_args$dvo$percentlookback,
                                navgRange           = strategies_args$dvo$navgRange ,
                                percentlookbackRange = strategies_args$dvo$percentlookbackRange,
                                crossOverSig        = crossOverSig,
                                addRules            = add_rules,
                                addDist             = add_dist,
                                addStopLoss         = addStopLoss,
                                addTrailStop        = addTrailStop,
                                addDistStopLoss     = addDistStopLoss,
                                addDistTrailLoss    = addDistTrailLoss,
                                stopLossRange       = strategies_args$dvo$stopLossRange,
                                trailLossRange      = strategies_args$dvo$trailLossRange,
                                stop_loss           = strategies_args$dvo$stopLoss,
                                trail_stop          = strategies_args$dvo$trailLoss )
        signals_vec = append(signals_vec, resultsObj$sigList)
        my_strategy = resultsObj$strategyObj
        strategyDistLabels = resultsObj$strategyDistLabels
    }

    if ( strat.st == "RSI" | strat.st == "VOTING" ) {
        resultsObj = rsi_Setup( strategy_name = my_strategy,
                                paramset_label,
                                numShares,
                                n                = strategies_args$rsi$n,
                                nRange           = strategies_args$rsi$nRange,
                                crossOverSig     = crossOverSig,
                                addRules         = add_rules,
                                addDist          = add_dist ,
                                addStopLoss      = addStopLoss,
                                addTrailStop     = addTrailStop,
                                addDistStopLoss  = addDistStopLoss,
                                addDistTrailLoss = addDistTrailLoss,
                                stopLossRange    = strategies_args$rsi$stopLossRange,
                                trailLossRange   = strategies_args$rsi$trailLossRange,
                                stop_loss        = strategies_args$rsi$stopLoss,
                                trail_stop       = strategies_args$rsi$trailLoss )
        signals_vec = append(signals_vec, resultsObj$sigList)
        my_strategy = resultsObj$strategyObj
        strategyDistLabels = resultsObj$strategyDistLabels
    }

   if ( strat.st == "SMA" | strat.st == "VOTING" ) {
       resultsObj = sma_Setup (strategy_name    = my_strategy,
                               paramset_label   = paramset_label,
                               numShares        = numShares,
                               nFast            = strategies_args$sma$nFast,
                               nSlow            = strategies_args$sma$nSlow,
                               fastRange        = strategies_args$sma$fastRange ,
                               slowRange        = strategies_args$sma$slowRange ,
                               crossOverSig     = crossOverSig,
                               addRules         = add_rules,
                               addDist          = add_dist ,
                               addStopLoss      = addStopLoss,
                               addTrailStop     = addTrailStop,
                               addDistStopLoss  = addDistStopLoss,
                               addDistTrailLoss = addDistTrailLoss,
                               stopLossRange    = strategies_args$sma$stopLossRange,
                               trailLossRange   = strategies_args$sma$trailLossRange,
                               stop_loss        = strategies_args$sma$stopLoss,
                               trail_stop       = strategies_args$sma$trailLoss )
```

```
    signals_vec = append(signals_vec, resultsObj$sigList)
    my_strategy = resultsObj$strategyObj
    strategyDistLabels = resultsObj$strategyDistLabels
}

if ( strat.st == "BBAND" | strat.st == "VOTING" ) {
    resultsObj = bband_Setup(strategy_name  = my_strategy,
                             paramset_label  = paramset_label,
                             numShares       = numShares,
                             n               = strategies_args$bband$n,
                             sd              = strategies_args$bband$sd,
                             n_range         = strategies_args$bband$n_range,
                             sd_range        = strategies_args$bband$sd_range,
                             crossOverSig    = crossOverSig,
                             addRules        = add_rules,
                             addDist         = add_dist,
                             addStopLoss     = addStopLoss,
                             addTrailStop    = addTrailStop,
                             addDistStopLoss = addDistStopLoss,
                             addDistTrailLoss = addDistTrailLoss,
                             stopLossRange    = strategies_args$bband$stopLossRange,
                             trailLossRange   = strategies_args$bband$trailLossRange,
                             stop_loss        = strategies_args$bband$stopLoss,
                             trail_stop       = strategies_args$bband$trailLoss )
    signals_vec = append(signals_vec, resultsObj$sigList)
    my_strategy = resultsObj$strategyObj
    strategyDistLabels = resultsObj$strategyDistLabels
}

if ( strat.st == "MACD" | strat.st == "VOTING" ) {
    resultsObj = macd_Setup (strategy_name = my_strategy,
                             paramset_label = paramset_label,
                             numShares        = numShares,
                             nFast            = strategies_args$macd$nFast,
                             nSlow            = strategies_args$macd$nSlow,
                             sig              = strategies_args$macd$sig,
                             fastRange        = strategies_args$macd$fastRange,
                             slowRange        = strategies_args$macd$slowRange,
                             sigRange         = strategies_args$macd$sigRange,
                             crossOverSig     = crossOverSig,
                             addRules         = add_rules,
                             addDist          = add_dist,
                             addStopLoss      = addStopLoss,
                             addTrailStop     = addTrailStop,
                             addDistStopLoss  = addDistStopLoss,
                             addDistTrailLoss = addDistTrailLoss,
                             stopLossRange    = strategies_args$macd$stopLossRange,
                             trailLossRange   = strategies_args$macd$trailLossRange,
                             stop_loss        = strategies_args$macd$stopLoss,
                             trail_stop       = strategies_args$macd$trailLoss )
    signals_vec = append(signals_vec, resultsObj$sigList)
    my_strategy = resultsObj$strategyObj
    strategyDistLabels = resultsObj$strategyDistLabels
}

if ( strat.st == "VOTING" ) {

    my_strategy = voting_setup(my_strategy,
                               numShares,
                               signals_vec,
                               crossOverSig = TRUE )

    test_init <- applyIndicators(my_strategy, mktdata = OHLC(QQQ) )
    test <- applySignals(strategy = my_strategy, mktdata = test_init)
}

if ( run_type == "wfa" ) {
```

```
      periodTrainLength <- periodTrainLength_var
      periodTestLength  <- periodTestLength_var
      period_wfa = period_wfa_var
      opt_metric = opt_metric_var

      .obj.func <- function(x)
      { print(x)
        which(x==max(x))
      }

       # *** Execute Single WFA

      suppressWarnings ( rm(res_sum_wfa) )
      res_sum_wfa    = runWFA(strategy_name       = strat.st,
                              paramset_label      = paramset_label,
                              portfolio_name      = portfolio_name,
                              account_name        = portfolio_name,
                              period_wfa          = period_wfa,
                              period_train_length = periodTrainLength,
                              period_test_length  = periodTestLength,
                              obj_func            = .obj.func,
                              cleanup             = cleanup_var,
                              verbose             = FALSE,
                              opt_metric          = opt_metric,
                              nsamples            = 0 )

      res_sum_wfa_all_df = bind_rows(res_sum_wfa_all_df, res_sum_wfa$summary_df_tbl )

      if ( cleanup_var == FALSE ) {
          res_sum_wfa_all_results = append(res_sum_wfa_all_results,
                                           list(strat.st = res_sum_wfa$results ) )
      }

  }
  else if ( run_type == "paramset") {

    results <- apply.paramset(strategy.st    = my_strategy ,
                              verbose        = FALSE,
                              paramset.label = paramset_label,
                              portfolio.st   = portfolio_name,
                              account.st     = portfolio_name,
                              nsamples       = 0)

    # Extra optimum hyper parameters from simulation
    df_opt = results$tradeStats %>%  filter( Max.Drawdown == max(Max.Drawdown) ) %>% slice_head( n =1 )
    opt_params = list()

    for ( x in 1:nrow(strategyDistLabels) ) {
       distLabel = strategyDistLabels[x, "distLabel"][[1]]
       param     = strategyDistLabels[x, "param"][[1]]
       opt_param_value = df_opt[1, distLabel ][[1]]
       ll = list()
       ll[param] = opt_param_value
       opt_params = append(opt_params, ll)
    }

    df_results = tibble( strategy = c(strat.st), trade_stats = list(results$tradeStats), opt_params = list(opt_params) )
    opt_results_all_tbl = bind_rows(opt_results_all_tbl, df_results)
    opt_results_sum_stats_tbl = bind_rows(opt_results_sum_stats_tbl , df_opt %>% select( Portfolio:End.Equity))

  } else if ( run_type == "regular" ) {

    # Execute the backtest
    applyStrategy(strategy = my_strategy, portfolios = portfolio_name)

    # Always run these three commands when the backtest has run:
    updatePortf(Portfolio = portfolio_name)
```

```
        updateAcct(name = portfolio_name)
        updateEndEq(Account = portfolio_name)

        # Save trade statistics
        tstats <- tradeStats(Portfolios = portfolio_name)

        # Save all tstats results
        res_sum_nonopt_all_df = bind_rows(res_sum_nonopt_all_df, tstats )

        rm(tstats)
      }

    } # end of for

    if ( run_type == "wfa" ) {
        if ( is.null(output_fil_var) == FALSE ) {
          write.csv(res_sum_wfa_all_df, file=output_fil_var)
        }
        return ( list(res_sum_wfa_all_df, res_sum_wfa_all_results ) )

    } else if ( run_type == "paramset") {
      return ( list(data = opt_results_all_tbl, sum_stats_df = opt_results_sum_stats_tbl) )

    } else if ( run_type == "regular" ) {
        return (res_sum_nonopt_all_df)

    } else {
      print("Unsupported run_type")
      return (-1)
    }

} # end of function


runWFA <- function(strategy_name,
                   paramset_label,
                   portfolio_name,
                   account_name,
                   period_wfa,
                   period_train_length,
                   period_test_length,
                   obj_func,
                   search_pattern_for_train_datasets = "T000000",
                   obj_args = list(x=quote(tradeStats.list$Net.Trading.PL)),
                   nsamples = 0,
                   anchored = FALSE,
                   verbose = TRUE,
                   savewf = FALSE ,
                   cleanup = TRUE ,
                   opt_metric = "Net.Trading.PL" ) {

    if ( ! ( period_wfa == "months" | period_wfa == "years"  ) ) {
       print("Only months and years are supported for now for period_wfa")
      return(-1)
    }

    results <- walk.forward(
      strategy.st    = strategy_name,
      paramset.label = paramset_label,
      portfolio.st   = portfolio_name,
      account.st     = account_name,
      period         = period_wfa,
      k.training     = period_train_length,
      k.testing      = period_test_length,
      obj.func       = obj_func,
      obj.args       = obj_args,
      nsamples       = nsamples,
```

```
  audit.prefix  = 'wfa',
  anchored      = anchored,
  verbose       = verbose,
  savewf        = savewf,
  saveenv       = TRUE ,      # NOTE: Need to set this to TRUE to access
                              # training set results (audit).
)

# Plot WFA:
# chart.forward(results)

sorted_names <- sort(names(results))
list_results <- mget(sorted_names[  grep( search_pattern_for_train_datasets,
                                          sorted_names ) ], results)

# extract num training period
numTrainPeriods = length( list_results )

# extract num testing periods
testing_periods_list = results$testing.parameters$testing.timespan
num_testing_periods = length(testing_periods_list)

# List of metrics output by quantstrat

metric_list = list ("Num.Txns", "Num.Trades" ,  "Net.Trading.PL",  "Avg.Trade.PL" ,
                    "Med.Trade.PL" ,      "Largest.Winner" ,    "Largest.Loser" ,
                    "Gross.Profits" , "Gross.Losses",    "Std.Dev.Trade.PL",   "Std.Err.Trade.PL" ,
                    "Percent.Positive",  "Percent.Negative",  "Profit.Factor",  "Avg.Win.Trade",
                    "Med.Win.Trade",  "Avg.Losing.Trade",   "Med.Losing.Trade",   "Avg.Daily.PL" ,
                    "Med.Daily.PL",  "Std.Dev.Daily.PL" ,  "Std.Err.Daily.PL" ,  "Ann.Sharpe" ,
                    "Max.Drawdown", "Profit.To.Max.Draw", "Avg.WinLoss.Ratio",  "Med.WinLoss.Ratio" ,
                    "Max.Equity", "Min.Equity"  )

metric_list_wfe <- vector()
metric_list_mean_train <- vector()
metric_list_mean_test  <- vector()

for( j in 1:length(metric_list) )  {
    curr_metric = metric_list[[j]]
    metric_list_wfe[j] = paste("wfe_",  curr_metric, sep="")
    metric_list_mean_train[j] = paste("ave_train_",  curr_metric, sep="")
    metric_list_mean_test[j]  = paste("ave_test_",  curr_metric, sep="")
}

# ***Annualized results to compare apples to apples

ann_multiplier_test = -1
ann_multiplier_train = -1

if ( period_wfa == "months" ) {
  # Test results are accumulative per period, 6 months period, will need to
  # be multiply by 2, 3 months by 4, 18 month * 1/1.5, 24 month by 1/2
  ann_multiplier_test = 1/ ( period_test_length/12 )

  # Train results
  ann_multiplier_train = ( 1/ ( period_train_length/12 ) )
} else {
  # years
  ann_multiplier_test = 1/ ( period_test_length)
  ann_multiplier_train = ( 1/ ( period_train_length ) )
}

# Create WFE for each metric created by quantstrat and save in a tibble

opt_values_metric_all <- vector()
opt_values_metric_all_mean_train <- vector()
opt_values_metric_all_mean_test <- vector()
```

```
for( j in 1:length(metric_list) )  {
  curr_metric = metric_list[[j]]
  opt_values_metric <- vector()

  for(i in 1:length(list_results))
  {
    temp <- mget(names(list_results[[i]]), list_results[[i]])
    temp_audit           <- mget(names(temp$audit), temp$audit)
    train_stats          <- temp_audit$tradeStats
    opt_combo_idx        <- temp_audit$param.combo.nr
    opt_train_obj        <- train_stats[opt_combo_idx, curr_metric]
    opt_values_metric[i] <- opt_train_obj
  }

  # Create Annualized metrics

  opt_train_metric <- opt_values_metric[i]
  ann_mean_opt     <- ( opt_train_metric/numTrainPeriods ) * ann_multiplier_train

  opt_test_metric      <- results$tradeStats[, curr_metric]
  ann_opt_test_metric  <- ( opt_test_metric/num_testing_periods ) * ann_multiplier_test

  wfe <- ann_opt_test_metric/ann_mean_opt

  opt_values_metric_all[j] = wfe
  opt_values_metric_all_mean_train[j] = ann_mean_opt
  opt_values_metric_all_mean_test[j] = ann_opt_test_metric

}

# Create Summary Data Frame to save metadata and results of WFA, and
# all WFE for all metrics

df_sum_metric_results = bind_rows( setNames( opt_values_metric_all,
                                             metric_list_wfe ) )

df_sum_mean_train_results = bind_rows( setNames( opt_values_metric_all_mean_train,
                                                 metric_list_mean_train ) )

df_sum_mean_test_results = bind_rows( setNames( opt_values_metric_all_mean_test,
                                                metric_list_mean_test ) )

summary_df_tbl = tibble(
  strategy          = strategy_name,
  period_train_len  = as.integer( period_train_length ),
  period_test_len   = as.integer( period_test_length ),
  period            = period_wfa,
  num_train_periods = numTrainPeriods,
  num_test_periods  = num_testing_periods,
  opt_metric        = opt_metric
)

summary_df_tbl = bind_cols(summary_df_tbl,  df_sum_metric_results)
summary_df_tbl = bind_cols(summary_df_tbl,  df_sum_mean_train_results)
summary_df_tbl = bind_cols(summary_df_tbl,  df_sum_mean_test_results)

if ( cleanup == TRUE ) {
  rm(results)
  rm(list_results)
  rm(testing_periods_list)
  rm(opt_values_metric_all)

  return ( list(summary_df_tbl = summary_df_tbl) )
} else {
  return ( list(summary_df_tbl = summary_df_tbl, results = results) )
}
```

```
}


runMultipleWFA <- function(period_train_len_range, period_test_len_range,
                           strategy_name, paramset_label, portfolio_name, account_name,
                           period_wfa, obj_func,
                           search_pattern_for_train_datasets = "T000000",
                           obj_args = list(x=quote(tradeStats.list$Net.Trading.PL)),
                           nsamples = 0, anchored = FALSE, verbose = TRUE,
                           savewf = FALSE , opt_metric = "Net.Trading.PL" ) {

  df_sum_wfa_all_tbl = tibble()

  for( i in 1:length(period_train_len_range) )  {

    for( j in 1:length(period_test_len_range) )  {

      cat("******** Executing WFA,  Traning Period = ", period_train_len_range[i],
          " Testing Period = ", period_test_len_range[j], " Period = ", period_wfa, "\n")

      res_sum_wfa_list = runWFA(strategy_name      = strategy_name,
                                paramset_label     = paramset_label,
                                portfolio_name     = portfolio_name,
                                account_name       = account_name,
                                period_wfa         = period_wfa,
                                period_train_length = period_train_len_range[i],
                                period_test_length  = period_test_len_range[j],
                                obj_func           = obj_func,
                                search_pattern_for_train_datasets = search_pattern_for_train_datasets,
                                obj_args           = obj_args,
                                nsamples           = nsamples,
                                anchored           = anchored,
                                verbose            = verbose,
                                savewf             = savewf,
                                opt_metric         = opt_metric )

      df_sum_wfa_all_tbl = bind_rows(df_sum_wfa_all_tbl, res_sum_wfa_list$summary_df_tbl )
    }
  }

  return(df_sum_wfa_all_tbl)
}

# ************************************************************************************************
# *********************Functions to draw charts and visualizations          **********************
# ************************************************************************************************

drawStrategiesHeatMaps <- function(opt_sum_results,
                                   strategies,
                                   metric_1, metric_1_label ) {

  for ( i in 1:length( strategies) ) {

    strategy_trade_stats = ( opt_sum_results$data %>% filter( strategy == strategies[i]) )[1, 'trade_stats'] [[1]][[1]]

    # Profit/MaxDD
    drawHeatMap(strategy_trade_stats[metric_1][[1]],
                ( strategy_trade_stats %>% select(1) )[[1]],
                ( strategy_trade_stats %>% select(2) )[[1]],
                metric_1_label,
                colnames(strategy_trade_stats)[1],
                colnames(strategy_trade_stats)[2], strategies[i] )
  }

}

drawHeatMap <- function(metric, paramOne, paramTwo, metricTitle,
```

```
                        paramOneTitle, paramTwoTitle, symbol_var)
{
  z <- tapply(X=metric,
              INDEX=list(paramOne,
                         paramTwo),
              FUN=mean)

  x <- as.numeric(rownames(z))
  y <- as.numeric(colnames(z))

  filled.contour(x=x,y=y,z=z,color = heat.colors,
                 xlab=paramOneTitle,ylab=paramTwoTitle)
  myTitle <- paste0(metricTitle, ": ", symbol_var)
  title(myTitle)
}

# *** Trade Related Analysis

tradeRelatedAnalysis <- function(supported_strategies, results_set )  {

  combine_sum = tibble()
  for ( i in 1:length(supported_strategies) ) {
    strategy_n = supported_strategies[i]
    results_n  = results_set[i]
    # trade relates
    tab.trades <- ( results_set[i][[1]]$data %>% filter(strategy == strategy_n) )$trade_stats[[1]]  %>%
      filter( Max.Drawdown == max(Max.Drawdown) )  %>%
      slice_head( n =1 ) %>%
      mutate(Trades = Num.Trades,
             Win.Percent = Percent.Positive,
             Loss.Percent = Percent.Negative,
             WL.Ratio = Percent.Positive/Percent.Negative) %>%
      select(Trades, Win.Percent, Loss.Percent, WL.Ratio) %>%
      mutate(strategy = strategy_n ) %>%
      select(strategy, 2:ncol(.))
    combine_sum = bind_rows(combine_sum, tab.trades)
  }

  knitr::kable ( combine_sum )
}

# profit related

profitRelatedAnalysis <- function(supported_strategies, results_set )  {

  combine_sum = tibble()
  for ( i in 1:length(supported_strategies) ) {
    strategy_n = supported_strategies[i]
    results_n  = results_set[i]
    # trade relates
    tab.trades <- ( results_set[i][[1]]$data %>% filter(strategy == strategy_n) )$trade_stats[[1]]  %>%
      filter( Max.Drawdown == max(Max.Drawdown) )  %>%
      slice_head( n =1 ) %>%
      select(Net.Trading.PL, Gross.Profits, Gross.Losses, Profit.Factor) %>%
      mutate(strategy = strategy_n ) %>%
      select(strategy, 2:ncol(.))
    combine_sum = bind_rows(combine_sum, tab.trades)
  }

  knitr::kable ( combine_sum )
}

# *** Averages

averagesRelatedAnalysis <- function(supported_strategies, results_set )  {

  combine_sum = tibble()
```

```
  for ( i in 1:length(supported_strategies) ) {
    strategy_n = supported_strategies[i]
    results_n  = results_set[i]
    # trade relates
    tab.trades <- ( results_set[i][[1]]$data %>% filter(strategy == strategy_n) )$trade_stats[[1]]  %>%
      filter( Max.Drawdown == max(Max.Drawdown) )  %>%
      slice_head( n =1 ) %>%
      select(Avg.Trade.PL, Avg.Win.Trade, Avg.Losing.Trade, Avg.WinLoss.Ratio) %>%
      mutate(strategy = strategy_n ) %>%
      select(strategy, 2:ncol(.))
    combine_sum = bind_rows(combine_sum, tab.trades)
  }

  knitr::kable ( combine_sum )
}

performanceStatsAnalysis <- function( supported_strategies, rets_strategies) {
  sum_results_tbl = tibble()
  metricsNames=c(
    "Cumulative Return",
    "Annualized Return" )
  for ( i in 1:length(supported_strategies)) {
    strategy_name = supported_strategies[i]
    tab.perf <- table.Arbitrary(rets_strategies[i][[1]],
                                metrics=c(
                                  "Return.cumulative",
                                  "Return.annualized"),
                                metricsNames=metricsNames)

    sum_tbl  = tibble( as.data.frame(tab.perf), rownames=metricsNames ) %>%
      #sum_tbl  = tibble( as.data.frame(tab.perf) ) %>%
      column_to_rownames( var = "rownames") %>%
      rename( !!quo_name(strategy_name) := QQQ.DailyEqPL)
    if ( i == 1 ) {
      sum_results_tbl = sum_tbl
    } else {
      sum_results_tbl =  bind_cols(sum_results_tbl, sum_tbl)
    }
  }
  knitr::kable(sum_results_tbl)
}
```