Vincenzo Brigandì
brigandi.vincenzo@yahoo.com
4th March 2024

Quantitative Analyst project TMC

MEAN REVERSION MODEL

Strategy definition

The goal of this exercise is to backest the proposed trading strategy and to check whether it is possible to improve it by tuning the parameters related to the entry signal.
I start backtesting the proposed strategy as descripted, then I will perform some parameters' tuning to check if it is possible to improve it.
The Mean Reversion *Entry signal* is the intersection of two signals S1 and S2. When they are both triggered the trading system enters a Long or Short position in the asset.
S1_long is triggered when the price of an instrument is simultaneously below:

- the 5% percentile calculated on the last 10 trading days (as convention for 2 weeks)
- the 15% percentile calculated on the last 21 trading days (as convention for 1 month)
- the 25% percentile calculated on the last 63 trading days

$$S_1 Long = 1\; if:$$

$$P_t < Pctl_{5\%}(P_t \ldots P_{t-10})$$
$$and$$
$$P_t < Pctl_{15\%}(P_t \ldots P_{t-21})$$
$$and$$
$$P_t < Pctl_{25\%}(P_t \ldots P_{t-63})$$

$$S_1 Long = 0\; elsewhere$$

S1_short is triggered when the price of an instrument is simultaneously above:

- the 95% percentile calculated on the last 10 trading days
- the 85% percentile calculated on the last 21 trading days
- the 75% percentile calculated on the last 63 trading days

$$S_1Short = -1\ if:$$

$$\begin{gathered} P_t > Pctl_{95\%}(P_t \dots P_{t-10}) \\ and \\ P_t > Pctl_{85\%}(P_t \dots P_{t-21}) \\ and \\ P_t > Pctl_{75\%}(P_t \dots P_{t-63}) \end{gathered}$$

$$S_1Short = 0\ elsewhere$$

S2 is an ON/OFF condition that is ON when the 20d and 60d moving averages are not breached, I interpreted this condition as a reinforcement of S1.
Hence, S2_long is triggered when the price of an instrument is below both the 20d and the 60d moving average (thus indicating a possible reversion towards the faster and the slower mean),

$$S_2Long = 1\ if:$$

$$\begin{gathered} P_t < MA_{20d} \\ and \\ P_t < MA_{60d} \end{gathered}$$

$$S_2Long = 0\ elsewhere$$

S2_short is triggered when the price of an instrument is above both the 20d and the 60d moving average.

$$S_2Short = 1\ if:$$

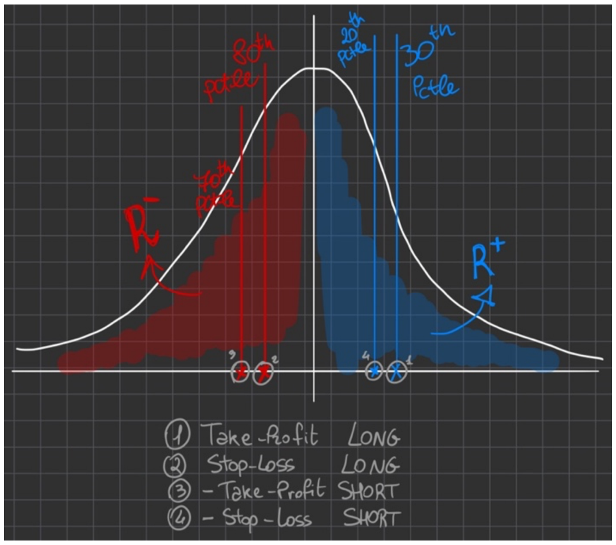
$$\begin{gathered} P_t > MA_{20d} \\ and \\ P_t > MA_{60d} \end{gathered}$$

$$S_2Short = 0\ elsewhere$$

The entry signal, $direction = \{Entry_{Short}, Entry_{Longt}\} = \{-1, 1\}$, is given by the combination of the two signals:

$$\begin{gathered} Entry_{Long} = S_1Long \times S_2Long \\ Entry_{Short} = S_1Short \times S_2Short \end{gathered}$$

Once a trade is open, Take Profit (TP) and Stop Loss (SL) returns are calculated using monthly returns. Starting from prices, the monthly return is defined as $R_{1M} = \frac{P_t}{P_{t-21}} - 1$ and $R^+{}_{1M} = R_{1M}|R > 0$, $R^-{}_{1M} = R_{1M}|R < 0$.
TP are defined as supportive 30th percentile move, so for long position $TP_{long} = Pctle_{30\%}(R^+_{1M})$, and for short position $TP_{short} = -Pctle_{70\%}(R^-_{1M})$.
SL are defined as counter 20th percentile move, so for long position $SL_{long} = Pctle_{80\%}(R^-_{1M})$, and for short position $SL_{short} = -Pctle_{20\%}(R^+_{1M})$.
If neither TP nor SL are hitted, the trade is automatically closed after 21 trading days at price P_{t+21}.



TP and SL computation visually explained

<u>TP / SL rule diagram</u>

- according to direction of the trade consider $R^+{}_{1M}$ or $R^-{}_{1M}$
- use an expanding window to compute monthly returns distribution (the window is expanding daily, so to have more and more observations when estimating TP/SL)
- if direction = 1 (buy):
    - take positive Monthly returns, calculate 30th pctl → TP
    - take negative Monthly returns, calculate 80th pctl → SL (SL more conservative than TP, let profits run)
- if direction = -1 (sell):
    - take negative Monthly returns, calculate 70th pctl → TP
    - take positive Monthly returns, calculate 20th pctl → SL (SL more conservative than TP, let profits run)
- if no TP/SL hitted, exit after 21 trading days.

For simplicity, returns are computed as:

$$return_{trade} = \left(\frac{P_{exit}}{P_{entry}} - 1\right) \times direction$$

where, $P_{exit} = price\ at\ which\ trade\ is\ closed$ and $P_{entry} = price\ at\ which\ trade\ is\ open$. This computation is clearly a simplification of reality, especially because a short position return does not involve any margin requirements.
In terms of trades sizing, since all trades must lose the same maximum amount of money, and the direction being $direction = \{-1, 1\}$:
If $direction = 1$: $DolVol_{received} - DolVol_{given} = qt \times P_{SL} - qt \times P_0 = -MaxDollarLoss$
If $direction = -1$: $DolVol_{received} - DolVol_{given} = qt \times P_0 - qt \times P_{SL} = -MaxDollarLoss$
Rewriting, given the SL Price: $P_{SL} = P_0 \times (1 + SL \times direction)$, the optimal quantity for each trade is:

$$qt = \frac{MaxDollarLoss}{direction \times (P_0 - P_{SL})}$$

For each long/short entry signal, the exit condition is computed as well as the exit pice P_{exit}. Then, having computed them it is possible to compute: the trade duration in days; the annualized volatility $Annual\ Volatility = \sqrt{252 \times Var(returns)}$ where $returns$ is the vector of daily returns of the prices between $date_{entry} = date\ of\ entry\ trade$ and $date_{exit} = date\ of\ exit\ trade$; the trade return; the avg. daily return $daily\ return = \frac{return_{trade}}{trade\ duration}$ and the 1Y Sharpe ratio of the trade defined as $Sharpe\ Ratio = \frac{daily\ return \times 252}{Annual.\ Volatility}$.
Finally, it is possible to compute the position dollar value at entry $PositionDollarValue = qt \times (P_{entry} + P_{exit})/2$, which is a proxy of how many dollars of the portfolio are invested at each trade; and the trades' weights defined as $weight_i = \frac{duration_i \times PositionDollarValue_i}{\sum_i duration_i \times PositionDollarValue_i}$, where $i = 1 \ldots NumberOfTrades$; and $Portfolio\ Sharpe\ Ratio = \sum_i Sharpe\ Ratio_i \times weight_i$.
The method used to calculate the weights is undoubtedly a simplification of the actual influence each trade exerts on the overall portfolio Sharpe ratio. However, it proves effective in assigning weights to each trade in proportion to its size and duration. This, in turn, allows for the computation of a weighted Sharpe ratio for the portfolio, where a trade with a larger size or longer duration carries more weight.

Strategy backtest:

In this paragraph, the main statistics of the backtested strategy for the period from 12th February 2014 to 20th January 2024 for the twenty series are presented. In total 11052 trades

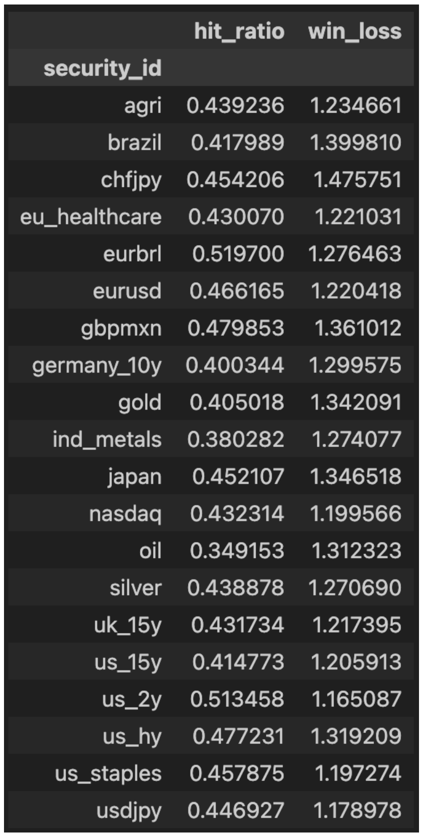
are executed with the rules outlined before and a max dollar loss per trade set at \$1000 (which is the 0.4% max loss per trade, having AUM of \$250k).
When executing trades, I operate under the assumption that upon the generation of a buy/sell signal, there is an immediate entry into the trade with no time lag in execution. I consider this assumption to be realistic, as it is improbable for a trader to wait an entire trading day before acting on a signal. I presume that the signal is generated at time t, with a price closely resembling the closing price, and the trade is promptly executed at the closing price at time t.
For each instrument, the hit ratio and win loss are computed. Hit ratio is defined as:

$$Hit\ Ratio_s = \frac{card(trades_s^+)}{card(trades_s)}$$

where $trades_s^+$ are the trades with positive returns for instrument s, and $card(A)$ is the cardinality of set A. The hit ratio measures the share of trades with positive returns over the total number of trades. Win loss is defined as:

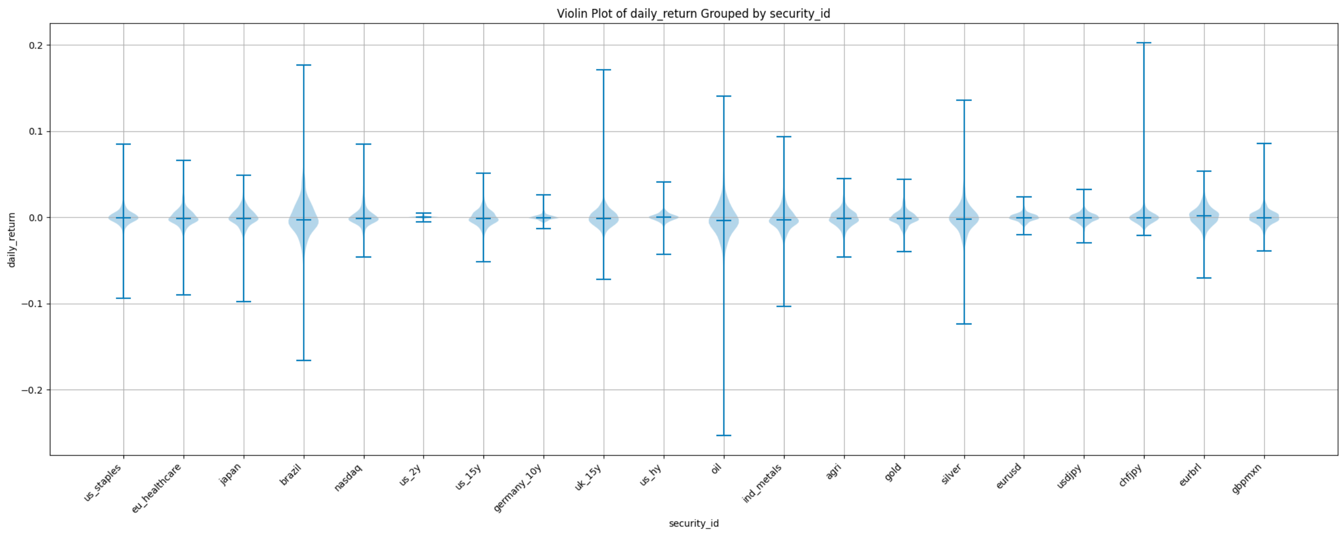
$$WinLoss_s = \frac{avg(return_{trades_s}^+)}{avg(|returns_{trades_s}^-|)}$$

which is basically the ratio of the mean return of positive trades over the mean of the absolute value of the negative trades for instrument s.
The hit ratio of the strategy is often below 0.5, with the exception of *eurbrl* and *us_2y*; the win loss ratio is always greater than 1. This means that positive return trades deliver higher return than negative return trades, but negative return trades are more frequent.

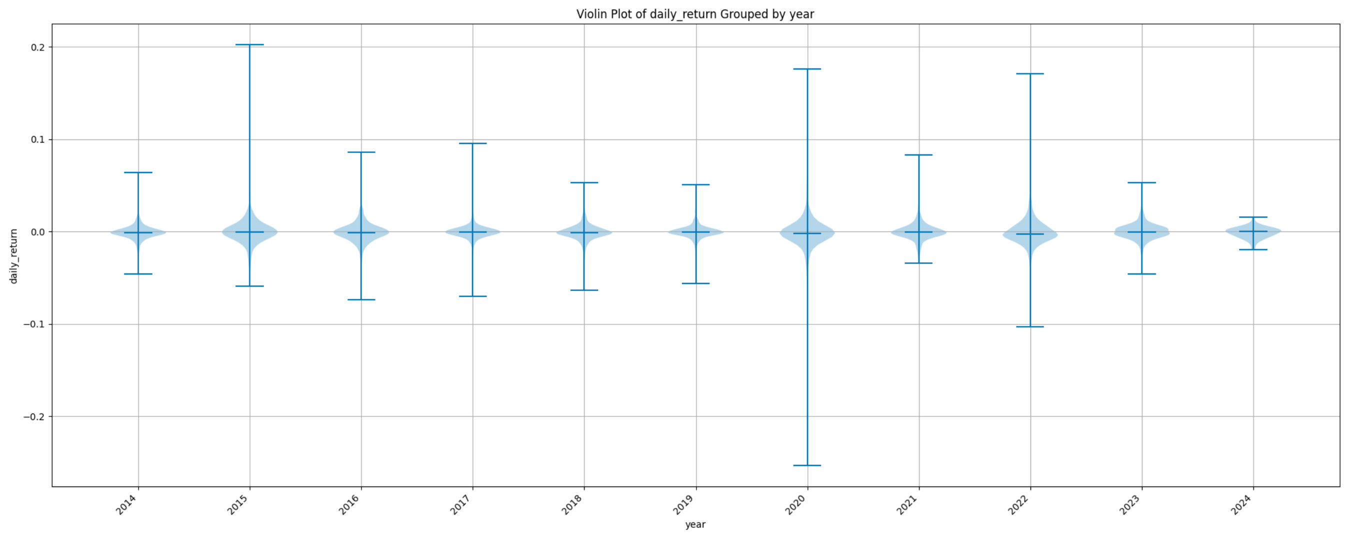


security_id	hit_ratio	win_loss
agri	0.439236	1.234661
brazil	0.417989	1.399810
chfjpy	0.454206	1.475751
eu_healthcare	0.430070	1.221031
eurbrl	0.519700	1.276463
eurusd	0.466165	1.220418
gbpmxn	0.479853	1.361012
germany_10y	0.400344	1.299575
gold	0.405018	1.342091
ind_metals	0.380282	1.274077
japan	0.452107	1.346518
nasdaq	0.432314	1.199566
oil	0.349153	1.312323
silver	0.438878	1.270690
uk_15y	0.431734	1.217395
us_15y	0.414773	1.205913
us_2y	0.513458	1.165087
us_hy	0.477231	1.319209
us_staples	0.457875	1.197274
usdjpy	0.446927	1.178978

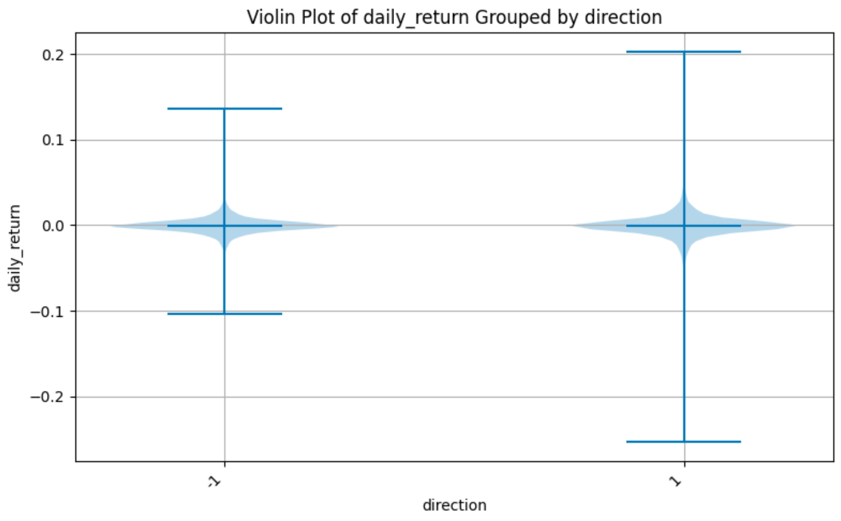
The plot below shows the distribution of trades' returns by security. *Nasdaq*, *uk_15y*, *chfjpy*, *gbpmxn* exhibit trades' returns skewed towards positive returns, while *eu_healthcare*, *japan*, *oil*, *eurbrl* towards negative returns.



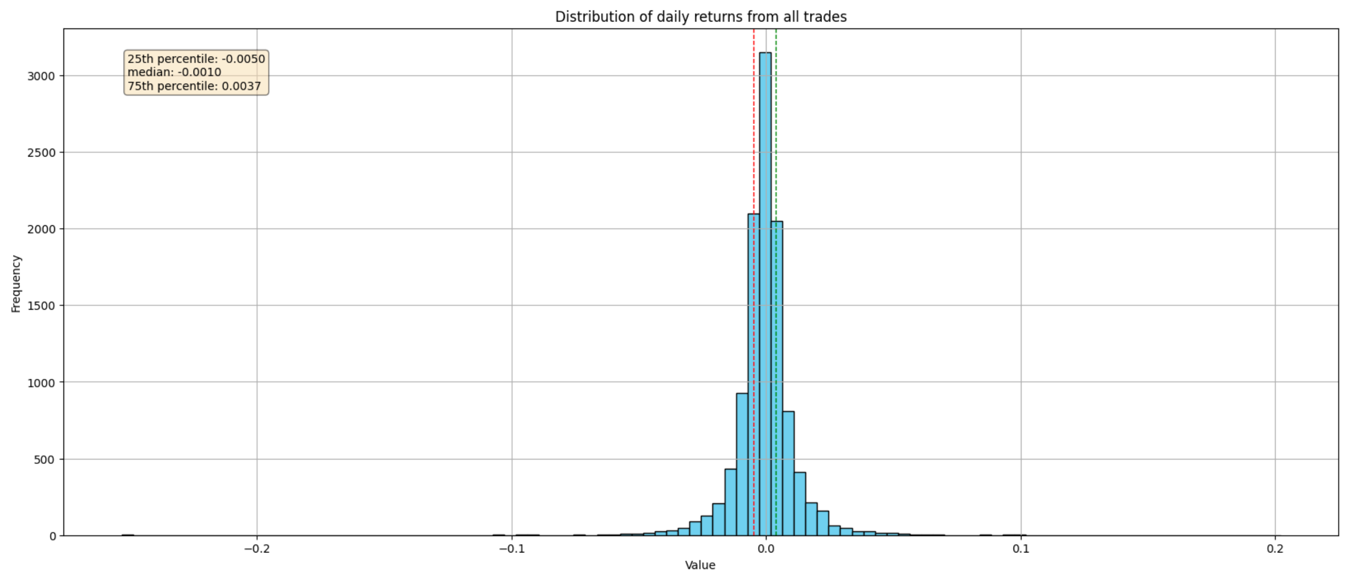
Trades performed in 2015, 2017, 2021, 2022 exhibit positive skew, while trades done in 2020 exhibit negative skew.



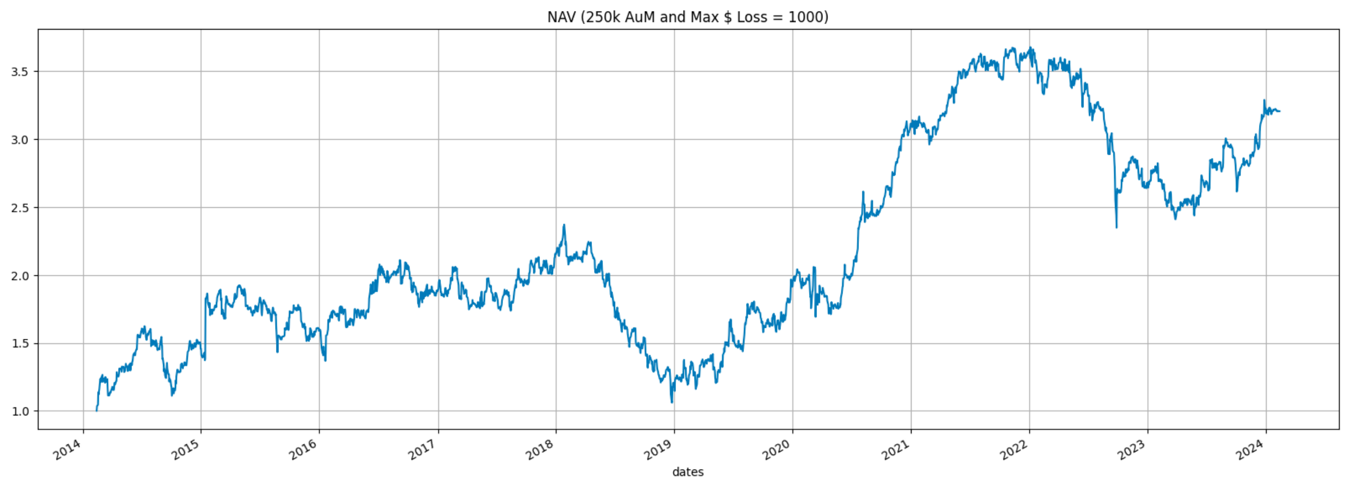
Short signal seems more effective than long signal in capturing the mean reversal potential as visible from the following chart

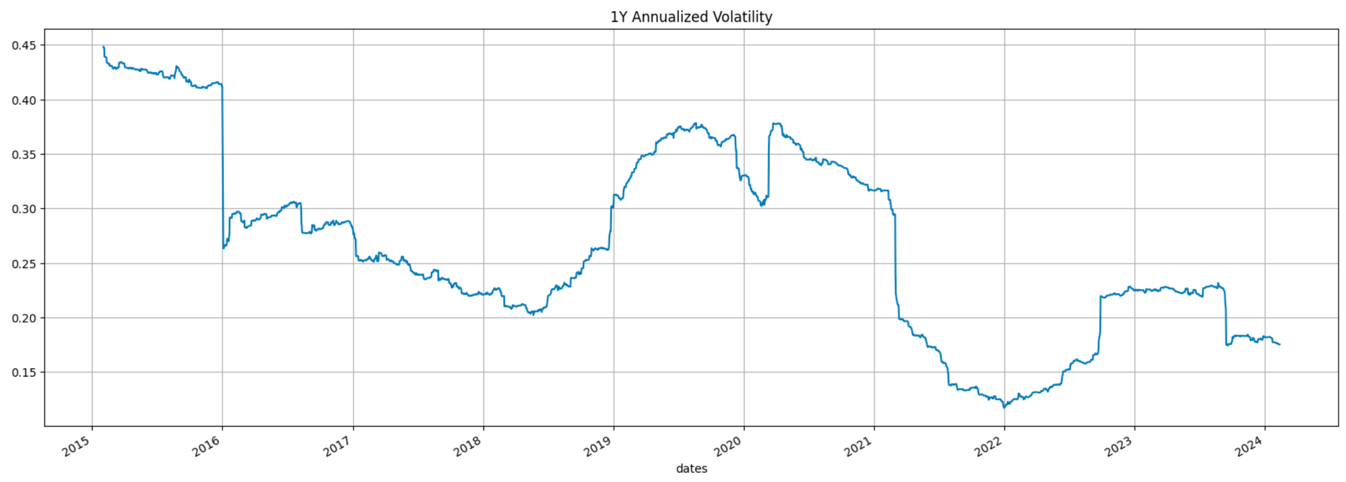


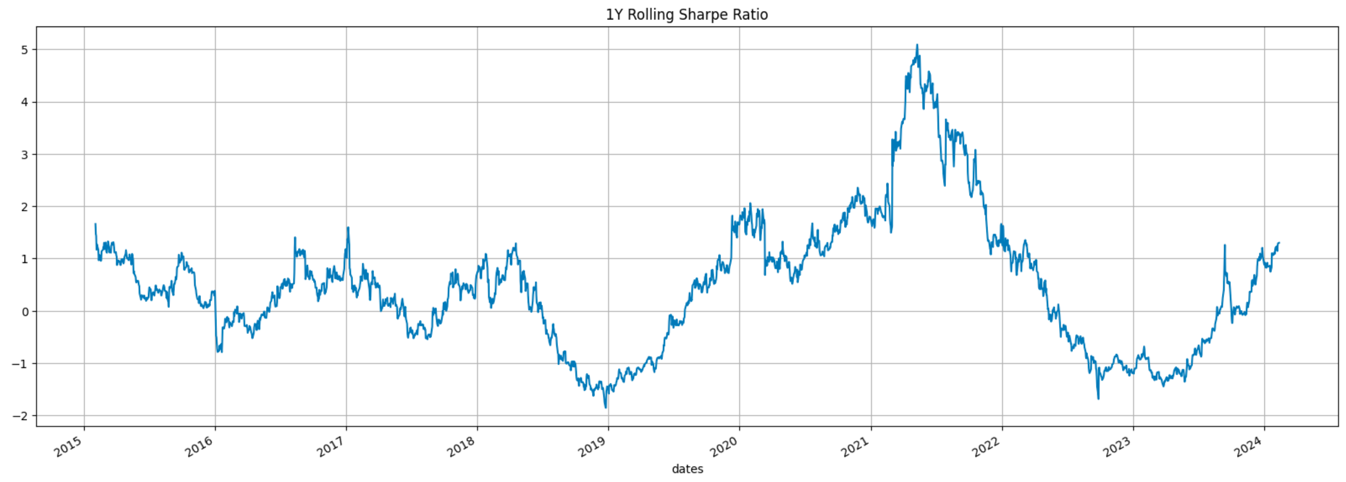
This is the complete distribution of all trades across all securities in the full period, the median daily return is slightly negative and the 25th and 75th percentiles are shown by the red dotted line and the green dotted line respectively. The 5th percentile (that can be interpreted as simple historical VaR) is equal to -1,7%.



Below the NAV of 250k invested and max dollar loss set at 1000. The NAV calculation considers each trade independently and in each trading day the portion of AUM not invested is mantained in cash. Each position is opened and closed independently from other simultaneously open positions in the same or other assets; this means that two or more simultaneous long or short positions on the same instrument can be open, if the signals are triggered in closeby dates.







The Portfolio Sharpe Ratio of the strategy, calculated as explained before, which is the metric that we aim at maximizing, is equal to 0.83.

Strategy improvement:

To improve the proposed strategy I employed a grid search on parameters that lead to the generation of S1_Long, S1_Short, S2_long and S2_Short. I explored all the possible combination of the following parameters:

- Long entry S1:

id	*Percentile*	*# days*
1.	10%, 20%, 30%	10, 21, 63
2.	10%, 20%, 30%	5, 10, 15
3.	10%, 20%, 30%	20, 40, 70

- Short entry S1:

id	*Percentile*	*# days*
1.	90%, 80%, 70%	10, 21, 63
2.	90%, 80%, 70%	5, 10, 15
3.	90%, 80%, 70%	20, 40, 70

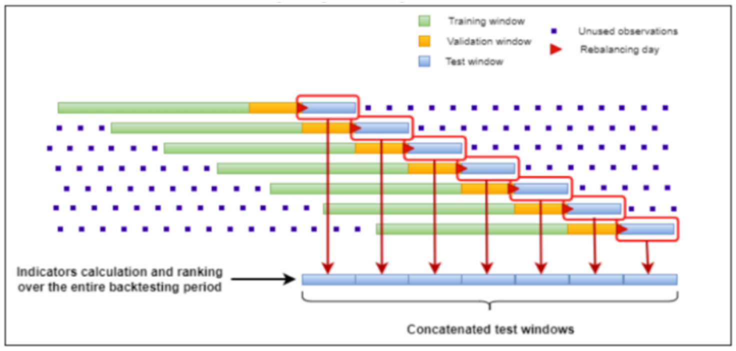
- Fast Moving Average:

id	*# days*
1.	10
2.	20
3.	30
4.	40

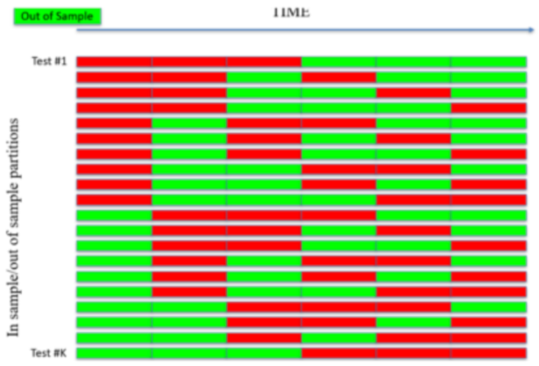
- Slow Moving Average:

id	*# days*
1.	50
2.	65
3.	80
4.	90

For each iteration of the grid search a different *id* from each table is selected and the strategy's Portfolio Sharpe Ratio is computed.
This method proves beneficial when adjusting the parameters of a strategy based on historical data, but it carries a substantial risk of backtest overfitting due to the non-ergodic nature of return series. The issue lies in refining and selecting an improved strategy solely based on backtest results, which should not serve as a research instrument. The strategies are tested on one of the countless potential realizations of the stochastic process governing prices, rather than the actual data-generating process itself. This approach can lead to a multiple testing problem, increasing the likelihood of backtest overfitting.
A more prudent approach, especially having longer timeseries of prices, involves tuning strategy parameters on a Training set, validating them on a Validation set, and keeping a separate Testing set of unknown data to assess the strategy's true performance once optimal parameters are identified. Nonetheless, this slightly more advanced approach requires consideration of potential different regimes in the three sets; therefore, employing a Walk Forward Out-of-Sample (OOS) or, even more effectively, a Randomized OOS is recommended.
An even more sophisticated option might involve generating synthetic data using Monte Carlo Methods or Kernel Density Estimators. These procedures, because of their complexity, are beyond the scope of the current exercise, however it is crucial to recognize the limitations of the process in question.

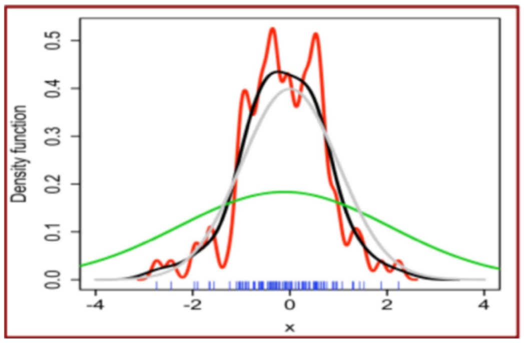

Training window
Validation window
Test window
Unused observations
Rebalancing day
Indicators calculation and ranking over the entire backtesting period
Concatenated test windows
Walk forward OOS

The strategy is tuned incrementally as new data arrives.



Randomized OOS

Strategy's parameter are tuned (Train+Valid) in the red timesteps and Tested in the green timesteps. The process is repeated in the K partitions of the dataset, and results are averaged cross-sectionally.



Kernel Density Estimation

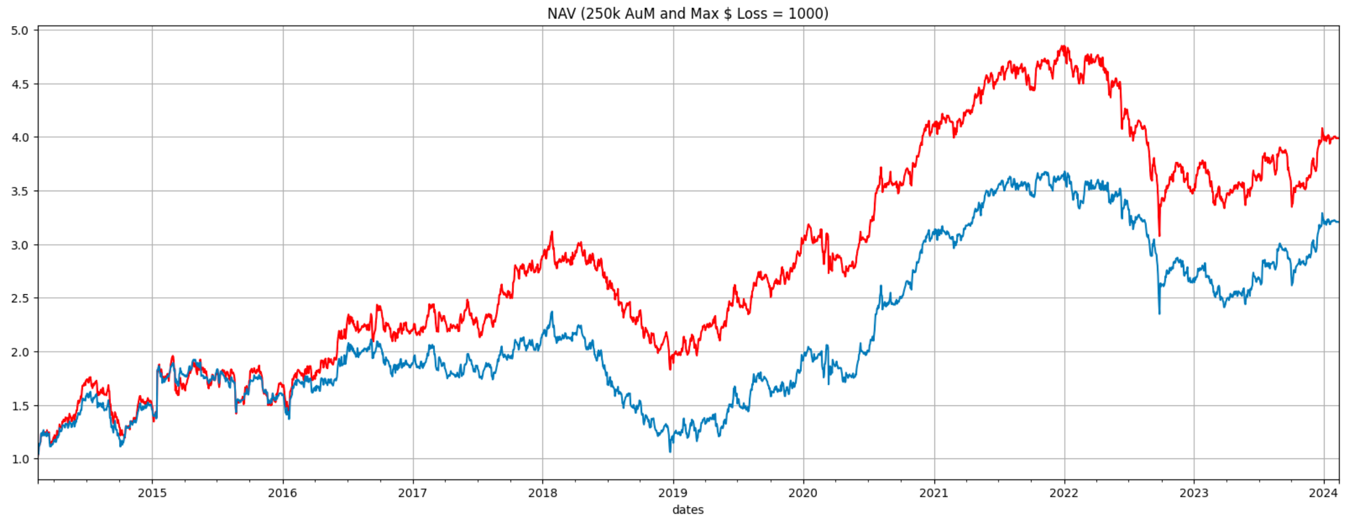
This method uses KDE to estimate the probability density function of a random variable based on kernel functions. It can be applied to the case of securities' returns to generate synthetic data for more robust backtesting processes.

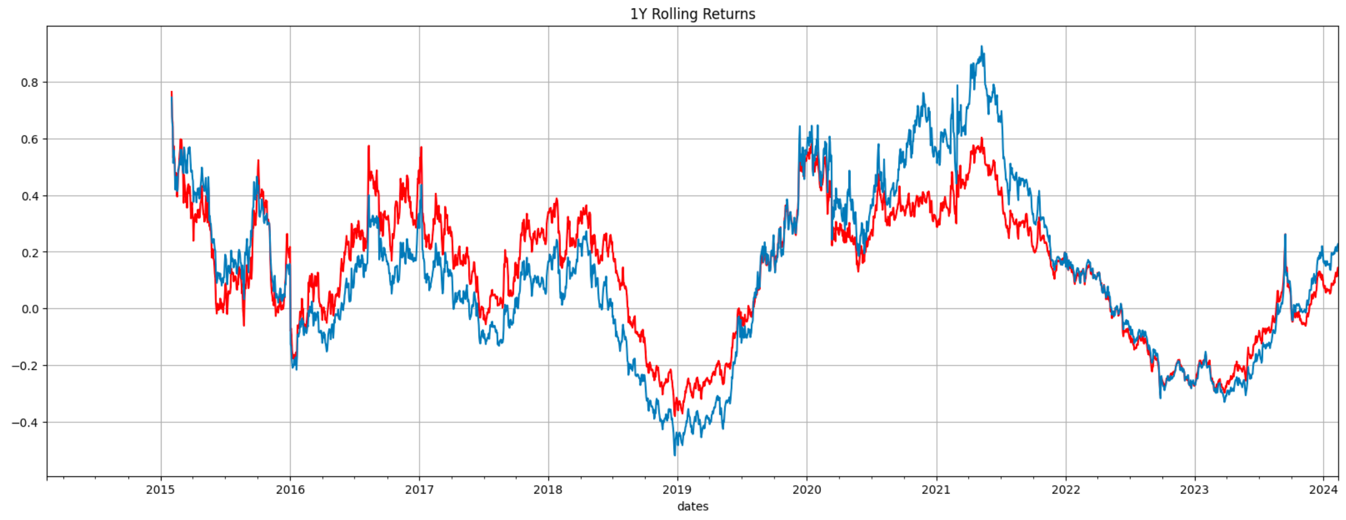
The parameters that maximise the Portfolio Sharpe Ratio are:

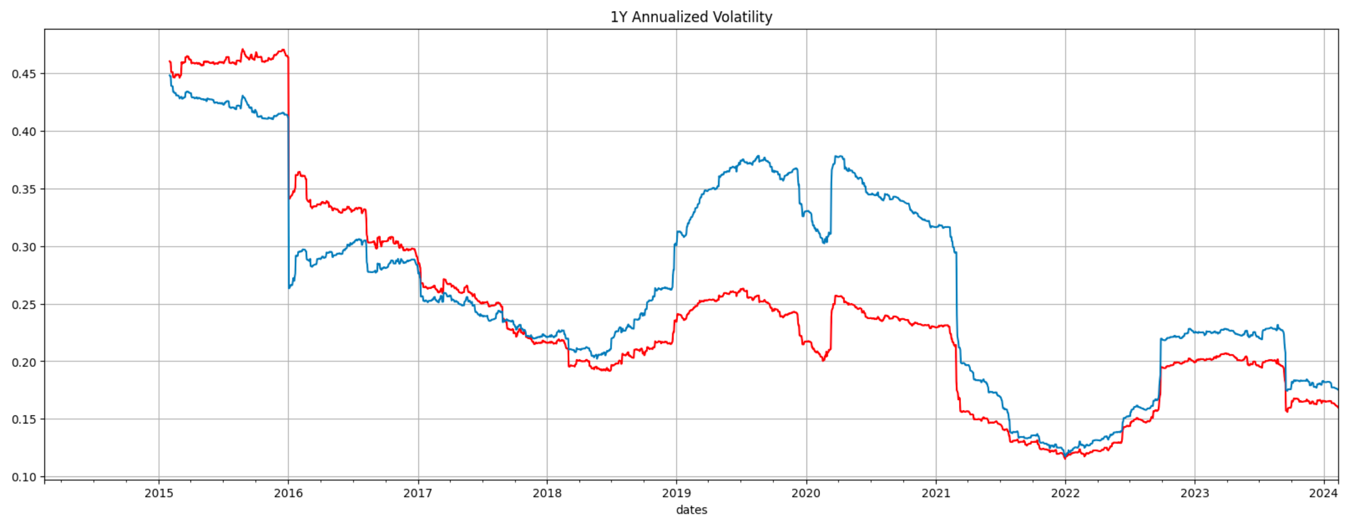
- Long entry S1: Percentiles = 10%, 20%, 30% respectively associated to # days = 5, 10, 15;
- Short entry S1: Percentiles = 90%, 80%, 70% respectively associated to # days = 10, 21, 63;

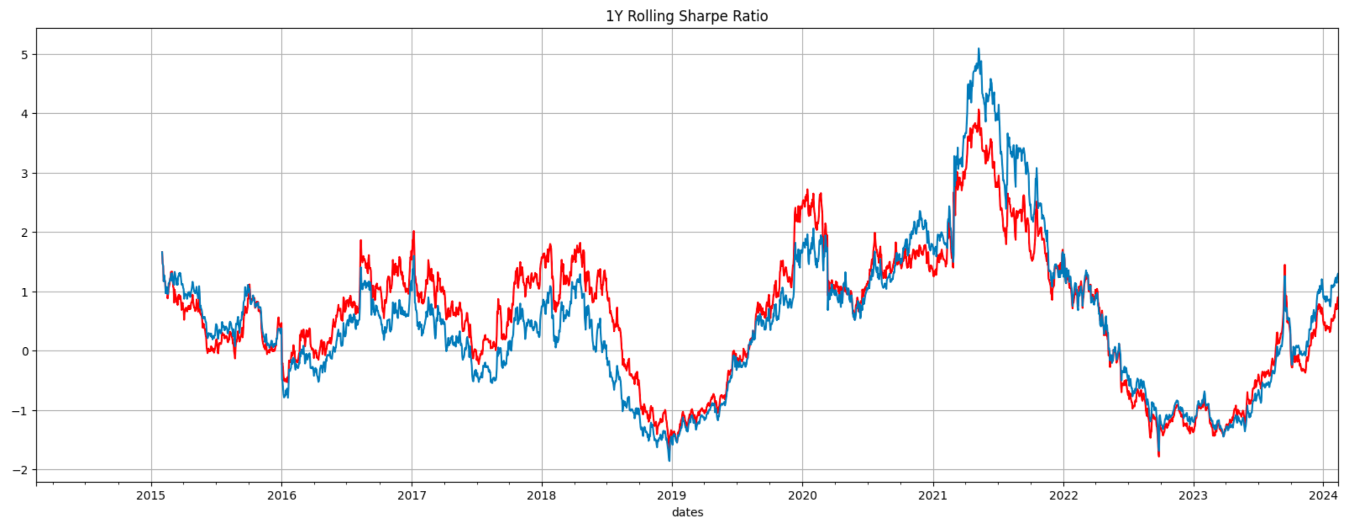
- Fast Moving Average = 40;
- Slow Moving Average = 50.

obtaining 13828 trades in the period, and the following NAV (in blue the old strategy, in red the proposed alternative):









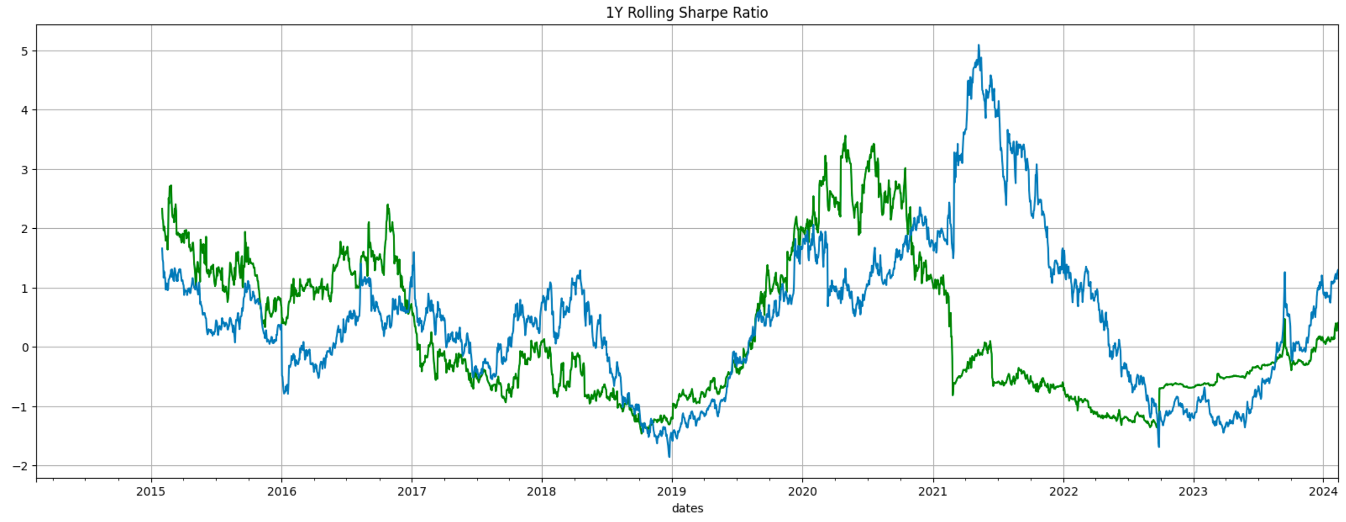
The improved strategy has higher returns in the period from 2016 to 2019 and significantly lower volatility from 2018 to 2021 and in 2023, resulting in a consistently higher or at most similar 1Y Sharpe ratio with the sole exception of 2021. The Portfolio Sharpe Ratio of the improved strategy is 1.05 versus 0.83 for the original strategy.

Impact of volatility:

As evident in the violin plot illustrating the yearly distribution of daily returns, the downside volatility of the bets remains relatively constant over the years, except for deviations in 2020 and 2022. In order to manage bets' volatility and to incorporate the historical volatility into the position sizing, I introduced a parameter $\gamma = 5\%$ that transforms the optimal quantity for each trades:

$$qt' = \frac{MaxDollarLoss \times \frac{\gamma}{3M\ Volatility}}{direction \times (P_0 - P_{SL})}$$

In such way, the optimal the maximum dollar loss for each trade is rescaled by the term $\frac{\gamma}{3M\ Volatility}$ which is dependent on the instrument's historical 3 months volatility. The higher this volatility, the lower the maximum dollar loss allowed for the trade. The parameter γ can be considered as a target volatility, i.e. if $3M\ Volatility = \gamma$ the ratio will be 1; otherwise if $3M\ Volatility > \gamma$ the ratio will be lower than 1, thus reducing the $MaxDollarLoss$ and qt' allowed for the trade, while the contrary is true if $3M\ Volatility < \gamma$.
In green the 1Y Rolling Sharpe Ratio of the strategy with the inclusion of gamma parameter.
The strategy performs very well in the years from 2015 to 2017 and in 2020, while underperforms from 2017 to 2019 and from 2021 to mid-2022.



In this context, the new strategy seems to complement the strategy without the inclusion of the gamma parameter, as each excels when the other falls behind. One potential refinement of this approach could involve a model that allocates to either strategy based on their relative momentum.
As a concluding note, an alternative method to integrate market regimes into the model – beyond the scope of this project – might involve employing an unsupervised model (such as K-means, Self Organizing Maps, Hierarchical Clustering or DBSCAN) on a collection of macro variables considered significant for capturing regimes. These variables could include real interest rates, term spread, credit spread, breakeven inflation, VIX, Put-Call ratio, commodity prices, market-implied volatility in rates, and other leading and coincident indicators. Subsequently, fine-tuning the strategy's parameters could be executed differently based on the identified market regime.

OPTIMAL SIZING PROBLEM

Function structure:

The objective of this task is to develop an algorithm capable of determining one of the specified variables based on the information provided by the others: **Number of trades per year**, **Magnitude of sigma stop**, **Percentage of assets under management (%AUM) lost per trade**, and **Annual volatility**.
To do so, I created a function called `optimal_sizing()`:

```
optimal_sizing(df,
               incipit_date = '2021-04-02',
               number_of_trades = None,
               ptf_vol_tgt = 0.15,
               aum_lost_sl = 0.02,
               sigma_sl = -1,
               sigma_tp = 1.5,
               max_trade_duration = 63,
               iterations = 100)
```

the variable calculated from the function is the one initialized with the value `None` (in this case `number_of_trades`).
NB: only `number_of_trades`, `ptf_vol_tgt`, `aum_lost_sl`, `sigma_sl` can be initialized with value `None`.
The other parameters that must be necessarily specified are:

- `df`: the dataframe with the prices for the 20 series;
- `incipit_date`: the date of the simulation. In this case it is as if we are running the simulation on the 1st of April 2022. The date is needed to compute the historical volatility and to simulate trades. By running the function with a different incipit date will therefore lead to a different result;
- `ptf_vol_tgt`: the portfolio annualized volatility target;
- `aum_lost_sl`: the share of AUM lost when a trade hits the Stop-loss;
- `sigma_sl`: the sigma used to set the Stop-loss.

These other parameters are optional and set to some default values:

- `sigma_tp`: the sigma used to set the Take-profit. Default is set to +1.5;
- `max_trade_duration`: the maximum allowed length of the trades measured in trading days. Default is set to 63 (3 months);
- `iterations`: the number of iterations used by the simulation to find the optimal parameter. For each run, the function outputs the average solution +/- the standard

deviation computed across the selected number of iterations. Default is set to 100, decreasing this number leads to faster computation but with lower precision.

This is an example of output:

```
Optimal Number of Trades: 34 ± 7 [with avg. portfolio volatility of 15.28%]
```

over 100 simulations the average number of trades needed to obtain a portfolio with 15% volatility, is 34 +/- 7. NB: as this approach is based on Monte Carlo simulations, running the function with the same parameters twice can lead to slightly different results; however the obtained solutions remain consistent across different simulations, if the `iterations` parameter is big enough.

If one wants to know what is the ideal value of sigma for the Stop-loss with 34 trades per year in input, he can run:

```
optimal_sizing(df,
               incipit_date = '2021-04-02',
               number_of_trades = 34,
               ptf_vol_tgt = 0.15,
               aum_lost_sl = 0.02,
               sigma_sl = None,
               sigma_tp = 1.5,
               max_trade_duration = 63,
               iterations = 100)
```

obtaining:

```
Optimal Sigma for SL: -1 ± 0 [with avg. portfolio volatility of 15.0%]
```

which is coherent with what specified in the first run of the algorithm, when searching for the optimal number of trades.

Once again, one can calculate the %AUM lost when a trade is stopped:

```
optimal_sizing(df,
               incipit_date = '2021-04-02',
               number_of_trades = 34,
               ptf_vol_tgt = 0.15,
               aum_lost_sl = None,
               sigma_sl = -1,
               sigma_tp = 1.5,
               max_trade_duration = 63,
               iterations = 100)
```

obtaining:

```
Estimated %AUM lost per trade: 1.88% ± 0.17%
```

And finally one can estimate the portfolio target volatility:

```
optimal_sizing(df,
                incipit_date = '2021-04-02',
                number_of_trades = 34,
                ptf_vol_tgt = None,
                aum_lost_sl = 0.02,
                sigma_sl = -1,
                sigma_tp = 1.5,
                max_trade_duration = 63,
                iterations = 100)
```

obtaining:

```
Estimated Portfolio Volatility: 15.13% ± 2.27%
```

Algorithm explanation:

The following are the main conceptual steps that the function performs when searching for a solution. A single iteration can be summarised as:

- Create a list of dates for simulated trades within a one-year period starting from `incipit_date`.
- While the portfolio volatility is below the target, enter a loop generating random trades;
- For each generated random trade, check if Take-profit or Stop-loss conditions are met;
- Compute the portfolio NAV and calculate the portfolio volatility.

To improve the robustness of the simulation, these operations are repeated a number of times that can be set changing the `iterations` value.

In this way the algorithm converges to the optimal solution which is the average of the different solutions found in each iteration.

Appropriate number of trades in a year:

The appropriate number of trades in a year with sizing based on:

- `ptf_vol_tgt`: 15%;

- `aum_lost_sl`: 2%;
- `sigma_sl`: -1.
- `sigma_tp`: +1.5;
- `max_trade_duration`: 63.

is equal to 34 trades circa. This number has been estimated by running the function once per year and averaging the obtained results.