# **6.**

# **Trend-Pullback Strategies**

In the previous chapters, we have seen that the financial market moves in trends and that every trend eventually ends, leading to a market reversal. We have then written the respective algorithms suitable for these two aspects of the market separately: trend in chapter 3-4 and reversal in chapter 5. Anyway, in reality, the two phenomena often occur simultaneously, giving rise to the typical market pattern in waves.

The objective of this chapter is to understand the trend-pullback dynamics, how to build some algorithms components suitable for these market phases and how to use Tradestation and Excel to assembly such components in order to give life to multiple appROAches to afford the trend-pullback market behavior.

This chapter we are going to delve into:

* Market rationale behind a Trend Pullback Strategy
* Writing Trend-Pullback components
* Assembling Trend-Pullback components
* Conducting Sensitivity Analysis
* Conducting Out of Sample Analysis

NOTE: For educational purposes, we will treat both long and short algorithms equally, including both long and short versions in all codes. However, in reality, the use of trend pullback strategies is only employed in the long side since the stock market, for structural reasons, tends to grow. For other financial assets such as commodities and forex that don’t have long bias, as we will see in the upcoming chapters, we will use different techniques.

**Market rationale behind a Trend-Pullback strategy**

In order to understand a trend-pullback logic let’s start with a couple of definition :

* pullback up
* pullback down.

A **pullback up** is a temporary decline in the price of a stock or the broader stock market after a period of upward movement. It's essentially a short-term reversal in the prevailing trend. Pullbacks are often characterized by a decrease in prices over a relatively short period, typically ranging from a few days to a few weeks, before the upward trend resumes. (Figure 6.1)

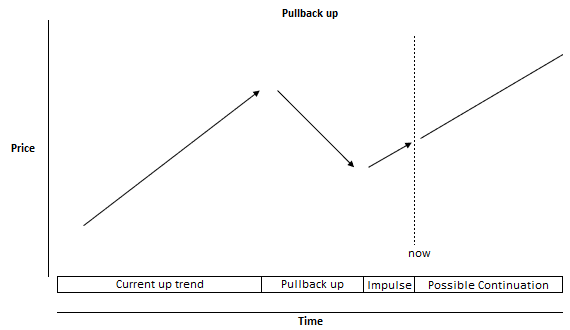


Figure 6.1 Pullback up

In Figure 6.1 you can see a typical price behavior during up trends

A **pullback down** is a temporary upward movement in the price of a stock or the broader market within the context of an overall downward trend. It's essentially a brief interruption in the downward movement before the prevailing downtrend resumes.

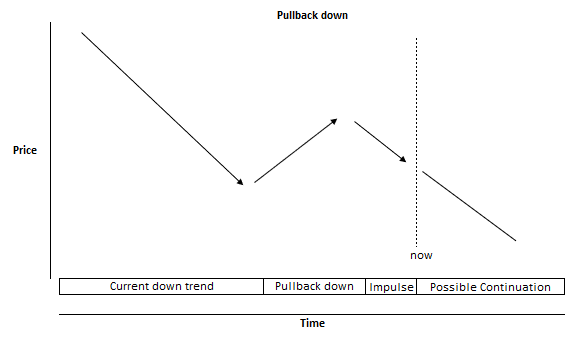


Figure 6.2 Pullback down

In Figure 6.2 you can see a typical price behavior during down trends.

Pullbacks are indeed a product of the interactions among market participants with diverse time horizons and objectives. Long-term investors often view pullbacks as opportunities to accumulate assets at lower prices, leveraging their focus on fundamental value and their willingness to hold positions over extended periods. In contrast, short-term traders may seize pullbacks as occasions to realize smaller gains or cut losses, reflecting their focus on short-term price movements and technical indicators.

Understanding these distinct motivations and behaviors is crucial for interpreting market trends and effectively navigating through pullbacks. By recognizing how different types of participants respond to market movements, investors can better anticipate price dynamics, adjust their strategies accordingly, and capitalize on opportunities presented by pullbacks. This awareness enables them to manage risk more effectively and align their actions with their investment objectives, whether they prioritize long-term growth, short-term gains, or risk mitigation.

Let’s now move onto the development side.

**Writing trend-pullback components**

An algorithm to identify a trend-pullback pattern consists essentially of 3 components:

1. **an existing trend**
2. **a pullback**
3. **an impulse in the trend direction**

**Existing trend**

In order to catch an existing trend, according to what we presented in Chapter 3, we know that we can use simple moving averages, but there are other algorithms, a bit more elaborated, to identify a trend: let’s delve into a couple of them:

* **Supertrend**
* **ADX**

**Supertrend**

The Supertrend indicator is typically based on the volatility of the asset's price. It consists of two main components: a moving average and a multiplier.

A moving average: The Supertrend indicator uses a moving average of the asset's price to determine the overall direction of the trend. This moving average can be calculated using different periods, such as 7 days, 14 days, or 50 days, depending on the trader's preference and the timeframe they are trading.

A multiplier: The Supertrend indicator also incorporates a multiplier that is usually based on the asset's volatility. This multiplier is multiplied by the average true range (ATR) of the asset's price to adjust the distance of the indicator from the price.

The resulting Supertrend line is plotted above or below the asset's price chart, depending on whether the trend is bullish or bearish. Traders typically use the crossover of the price and the Supertrend line as potential entry or exit signals. When the price crosses above the Supertrend line, it may indicate a bullish trend, while a crossover below the Supertrend line may suggest a bearish trend.

We can write the Supertrend indicator as follows:

inputs:Price(close),Supertrend\_Len( 22), Supertrend\_Multiplier( 20);

variables: Volty( 0 ), Supertrend\_( 0 );

Volty = Supertrend\_Multiplier \* AvgTrueRange( Supertrend\_Len )[1];

if Price > Supertrend\_[1] and Price[1] > Supertrend\_[2] then

Supertrend\_ = MaxList( Supertrend\_[1], Price - Volty )

else if Price < Supertrend\_[1] and Price[1] < Supertrend\_[2]

then

Supertrend\_ = MinList( Supertrend\_[1], Price + Volty )

else if Price > Supertrend\_[1] then

Supertrend\_ = Price - Volty

else

Supertrend\_ = Price + Volty ;

plot1(Supertrend\_,"Supetrend",green);if close > Supertrend\_ then setplotcolor(1,red);

In Figure 6.3 you can see the Supertrend indicator applied to a chart.

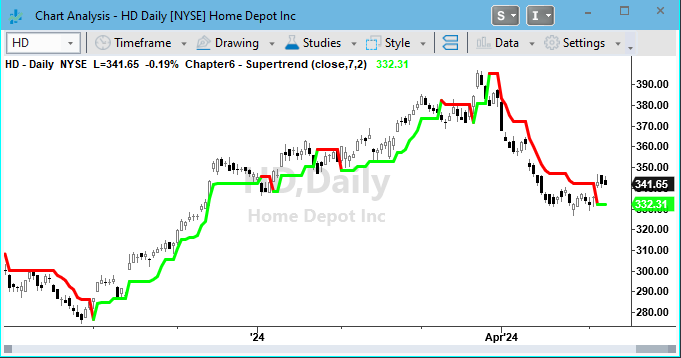


Figure 6.3 Supertrend Indicator

In Figure 6.3 you can see that when the Supertrend indicator is below the price, it colors green, indicating a bullish trend. Conversely, when the Supertrend indicator is above the price, it colors red, indicating a bearish trend,

**ADX**

Another useful indicator the detect a trend is ADX. ADX stands for Average Directional Index. It's a technical indicator used to measure the strength of a trend in a financial instrument. ADX has been developed by J. Welles Wilder in the late 1970s.

ADX can be interpretated as follows:

* ADX values below a low trigger suggests a weak or non-existent trend.
* ADX values above a high trigger suggests a strong trend.

As you see ADX provides a trend strength evaluation, but it tells nothing about direction. To detect trend directions we need use some additional information. ADX is generally used paired with +DI (Positive Directional Indicator) and -DI (Negative Directional Indicator) .

Here's a breakdown of such indicators:

* +DI (Positive Directional Indicator) measures the strength of the upward movement in price. It's derived from comparing Highs.
* -DI (Negative Directional Indicator): Conversely, the -DI line measures the strength of the downward movement in price. It's derived from comparing Lows

We can write the ADX indicator as follows:

inputs: ADXLength( 14 ),TriggerLevel(40) ;

variables: double ADXValue( 0 );

ADXValue = ADX( ADXLength );

Plot1( ADXValue, "ADX" ,red);

if ADXValue>TriggerLevel then setplotcolor(1,blue);

Plot2( TriggerLevel, !( "TrigLevel" ) );

Where ADX is the Tradestation built-in function which requires only a length as input

We can write the +DI -DI indicator as follows:

inputs: Length( 14 ) ;

variables: oDMIPlus( 0 ), oDMIMinus( 0 ), oDMI( 0 ), oADX( 0 ),

oADXR( 0 ), oVolty( 0 ) ;

Value1 = DirMovement( H, L, C, Length, oDMIPlus, oDMIMinus, oDMI, oADX, oADXR,

oVolty ) ;

plot1(odMIPlus,"+DI",green);

plot2(odMIMinus,"-DI",red);

Where DirMovement is the Tradestation built-in function which requires 1 length as inputs and provides both odMIPlus (the +DI) and odMIMinus (the -DI)

In Figure 6.4 you can see the Adx indicator applied to a chart

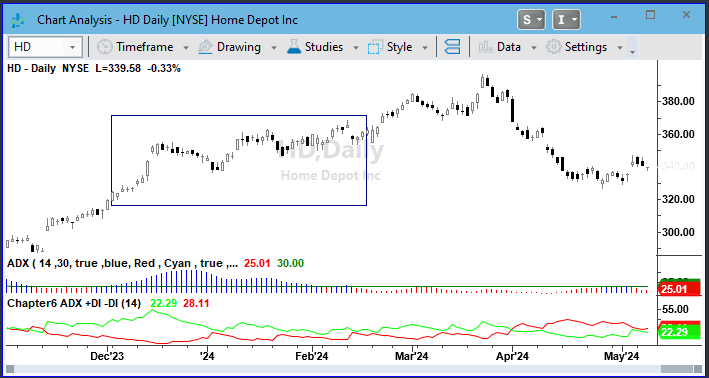


Figure 6.4 ADX

In subchart 1 of Figure 6.4 the Average Directional Index (ADX) appears as a histogram, with a threshold set at 30. In subchart 2, two lines are visible: the Positive Directional Indicator (+DI) line, depicted in green, and the Negative Directional Indicator (-DI) line, shown in red. As the ADX surpasses 30, its color changes, indicating the presence of a trend. To determine the direction of this trend, observe the relationship between the +DI and -DI lines. When the +DI line is positioned above the -DI line, it signifies a bullish trend. Conversely, if the -DI line is above the +DI line, it indicates a bearish trend (as depicted in the rectangle).

**Pullbacks**

In order to catch pullbacks we can use several oscillators: let’s delve into a couple of them :

* **RSI**
* **Stochastic**

**RSI**

RSI stands for Relative Strength Index. It's a popular technical analysis tool used by traders to measure the magnitude of recent price changes to evaluate overbought or oversold conditions in an asset. RSI is calculated using a formula that compares the average gains and losses over a specified time period, typically 14 days. The result is a value that ranges from 0 to 100. Traditionally, an RSI above 70 indicates that an asset is overbought and may be due for a correction, while an RSI below 30 indicates that it is oversold and may be due for a bounce back.

TradeStation provides a built-in function to calculate RSI, here is the indicator’s script:

inputs:

Price( Close ), Length( 14 ) , OverSold( 30 ) , OverBought( 70 ) ,OverSColor( Cyan ) ,OverBColor( Red ) ;

variables:

RSIValue( 0 ) ;

RSIValue = RSI( Price, Length ) ;

Plot1( RSIValue, !( "RSI" ) ) ;

Plot2( OverBought, !( "OverBot" ) ) ;

Plot3( OverSold, !( "OverSld" ) ) ;

{ color criteria }

if RSIValue > OverBought then

SetPlotColor( 1, OverBColor )

else if RSIValue < OverSold then

SetPlotColor( 1, OverSColor ) ;

Where the RSI command call the in-built Tradestation function

In order to identify a pullback-up we can write the condition:

if **RSI**(close, RSILen) < oversold

conversely, in order to identify a pullback down we can write:

if **RSI**(close, RSILen) > overbought

where close and RSILen are the 2 inputs required by the function and oversold and overbought are the triggers to activate the entry conditions.

NOTE: Oversold refers to a situation where market participants may have pushed the asset price down to an excessive degree, potentially creating buying opportunities for traders looking to capitalize on a potential rebound in price.

Overbought refers to a situation where market participants may have pushed the asset price up to an excessive degree, potentially creating selling .

In Figure 6.5 you can see an RSI oscillator applied to a chat:

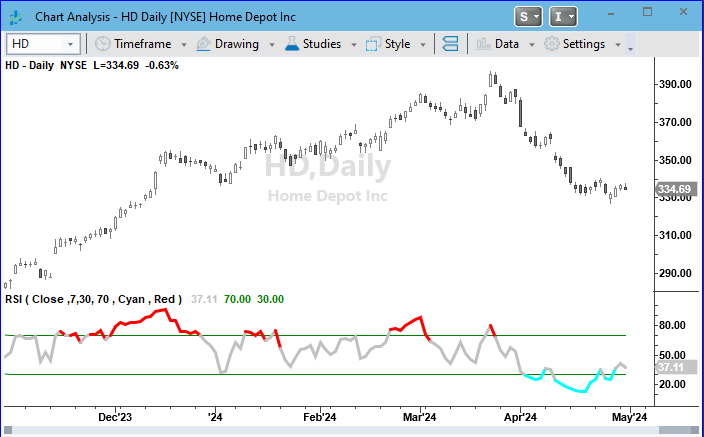


Figure 6.5 RSI

In Figure 6.5 you can see an RSI oscillator based on a 7 days length, oversold=30 and overbought=70. As you see the indicators color differently if it is above the overbought level or below the oversold level.

RSI is very useful to catch sharp movements in both directions and the oversold/overbought triggers help in easily determine how much profound the movement should be. For example, an oversold of 20 will require a deeper price down movement than an oversold of 50 and so on.

**Stochastic**

Stochastic Oscillator is one of the best algorithm to identify overbought or oversold conditions in a trading instrument. It compares the current closing price of a security to its price range over a specific period. The Stochastic Oscillator consists of two lines: %K and %D.

The %K line represents the current price relative to the price range over a specified period, typically 14 periods. The %D line is a moving average of the %K line, often a three-period moving average of %K.

Traders use the Stochastic Oscillator to identify potential reversal points in the market. When the oscillator crosses above a certain threshold (e.g., 80), it suggests that the asset may be overbought and could experience a downward correction. Conversely, when the oscillator crosses below another threshold (e.g., 20), it indicates that the asset may be oversold and could experience an upward correction.

We can write the Stochastic oscillator as follows:

inputs: PriceH( High ) , PriceL( Low ), PriceC( Close ) ,StochLength( 14 ) ,SmoothingLength1( 3 ) , SmoothingLength2( 3 ) ,SmoothingType( 1 ) , OverSold( 20 ),Overbought(80);

variables: ReturnValue( 0 ), oFastK( 0 ), oFastD( 0 ), oSlowK( 0 ), oSlowD( 0 ) ;

ReturnValue = Stochastic( PriceH, PriceL, PriceC, StochLength, SmoothingLength1, SmoothingLength2, SmoothingType, oFastK, oFastD, oSlowK, oSlowD ) ;

Plot1( oSlowK, !( "SlowK" ) ) ;

Plot2( oSlowD, !( "SlowD" ) ) ;

Plot3( OverBought, !( "OverBot" ) ) ;

Plot4( OverSold, !( "OverSld" ) ) ;

Where the command ReturnValue=Stochastic(….) calls the Tradestation in-built function.

The Stochastic function is a Multiple-Output function.

A multiple-output function has two types of parameters or "inputs" - input parameters

and input/output parameters. The values of the input parameters are passed into the

multiple-output function, but not modified by the function. The values of the input/

output parameters are passed into the multiple-output function, modified by it, and

the modified values are then inherited by - or output to - the calling routine.

For example, look at the first lines of the Stochastic function:

inputs:

PriceH ( numericseries ),

PriceL ( numericseries ),

PriceC ( numericseries ),

StochLength ( numericsimple ),

Length1 ( numericsimple ),

Length2 ( numericsimple ),

SmoothingType ( numericsimple ),

oFastK( numericref ),

oFastD( numericref ),

oSlowK( numericref ),

oSlowD( numericref );

I this example the function requires 7 inputs (numricseries) and provides 4 output (numericref).

In figure 6.6 you can see how the stochastic oscillator looks like when applied to a chart:



Figure 6.6 Stochastic Oscillator

In figure 6.6 you can see a Stochastic oscillator showing sell opportunities when above the overbought level (ellipses) and buying opportunities when below oversold (square).

**Final impulse**

In order to catch the final impulse towards the existing trend direction we can use a couple of different alternatives:

* a close above/ below the previous one
* oversold / overbought cross

**A close above below the previous one**

For example, a basic script to detect an impulse in the long direction is:

if close > close[1]

conversely a script to detect an impulse in the short direction is:

if close > close[1]

In alternative to this basic trigger we can use the same indicator we used to detect the pullback.

**Oversold/Overbought**

For example, to identify an up impulse after an RSI oversold we can write:

if RSI(close, RSILen) cross over oversold

where the instruction cross over will require for the RSI to have been below the oversold level and then above in the next bar.

Conversely, to catch a down impulse after an RSI overbought we can write:

if RSI(close, RSILen) cross under overbought

where the instruction cross under will require for the RSI to have been above the overbought level and then below in the next bar.

Conversely, if we chose the Stochastic oscillator we can detect an up impulse by writing:

if oslowD cross over oversold

Conversely, to catch a down impulse we can write:

if oslowD cross under overbought

where the instruction oslowD is the Stochastic value output.

**Assembling components**

We have seen so far that a trend pullback strategy consists of three components: a trend, a pullback, and a final movement in the direction of the trend. We have also seen that each of the three components can be written with different algorithms: it's a bit like having building blocks to assemble together, bearing in mind any components has its specific purpose.

In Table 6.1 you can see 9 different combinations regarding the first 2 main components: trend and pullback

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms to detect trends** | **Algorithms to detect pullbacks** | | |
|  | **SMA** | **RSI** | **Stochastic** |
| **SMA** | SMA trend + pullback below an SMA | SMA + RSI oversold | SMA + Stochastic oversold |
| **Supertrend** | Supertrend + pullback below a short SMA | Supertrend + RSI oversold | Supertrend + Stochastic oversold |
| **ADX** | ADX + pullback below a short SMA | ADX + RSI oversold | ADX + Stochastic oversold |

Table 6. 1

As you see in Table 6.1 given the same rationale you can write the strategy in different ways depending on the traders’ familiarity with technical indicators and, of course, depending on what the back-testing results will tell us.

So, for example, if we identify up-trends with 2 SMA we can write the following 3 alternatives:

if FastSMA > SlowSMA and close < PullBackSMA

if FastSMA > SlowSMA and MyRSI cross over oversold

if FastSMA > SlowSMA and oslowD cross over oversold

where the condition FastSMA > SlowSMA detects the existing up trend.

If we use the Supertrend indicator we can write the following 3 alternatives:

if close>Supertrend and close < PullBackSMA

if close>Supertrend and DMIPlus>DMIMinus and MyRSI cross over oversold

if close>Supertrend and DMIPlus>DMIMinus and oslowD cross over oversold

Where the condition close>Supertrend detects the existing up trend.

If we use ADX we can write the following 3 alternatives:

if ADXValue>AdxTrigger and DMIPlus>DMIMinus and close < PullBackSMA

if ADXValue>AdxTrigger and DMIPlus>DMIMinus and MyRSI cross over oversold

if ADXValue>AdxTrigger and DMIPlus>DMIMinus and oslowD cross over oversold

Where the condition ADXValue>AdxTrigger and DMIPlus>DMIMinus detects the existing up trend.

In the Github repository you can find the full strategies codes.

For the moment let’s select just the following combination:

* 2 SMA to select the trend
* RSI to detect a pullback
* the same RSI oversold cross-over as entry signal (impulse)

Here you find the strategy’s script:

Input: Dir(1),FastLen(20),SlowLen(200),TradeSize(10000),RSILen(5),RSITrigger(30);

Var: Qty(0);

Qty = TradeSize/Close;

if dir=1 then begin

If average(close,FastLen) > average(Close,SlowLen)

and RSI(close,RSILen) cross over RSITrigger

then Buy ("New Long") Qty Shares next bar open;

End;

if dir=-1 then begin

If average(close,FastLen) < average(Close,SlowLen)

and RSI(close,RSILen) cross under RSITrigger

then Sellshort ("New short") Qty Shares next bar open;

End;

Input:BaRSIN(20);

If BarsSinceentry = BaRSIN then begin

if mp = 1 then Sell ("xL") next bar open;

if mp =-1 then buytocover ("xS") next bar open;

End;

In Figure 6.7 you can see the entry signal on HD (Home Depot Inc.) chart

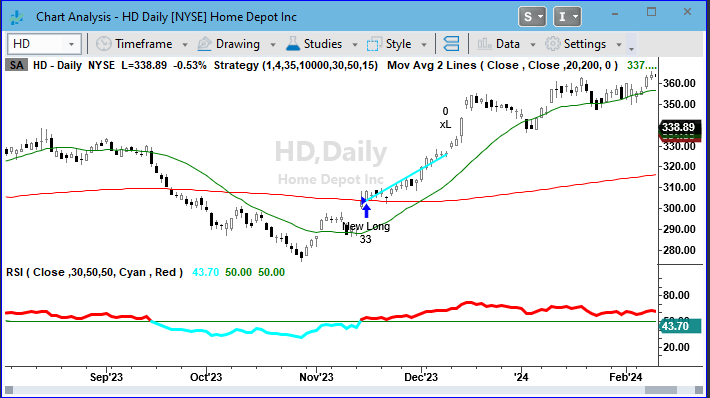


Figure 6.7 Trade Signals on chart

In Figure 6.7, an arrow indicates a NewLong signal, coinciding with the Fast Moving Average positioned above the Slow Moving Average and the RSI crossing the oversold level for the first time. Before proceeding to review performance summaries, it's crucial to ensure the accuracy of entry signals.

In Figure 6.8 you can see an equity line result for the applied strategy.

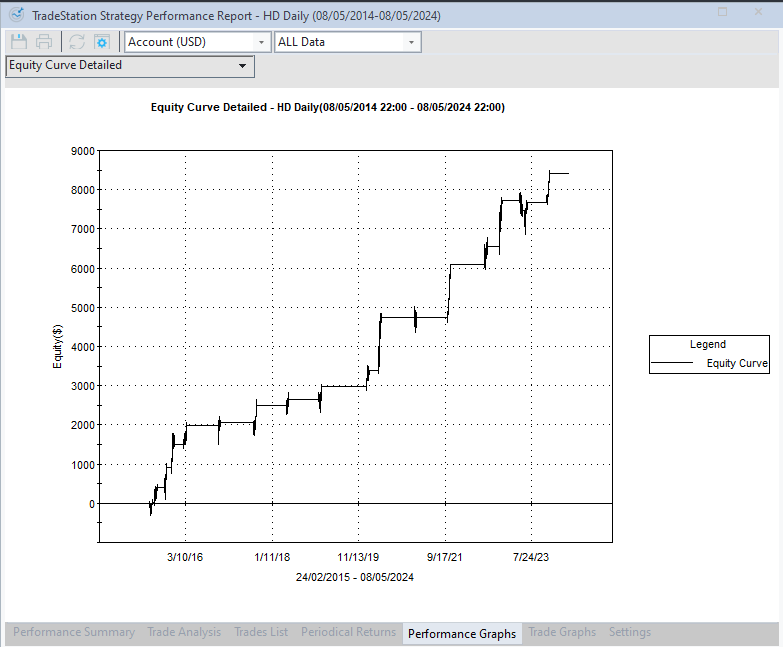


Figure 6.8 Tradestation Equity Line

In Figure 6.8 you can see the final equity line: even if the equity looks promising, it means nothing without checking the strategy robustness. For such purpose, let’s now move onto some sensitivity analysis to stress-out our strategy.

**Sensitivity analysis**

Versus the previous chapters, we now have many more parameters in the strategy, so we need to plan how to conduct a sensitivity analysis. You know that the overfitting nightmare is always lurking around the corner, so the way we proceed can make all the difference. I generally suggest sticking to the rationale. For example, we have three different components: trend, pullback, and impulse. Therefore, the best appROAch is to test them separately. This is why we are going to proceed with 2 multiple sensitivity analysis:

* A multiple sensitivity analysis on the pullback detection using RSI
* A multiple sensitivity analysis on the trend detection using 2 SMA

We don’t need to conduct a sensitivity analysis on the impulse component as it has no setting. So let’s start.

**Running an RSI multiple sensitivity analysis on HD (Home Depot Inc.)**

To establish a starting point, according to what we described in chapter 5, we are going to start with HD (Home Depot Inc.), because, if you remember, with a beta of 1.1 is right at the middle of the dow jones 30 list sorted by beta.

Let’s run a first optimization with settings as in Table 6.2

|  |  |
| --- | --- |
| **Inputs** | **Values** |
| Dir | 1 |
| FastLen | 20 |
| SlowLen | 200 |
| RSILen | From 7 to 50 increment 1 |
| RSITrigger | From 20 to 60 step 5 |
| BaRSIN | 15 |

Table 6.2 – RSI Sensitivity Analysis Settings for HD (Home Depot Inc.)

The rationale behind this optimization, which counts 396 runs, is to discern how altering the two RSI parameters, RSILen, RSITrigger, impacts the performance of the strategy concerning the selected symbol, while keeping the remaining inputs (FastLen,SlowLen, BaRSIN) constant. Essentially, we aim to determine the optimal settings to identify a pullback in a trend set by the two SMA (20, 200) and maintaining positions for 15 days. This step by step optimization based on market rationales helps a lot in avoiding overfitting.

In order to avoid overfitting, we do need sensitivity analysis. By following the same procedure of Chapter 5, we can copy the Tradestation PrintLog results and producing the following Heatmaps:

* Profit Heat Map to understand how the strategy perform in the inputs range set
* Number of Trades HeatMap to calculate how frequent the strategy trades in the inputs range set
* ROA Heat Map to select the final best parameters

In Figure 6.9 you can see a Profit Heat Map for our benchmark symbol (HD Home depot Imc.).

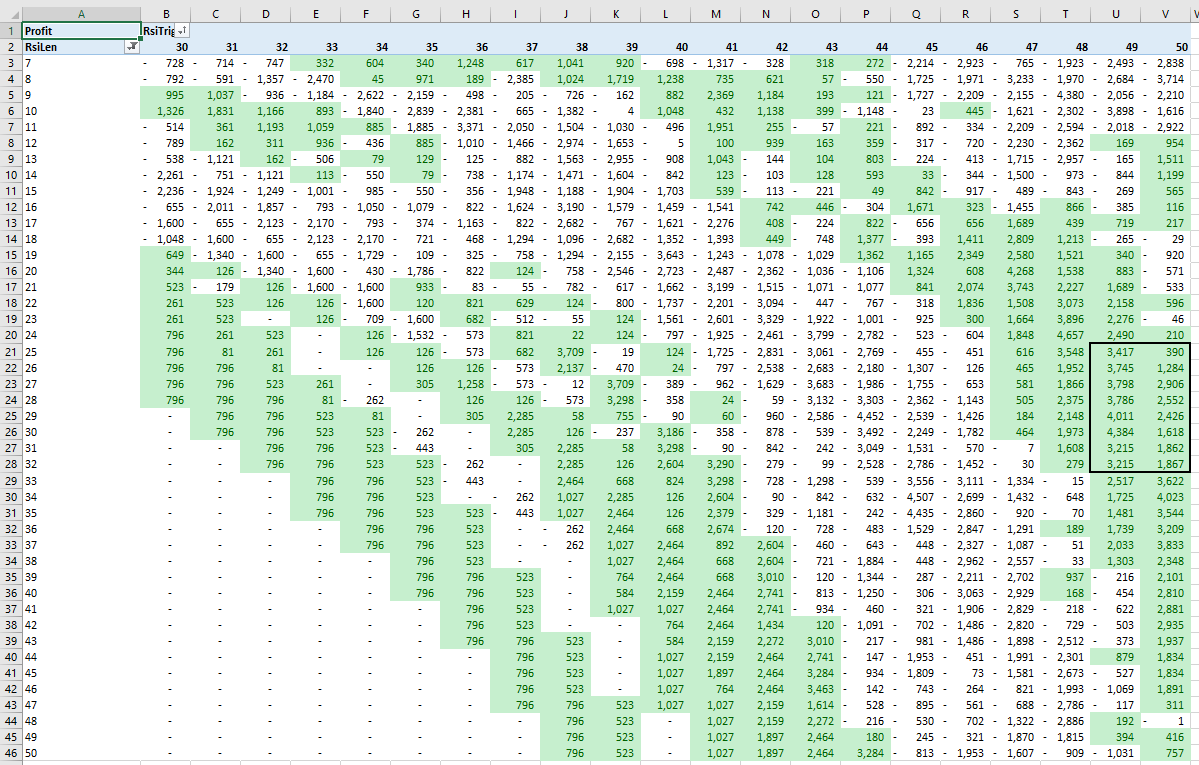


Figure 6.9 Profit Heat Map – Sensitivity Analysis RSI

As you can see in Figure 6.9 we have configured the conditional formatting to highlight the parameters combinations with a positive profit.

In order to analysize the number of trades it is very useful to build a Number of Trades Heat Map as in Figure 6.10.

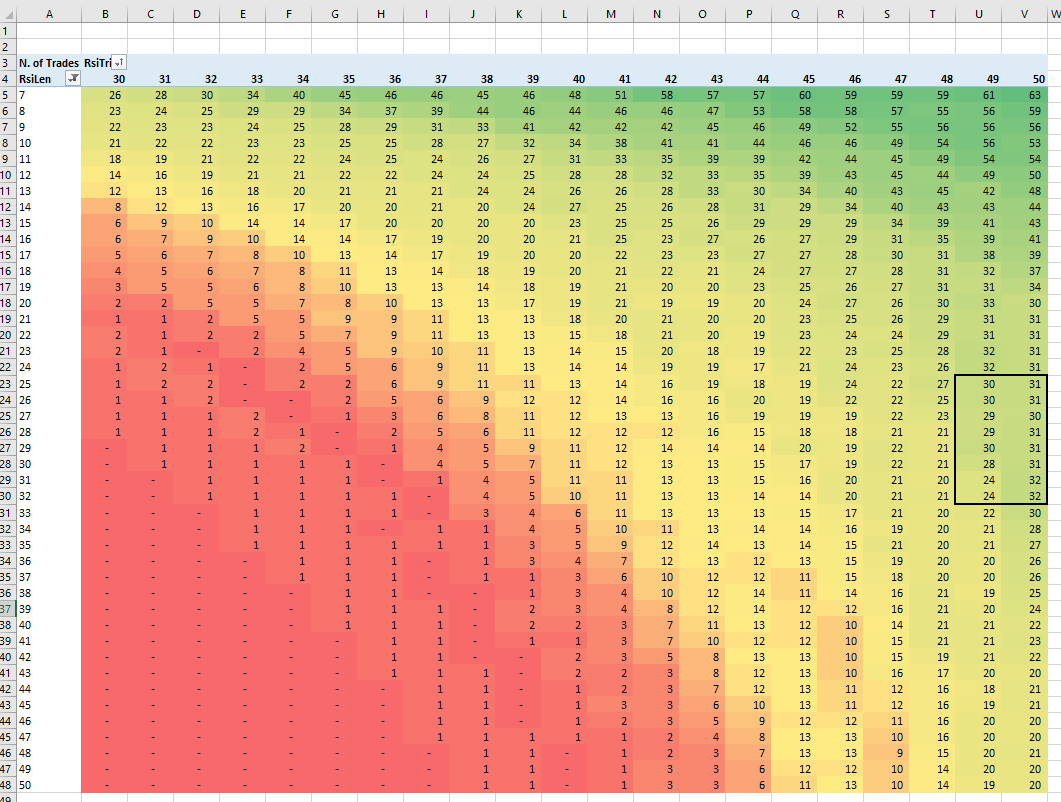


Figure 6.10 Number of Trades Heat Map Sensitivity Analysis RSI

In Figure 6.10 you can see the Number of Trades Heat Map, showing that the trades are concentrated on the top-right of the table, while on the bottom-left there is almost no-trade. Such exercise is useful when you need to analyze a new algorithm behavior, for example we can understand that the higher the RSI len, the lower the number of trades, the higher oversold level the lower the number of trades.

In order how the parameters settings move the ROA, as we have already done in the previous chpters,, we can build a ROA heat map as in figure 6.11

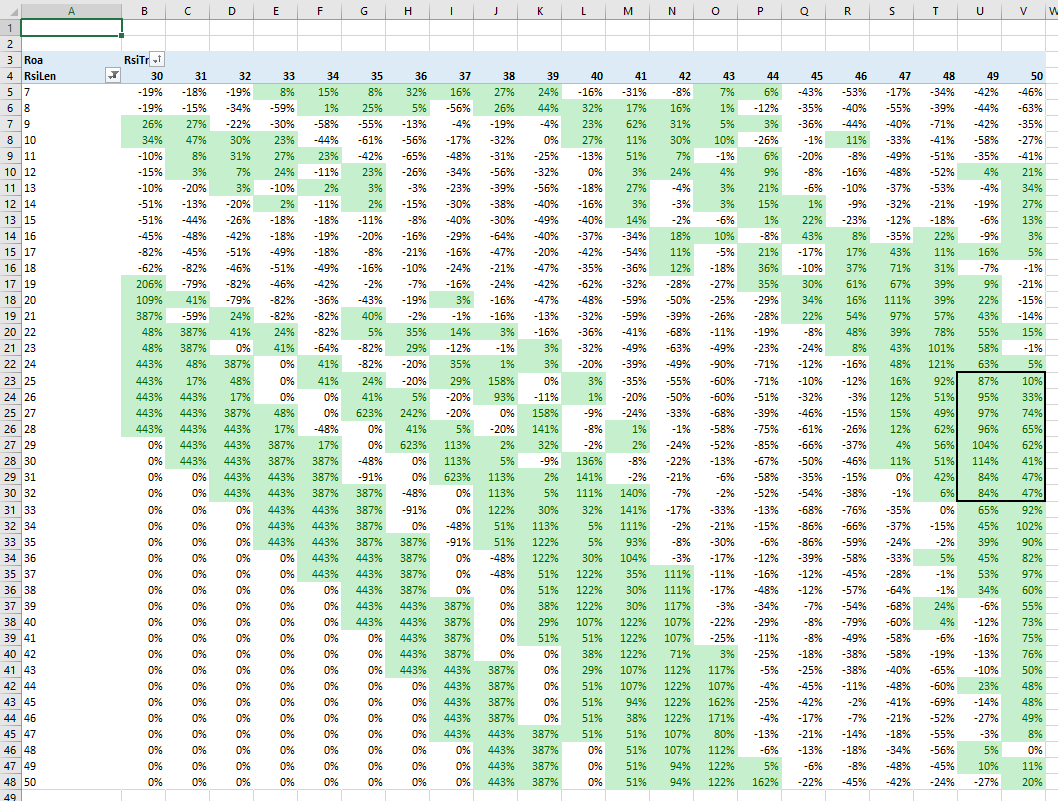


Figure 6.11 – ROA Heat Map showing Multiple Sensitivity Analysis on RSI

As you can see in Figure 6.11 we've configured the conditional formatting to highlight cells where ROA (Return on Assets) is greater than 4.

By jointly analyzing Figure 6.10 (Number of Trades Heat Map) and Figure 6.11 (ROA Heat Map), we can identify a consistent region of stable ROA, characterized by at least 30 trades, when RSI Length falls within the range of 26 to 32 and is oversold in the range of 49 to 50.

Comparing these two Heat Maps is invaluable because it can reveal if you're inadvertently focusing on a strategy with a limited number of trades, which may lack statistical significance. However, it's crucial to strike a balance between the number of trades and strategy performance to avoid overtrading, keeping in mind that even a strategy with a few trades on a single stock, may be applicable across the entire stock universe.

Let's select the settings\_ Len=30 and oversold =40 as this combination is in the middle of positive results.

**Running an SMAs multiple sensitivity analysis on HD (Home Depot Inc.)**

Once obtained the best parameter set for RSI, we need to check what’s going on with SMA lengths. So, mainintaing the best parameters set for RSI let’s run a second optimization with settings as in Table 6.3

|  |  |
| --- | --- |
| **Inputs** | **Values** |
| Dir | 1 |
| FastLen | From 3 to 30 increment 1 |
| SlowLen | From 30 to 200 increment 5 |
| RSILen | 30 |
| RSITrigger | 40 |
| BaRSIN | 15 |

Table 6.3 – SMAs Sensitivity Analysis Settings

By running an optimization with the settings in Table 6.3 we can obtain a ROA Sensitiity Analysis as In Figure 6.12

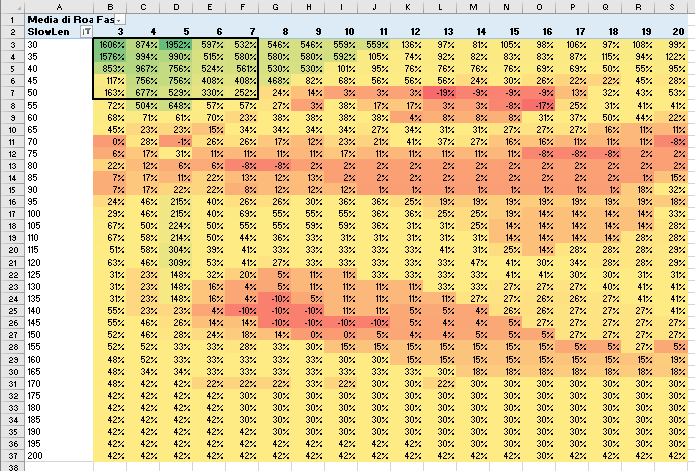


Figure 6.12 ROA Sensitiity Aalysis for SMAs

From Figure 6.12, we can observe that there is a stable area of ROA with Slowlen ranging from 30 to 50 and FastLen ranging from 3 to 7.

Let’s select the pair SlowLen=40, FastLen =5 which is right in the middle of the highlighted top left area. We now have a final parameter set, so, next step is to apply it to the entire Out of sample DowJones 30 list.

In the current validation process, we first run a double sensitivity analysis on RSI parameters while leaving the other strategy settings unaltered. Then, we conduct a double sensitivity analysis on SMA settings while not altering the other settings. When dealing with multiple sensitivity analyses, it is advisable to proceed step by step by analyzing pairs of parameters. This two-dimensional analysis allows for easier comprehension of optimization results using heat maps.

**Out of sample analysis**

When we conduct sensitivity analysis, regardless of its complexity, we are using in-sample data. On the other hand, we know that to evaluate the robustness of a strategy, it's advisable to conduct an out-of-sample analysis. The purpose of such analysis is to understand how algorithms perform on unseen data.

In this section, we are going to utilize the parameters obtained from the optimization conducted on a single asset (HD Home Depot Inc) across the entire Dow Jones 30 index.

In Figure 6.13 you can see the results of Out of sample analysis on the Dow Jones 30 list.

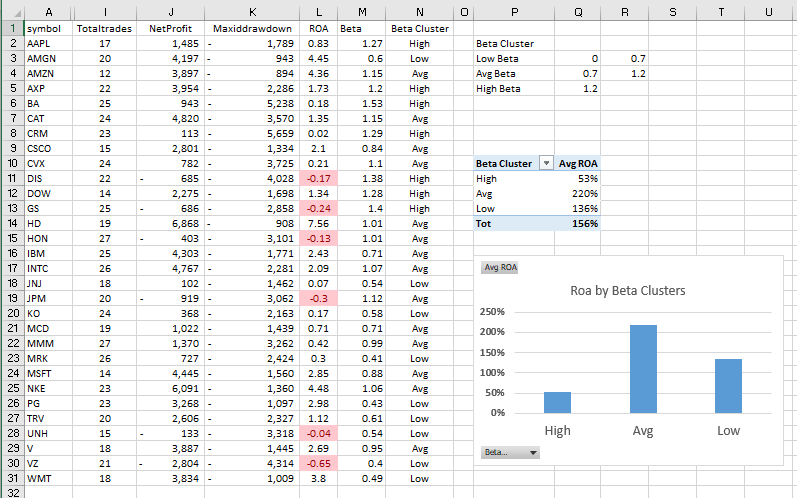


Figure 6.13 Out of sample analysis on the Dow Jones 30 list

By analysing Figure 6.13 we can draw the following considerations:

* out of 30 symbols, 24 of them show profitable results
* ROA vary from negative value to a maximum of 7.56
* Looking at the beta cluster analysis, we can see that the average beta cluster, representing stocks with volatility very similar to the index, shows an average ROA of 220%. On the contrary, both the low beta clusters and high beta clusters demonstrate lower performance.

The rationale behind the beta cluster analysis approach lies in the fact that stocks are not all equal due to their varying volatility. The volatility of individual stocks is a factor that allows us to group them and thus manage an optimization process that, on one hand, enables an out-of-sample analysis and on the other hand, is conducted on stocks that are not too diverse from each other.

**Summary**

In this chapter, we delved into a very important market pattern: the trend-pullback. Initially, we learned the rationale behind such market behavior and split it into three components: a trend, a pullback, and an impulse. Next, we wrote several algorithms to identify the different components and assemble them in various ways, akin to a game of bricks. Then, we conducted a multiple sensitivity analysis to check strategy robustness.

As you may have noticed, thus far, we focused solely on entry logics, leaving exits to only some holding periods. As you will better understand in the next chapter dedicated to Risk Management, the choice of building algorithms that separate entries from exits is not arbitrary, but rather has a valid reason behind it.