Algorithmic Trading

Abstract

In the analysis of financial time series data, techniques like support and resistance are very popular amongst traders. This text presents an approach using which the reader can quantitatively anticipate price movements in the near future, given some data.

1 Basic Terminology

Support and Resistance are used in technical analysis of stock prices. Technical analysis involves observing the trend and patterns in financial data and trying to quantitatively predict the price movement. We briefly describe the terms "**Support**" and "**Resistance**" below-

Support A support level is a trendline where the price tends to find "support" as it falls. Traders believe that the price is more likely to bounce off this level rather than break through it. At this price level, demand is thought to be strong enough to prevent the price from declining any further. However, once the price breaches this level, traders redefine a new support, taking into account the recent lag data. Buyers purchase shares of a stock near the support level, as the price of the stock is expected not to decrease.

Resistance Resistance is, in some sense, a complement to the support. Resistance refers to the historicaly observed upper bound of the price of a stock, and is expected to less likely be breached. At this price level, selling of a stock is thought to be strong enough to prevent the stock price from rising further. However, once the price has breached this level, the traders redefine a new resistance level taking into account the recent lag data.
[1]

[1] Should I add a representative figure for Support and Resistance???

Both resistance and support are characterized by straight lines in the given time series data. Taken together, they give a quantitative forecast of a range of price movements.

2 Approach

We describe an approach for a given data series with each price point located at a discretized fixed length from each other, say for example, like the represenative data series given below:-

Table 1: Apple Data

Date	Closing Price (in $)
2017-08-23	157.597
2017-08-24	156.897
2017-08-25	157.478
2017-08-28	159.064
2017-08-29	160.483
2017-08-30	160.916

It is seen that the Closing price is recorded at one day intervals. Now assume we are given data of length n i.e, n observations are recorded. The most intuitive thing to do is plot out the entire data series. (Refer Figure 1. on next page.)

We then realize that in order to find the lines that correspond to *Support* and *Resistance* we **cannot** use the entire data series obtained by us, because doing so can give us an inaccurate representation of the trends [2]. It intuitively follows that we must choose a particular window of observations, ***'l'*** called **lags** [3], the value of which will let us qualitatively plot the *Support* and *Resistance*. Once we choose a particular value of *'l'*, we can then proceed to find out the equation of lines that correspond to *Support* and

[2] Should I add a figue here that shows why??

[3] *the term lags will typically mean the number of observations that are considered while computing the trendlines.*

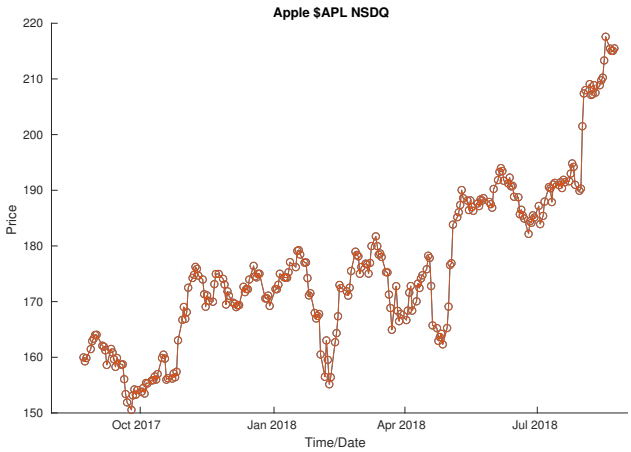

Figure 1: Apple Stock Price over FY 2017-2018

Resistance trendlines. Once this is done, we can then determine the time interval till which the *Support* and *Resistance* line is obeyed. The motivation behind this approach is to plot an empirical distribution of *time intervals* vs *lags* within which the *Support* and *Resistance* trendlines are obeyed. We rigorously state our goals, in the following paragraph.

3 Formalizing the Goals!

Suppose we are given a tuple $\{S_t = \$\ ,\ t = 0, 1, 2, ..., n\}$ corresponding to a stock data series, where S_t represents the price point of the said stock at corresponding t [4].

We denote the "present time" by t, (*Note that this 't' is different from the above mentioned t, which implies the location of* S_t *data point*). Now, given a value of l, where it denotes the units of lag, i.e, the number of observations before t, that are to be considered while estimating the trendlines for *Support* and *Resistance*. It follows then that the term *lagdata* corressponds to the tuple

$$lagdata = \{S_{t-i}, t - i\}, \text{such that i } \epsilon\ [0, l]$$

We have $|lagdata|$ a set of $[l + 1]$ points. We now restrict our attention to the *lagdata* set and consider the collection of all the lines that pass through

[4]Note that t can also represent the index of a list of S_t if such a list is made.

any two points in the *lagdata* set. It immediately is apparent that there are too many such lines possible.

Our purpose is to choose a pair of trendlines such that-

- The trendlines contain all the points ϵ *lagdata* in their spread [5].
- The spread of the trendlines chosen is the ***least***

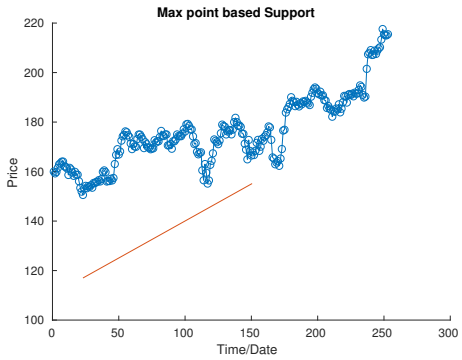

Figure 2: One of the many possible "*Support*" lines

In order to satify the desired conditions, we bring in a minimization problem. The benefit of doing so is the ability to choose a 'unique' line. For example, if we consider the data points that make up the *lagdata* set, the plot effected by our selection can be considered analogous to a piece of metal wire having the same 2-D geometry as *lagdata*. Computing a trendline that is closest to the data is possible only if we have some global parameter with which we compare all possible 'best' fits. One global parameter readily follows from the visualization just made. We simply minimize the distance of the center of gravity of the dataset (visualized as a metal piece), to the trendline that contain the dataset within their spread [6].This minimizaion can be acheived by "moving" the computed trendlines wrt the global parameter.

We explicitly outline the details below-

[5] see figure 2.
[6] see figure 3.

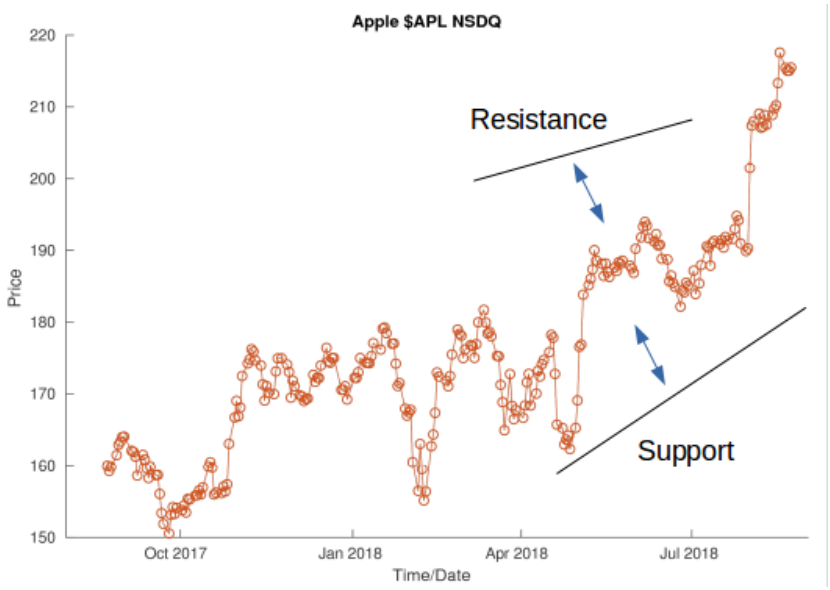

Figure 3: Moving the "*Support*" and *"Resistance"* lines

Computing the Trendlines Assume that the center of gravity is located at $g = (g_1, g_2)$ in the x-y plane. It immediately follows that the x component of the tuple (g_1, g_2) is located at $x = [t - \frac{l}{2}]$. This is because the points are given equal weights, ie the Center of Gravity will lie at the center of the x-projection of the line. The y component of the tuple (g_1, g_2) would then be located at $y = [\frac{1}{l+1} \sum_{s=0}^{l} X_s]$[7]. i.e -

$$(g_1, g_2) = (t - \frac{l}{2}, \frac{1}{l+1} \sum_{s=0}^{l} X_s)$$

We now choose two points $(s_1, X_(s1))$ and $(s_2, X_(s2))$ in the set $lagdata$. Given these two points, the equation of line that passes through them is given by[8]

$$L = X_{s_1} + \frac{X_{s_2} - X_{s_2}}{s_2 - s_1}(s - s_1)$$

The tuple $(s, L(s))$ is the locus of the desired line L.

[7]Should I explain this???
[8]give any explanation??

Now that we have the locus of the line L, we move the line towards or away from the center of gravity. Let $J(s_1, s_2)$ be the perpendicular distance between the point (g_1, g_2) and the line. Then, [9] we have-

$$J(s_1, s_2) = \frac{|X_{s_1} - X_{s_2}|g_1 - (s_2 - s_1)g_2 + (X_{s_1}s_2 - X_{s_2}s_1)}{\sqrt{(X_{s_1} - X_{s_2})^2 + (s_2 - s_1)^2}}$$

We obtain the equation for *support* by solving the minimization problem described below-

$$\min_{t-l \le s_1 < s_2 \le t} J(s_1, s_2) \text{ subject to} \{X_k \ge L(k; s_1, s_2) \forall\ 0 \le k \le l\}.$$

Let (s_1', s_2') be the minimizer for the above statement, we then can show that the equation of support is given by:

$$L(s) = X_{s_1'} + \frac{X_{s_2'} - X_{s_1'}}{s_2' - s_1'}(s - s_1')$$

The equation of resistance can then be obtained by a similar approach . refer to footnote 9

Proving uniqueness and existence [**Srishti**] It is of academic interest to prove that there exists an unique support and resistance trendline in the given data. We formalize this concept as follows.

Theorem 1. *There exists a support line for a time series data. For a given support line L_s, $(g_1, L(g_1)) \in C(A \cap L_s)$, where (g_1, S_{g_1}) is the center of gravity of $\{S_t\}$. Furthermore, let $t^* = argmin\{S_t\}$ and $t^* \neq g_1$, then the support line is unique*

Proof. $A = \{(t, S_t) | 1 \le t \le N\}$, $\#A < \infty$
S_{t^*} =min$\{S_t\}$
Consider the line $L : y = S_{t^*}$.
Case 1: $t^* < g_1$
$X_o = \{(t, S_t) | t \ge g_1\}$
We rotate the line L in counter clockwise direction. Let θ_o be the angle rotated such that the first $t_1 \in X_o$ satisfies the equation of line L.
Subcase 1: $t_1 > g_1$
In this case, we have found the required support.

[9] Again!!!... Should we explain this??

Subcase 2: $t_1 < g_1$
We replace t^* by t_1 and repeat the same procedure until we get $t \in X_o$. We are guaranteed to find such a t as A is a finite set.

Case 2: $t^* > g_1$
$Y_o = \{(t, S_t) | t \leq g_1\}$
We rotate the line L in clockwise direction. Let ω_o is the angle rotated such that first $t_1 \in Y_o$ satisfies the equation of line L. Following similar arguments as above, we can establish that a support line exists.

Case 3:$t^* = g_1$
In this case, we consider all the points in the set A $\{g_1\}$. Following the same procedure as above, we get a support line for $\{S_t\}$.

We now prove the uniqueness of the support when $t^* \neq g_1$. By the definition of support, we know that all the points of $\{S_t\}$ must lie above the support. Also, by the hypothesis condition we have, $(g_1, L(g_1)) \in C(A \cap L_s)$. Therefore, the support line is unique. □

Empirical Distribution of $\delta^+(l)$ and $\delta^-(l)$: It is important to revisit the motivation for the preceeding analysis. We essentially desired a way using which we could check the reliability of the trendlines obtained. What we essentially mean by the term reliability is, how long (i.e, for how many time steps) will the price movements be contained within the spread of lines of support and resistance. Extending this idea a bit more, we choose to analyse the data in varying slices while keeping a some parameter, constant, throughout the duration of the analysis.

For example, if we are given some data series with 200 data points, recall, we had initially endeavored to compute the trendlines given some particular value of l, the lag parameter. Lets, for the sake of this example, assume that the chosen value of l is 50. Then it follows that we could have constructed trendlines for the data in any of the 150 *sliding windows* created[10]. Thus, we can have a total of 150 such pairs of trendlines for the given value of l. We formalize this in the following paragraph.

[10]Add a representative figure???

Given any value of l, we say $\delta^+(l)$ represents the number of time steps the data series has obeyed the computed resistance trendline. Analogously, we say, $\delta^-(l)$ represents the number of time steps the data series obeys the computed support trendline. As discussed in the preceeding section, for a given value of l, the values taken by $\delta^+(l)$ and $\delta^-(l)$ are treated as random series, i.e,

$$\delta^+(l) = \{\delta^+(l)_l, \delta^+(l)_{l-1}, ...\delta^+(l)_{n-l}\}$$
$$\delta^-(l) = \{\delta^-(l)_l, \delta^-(l)_{l-1}, ...\delta^-(l)_{n-l}\}$$

In order to qualitatively deal with the data inferred from the series above, we first lay the define a few more terms.

We first define what a cumulative distribution function (CDF) is. Mathematically, a CDF of a real valued random variable X is the probability that X will take a value less than or equal to X. i.e,

$$F_X(x) = P(X \leq x)$$

We next define *Empirical Distribution* as the CDF associated with the empirical measure of a sample. i.e, it is a step function that jumps up by $\frac{1}{n}$ at each of the n data points. The value of the empirical distribution at any point is the fraction of observed values that are equal to or lessor than the chosen point. Mathematically, given a set $(x_1, x_2, ..., x_n)$ such that all $x_i \, \epsilon \, F_X(x)$

$$\hat{F}_n(t) = \frac{\text{number of elements in sample } \leq t}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{x_i \leq t}$$

where $\mathbf{1}_A$ is the indicator function. A sample empirical distribution plot has been attached.

If $\{y_{(1)}, y_{(2)}, ..., y_{(n)}\}$ is an ordered set; the empirical distribution would look like :-

It now follows that for a given value of l; we can have atmost $(n - l)$ points, ie, there are atmost $n - l$ values of δ. We can plot the empirical distributions for each given value of l and analyse the results to find out the optimal choice that one should make for l.

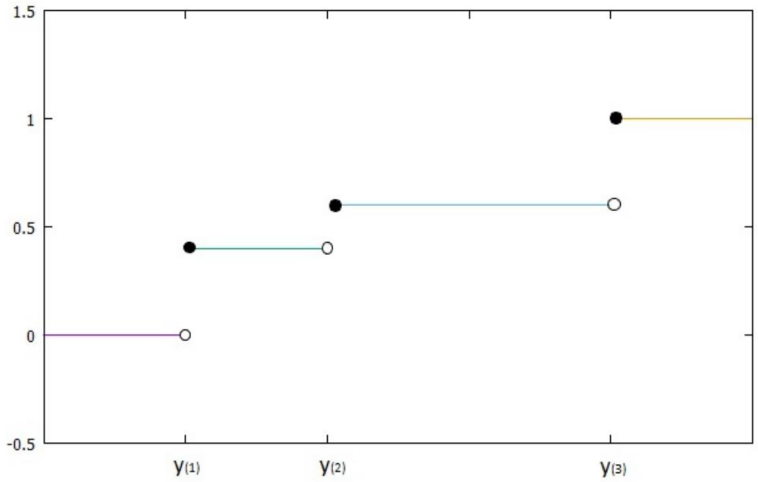

Figure 4: Representative CDF for an ordered set of n iid's.

Now that we have laid down the necessary groundwork, in the next section, we formalize the pseudo code version of the analysis that is conducted on any given data series.

4 Approaching the data; Algorithmically

We write the pseudocode in a modular form; ie; outlining the process in a block format.

Step 1 Finding the trendlines

```
# The SnR module
# Pseudocode for Generating the Support and Resistance
   Trendlines

#INPUT = Array of Price points, value of 'l', & 't'

def snr():

# Code
if 'l' > 't';
```

```
    return ()
else;
    pass

for ('l' & 't');
    Construct Sliding Window till 't-l'
    S = #Support
    R = #Resistance

    #NOTE: Computed using the formulae described above
end

for (S & R)
    delta_p = # value(s) of t at which price point crosses R
    delta_m = # value(s) of t for which price point crosses S

return(S, R)


#OUTPUT = Equation of trendlines; based on 'l' & 't' ; & the
    values of delta_p and delta_m
```

Step 2 Computing $\delta^{-}(l)$ random series

```
    # The delta module
    # Pseudocode to generate the random series delta for
        varying t
    def deltaser():

    #INPUT : value of 't' and 'l'

        random_series = []

        for i in range(t, t-l)
            p = snr(data, l, i)
            random_series.append(p)

        return (random_series)
```

Step 3 Computing empirical distribution

```
# Pseudo Code for Computing the Empirical Distribution
#Input = Output of deltaser() function

for i in range(1, n)
   sort(deltaser(i))

   val = max(deltaser(i))
   plotvalues = val/length(deltaser(i))

empirical_plot = plot(plotvalues)
return (empiricial_plot)
```

5 Actualizing the Algorithm

[**Purva**] We obtained data of the stock movement of Apple Inc. from NASDAQ. The data was then cleaned and the code attached in the appendix was used to generate the following plot representing the empirical distribution.

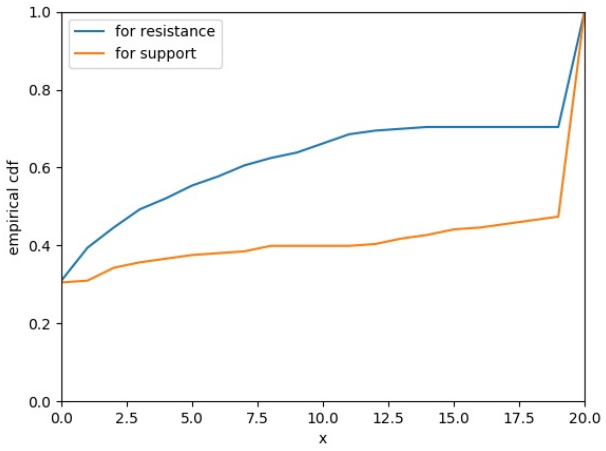

Figure 5: CDF Obtained after simulation

Appendix A [Purva]Code for generating the cdf

```
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as plt
#Given l and time t, find out the support and resistance
    by finding out the time points, through which the lines
    should pass.
#col is the input vector of time series.
# The output is a vector m[0,1,2,3] where m[1] is a list
    containing the time points so that the support line is
    obtained by joining (m[1][0],col[m[1][0]]) and
    (m[1][1], col[m[1][1]]).
# m[0] gives the distance of CG from the support line.
# Similarly m[2] gives the distance of CG from the
    resistance line and m[3] gives the resistance line.

#defining function SnR
def SnR(data,t,l):
cond1=[[0 for x in range(l+1)] for y in range(l+1)]
cond2=[[0 for x in range(l+1)] for y in range(l+1)]
J=0
m=[0 for x in range(4)]
x=list(data[t-l:t+1])
g=[t-0.5*l,sum(x)/(l+1)]
m[0]=10**8
m[2]=10**8
if l%2==1:
for i in range(0, math.floor(l/2)+1):
for j in range(math.floor(l/2)+1,l+1):
if i-j==0:
continue
a=np.abs((x[i]-x[j])*g[0]-(i-j)*g[1]+ (x[j]*i-x[i]*j))
b=np.sqrt((x[i]-x[j])**2+(i-j)**2)
J=a/b
C= (x[i]-x[j])/(i-j)
for k in range (0,l+1):
```

```
if x[k]< x[j]+C*(k-j):
cond1[i][j]+=1
elif x[k]== x[j]+C*(k-j):
cond2[i][j]+=1
if cond1[i][j]==0:
m[0]=min(m[0],J)
if m[0]==J:
m[1]=[i,j]
elif cond1[i][j]+cond2[i][j]==l:
m[2]=min(m[2],J)
if m[2]==J:
m[3]=[i,j]
if l%2==0:
for i in range(0, math.floor(l/2)+1):
for j in range(math.floor(l/2),l+1):
a=np.abs((x[i]-x[j])*g[0]-(i-j)*g[1]+ (x[j]*i-x[i]*j))
b=np.sqrt((x[i]-x[j])**2+(i-j)**2)
J=a/b
C= (x[i]-x[j])/(i-j)
for k in range (0,l+1):
if x[k]< x[j]+C*(k-j):
cond1[i][j]+=1
elif x[k]== x[j]+C*(k-j):
cond2[i][j]+=1
if cond1[i][j]== 0:
m[0]=min(m[0],J)
if m[0]==J:
m[1]=[i,j]
elif cond1[i][j]+cond2[i][j]==l:
m[2]=min(m[2],J)
if m[2]==J:
m[3]=[i,j]
m[1][0]+=t-l
m[1][1]+=t-l
m[3][0]+=t-l
m[3][1]+=t-l
x1, y1 = m[1],[data[m[1][0]],data[m[1][1]]]
x2, y2 = m[3],[data[m[3][0]],data[m[3][1]]]
return [x1,y1,x2,y2]
```

```
#defining function to compute delta
def delta(data,t,l,x_sup, y_sup, x_res, y_res):
x=list(data[t-l:t+1])
# for calculating delta_plus (based on resistance equation)
for i in range(t+1, len(data)):
L_plus = y_res[1] +
    ((y_res[1]-y_res[0])*(i-x_res[1])/(x_res[1]-x_res[0]))
if data[i] > L_plus:
break
if i >= t+l+1:
i = min(i,l)
delta_plus = i
else:
delta_plus = i-t-1

#for calculating delta_minus (based on support equation)
for j in range(t+1, len(data)):
L_minus = y_sup[1] +
    ((y_sup[1]-y_sup[0])*(j-x_sup[1])/(x_sup[1]-x_sup[0]))
if data[i] < L_minus:
break
if j >= t+l+1:
j = min(j,l)
delta_minus = l
else:
delta_minus = j-t-1
return [delta_plus,delta_minus]

#defining function to determine the empirical distribution
    of delta_plus and delta_minus
def emp_delta(delta_plus_list, delta_minus_list):
    #delta_plus_list and delta_minus_list are arrays
    containing delta values for t = l, l+1, ... , N
A = sorted(delta_plus_list)
B = sorted(delta_minus_list)
F_delta_plus = [0 for x in range(max(delta_plus_list)+1)]
F_delta_minus = [0 for x in range(max(delta_minus_list)+1)]
#empirical distribution of delta_plus
for k in range(max(delta_plus_list)+1):
count1 = 0
```

```
for i in range(len(A)):
if A[i]<= k:
count1 = count1 +1
else:
break
F_delta_plus[k] = count1/(len(delta_plus_list))

#empirical distribution of delta_minus
for k in range(max(delta_minus_list)+1):
count = 0
for j in range(len(B)):
if B[j]<= k:
count = count +1
else:
break
F_delta_minus[k] = count/(len(delta_minus_list))
return [F_delta_plus, F_delta_minus]

#calling the functions

data = pd.read_csv('/Datasets/apple_c.csv')
y = data['Close']
l = 20
delta_plus = [0 for z in range(l,len(y)-l)]
delta_minus = [0 for z in range(l,len(y)-l)]
for t in range(l+1,len(y)-l):
[x1,y1,x2,y2] = SnR(y, t, l)
[a,b] = delta(y,t,l,x1,y1,x2,y2)
delta_plus[t-20] = a
delta_minus[t-20] = b

[emp_delta_plus, emp_delta_minus] = emp_delta(delta_plus,
    delta_minus)
emp_delta_plus =
    list(np.around(np.array(emp_delta_plus),3))
emp_delta_minus =
    list(np.around(np.array(emp_delta_minus),3))

print ("Delta_plus = ", delta_plus)
print ("Delta_minus = ", delta_minus)
```

```
print ("Empirical distribution of Delta_plus = ",
    emp_delta_plus)
#print (len(emp_delta_plus))
print ("Empirical distribution of Delta_minus = ",
    emp_delta_minus)
#print (len(emp_delta_minus))

#plotting the cdf
x = []
for i in range(l+1):
x.append(i)
plt.plot(x,emp_delta_plus, label = 'for resistance')
plt.plot(x,emp_delta_minus, label = 'for support')
plt.xlabel('x')
plt.ylabel('empirical cdf')
plt.title = 'Graph of empirical cdf of delta values for
    support and resistance'
plt.xlim(0,l)
plt.ylim(0,1)
plt.legend()
plt.show()
```