Lecture 4

First Execution Algos

Target Functions

Tolerance Factors

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Administrative

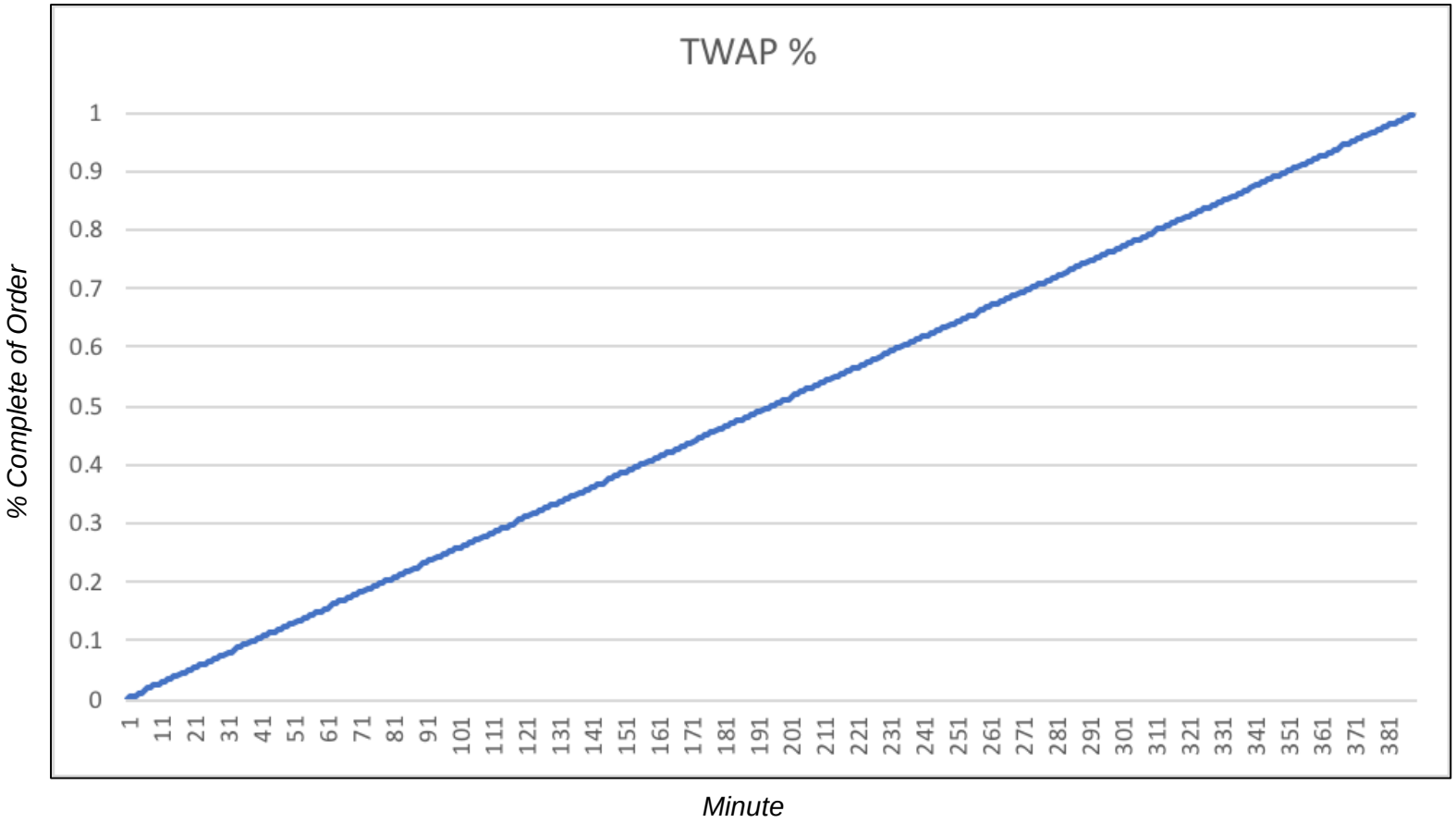
- First Homework out this weekend
  - No, really. I promise!
- Group Project Proposal to be assigned as well, due ~week 8 or 9.
  - 1 page or less, describe what you want to do, any background research, what market data you need, etc.
  - No rush, but start thinking about it.

Lecture 4 Agenda

- Finish Lecture 3 – From Implementation Shortfall
- Our First Execution Strategy: TWAP
- Tick data / algo test harness
- Algos with Tick Data
- Target Functions
- Tolerance Factors

Our First Execution Strategy: TWAP

TWAP – Time Weighted Average Price



Given *n* total shares and *m* minutes to trade,
execute $\frac{n}{m}$ shares per minute

TWAP

TWAP – minute data.ipynb

Introduction to a Strategy Simulation Framework

Implementing a Test Harness

1. Get market data
2. Pre-process into standardized fields
3. Sanity check data (mins, maxes, etc.)
4. Load data and loop
5. Collect quotes and market data
6. Implement any sanity checking
7. Execute your strategy!
8. Collect output
    a) Trade count (shares, notional)
    b) Performance / P&L
    c) Slippage

Implementing a Test Harness

1. simtools.py

2. Tick Data Setup.ipynb

3. Trading Strategy Test Harness.ipynb

A Naïve TWAP with Tick Data

TWAP - tick.ipynb

Shortcomings of the Naïve Approach

- In the example TWAP, the simplistic formulation is aggressive and reactive.
- The strategy “chases” volume after a predefined interval is complete.
- We know that market prices will fluctuate within each interval, and we should opportunistically be able to switch between passive and aggressive styles in order to improve performance and capture spread where possible.
- We can address these issues by defining a **target function** and a **tolerance factor.**

Target Functions

Target Functions

Both TWAP and VWAP require you to know what your target % complete is at any given time.

- Create a cumulative volume function such that:

  Target % complete at time $t = f(t)$

  TWAP : even distribution over n bins, e.g. 1/390 * t total shares to each bin

- Cumulative TWAP:

  Target % complete at time t = bin t / total bins * total shares

  e.g. for 1 minute bins, an order to buy 100000 shares at minute 39 should be

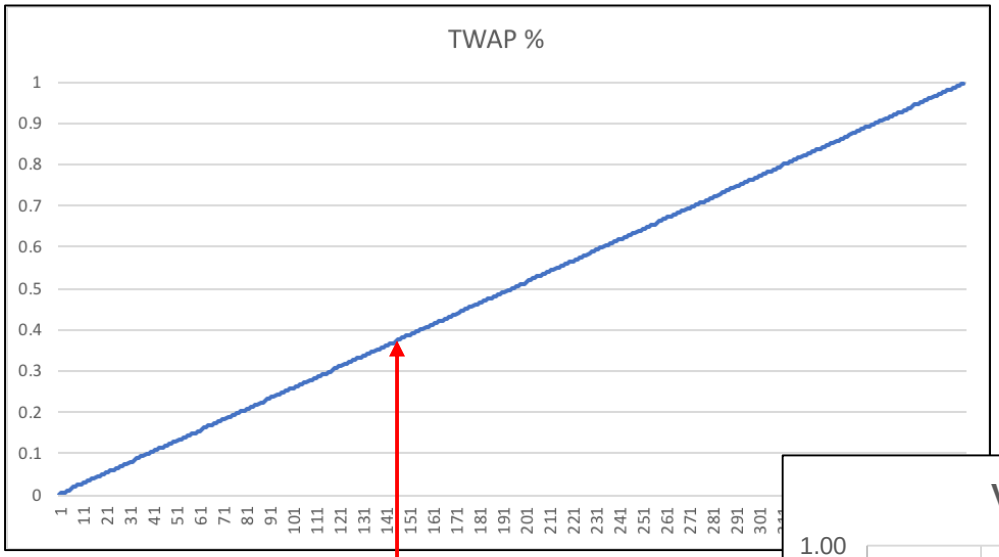
  (39 / 390) * 100000 = 10000 complete.

- Cumulative VWAP : Target % complete at time t = something like

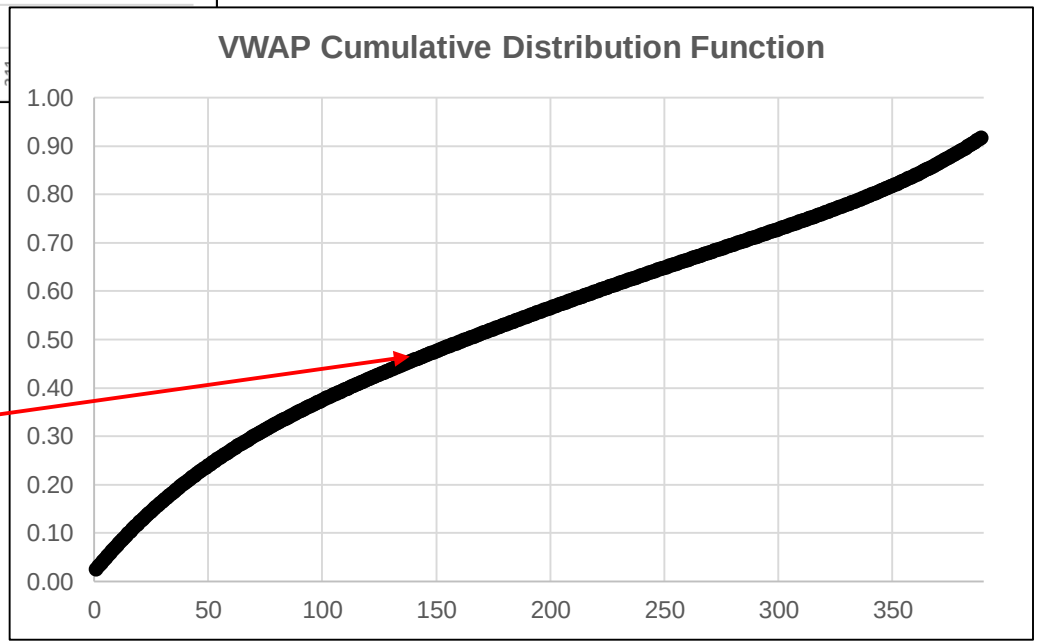
  Target % = $c_1t + c_2t^2 + c_3t^3 + c_4t^4 + c_5t^5$

  More on this later…

Target Function – Visual Intuition



Charts of cumulative volume, i.e. total order quantity q = 100%



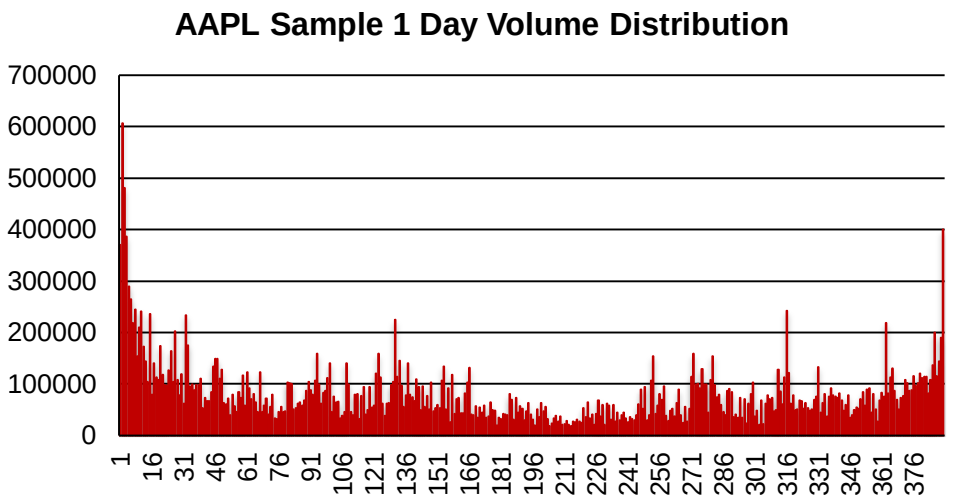
At time *t* you should be *x%* complete

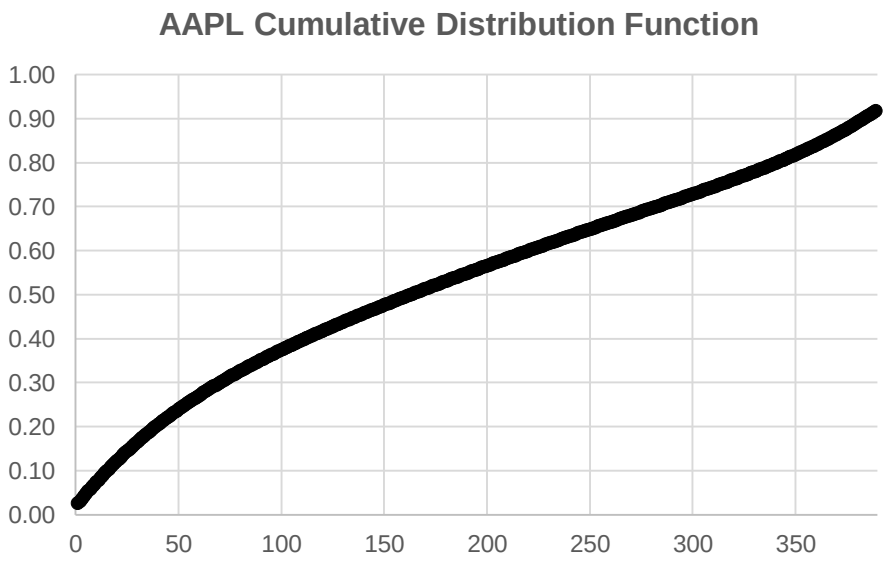
VWAP: Volume Distribution Profile

1. The distribution of volume over the course of a trading day in the US is quite stable.

2. It can be used to estimate the proportion of volume that will trade *on average* in a given segment of the day.

3. Using the profile as a guide, you can apply other techniques to improve order placement and volume prediction





Building a Volume Profile - Data

AAPL US Equity
BarTp=T

Date	OPEN	HIGH	LOW	LAST_PRICE	NUMBER_TICKS	VOLUME	VALUE
9/15/16 9:30	113.86	114.25	113.49	114.09	6,141	2,531,717	288,265,408
9/15/16 9:31	114.099	114.35	114	114.1099	5,341	1,390,148	158,717,056
9/15/16 9:32	114.1	114.28	113.72	113.7519	3,637	948,012	108,137,632
9/15/16 9:33	113.8	114.01	113.74	113.91	2,827	640,697	72,981,640
9/15/16 9:34	113.9001	114.2	113.84	114.17	2,418	580,885	66,263,004
9/15/16 9:35	114.17	114.4	114.17	114.252	3,742	858,753	98,137,392
9/15/16 9:36	114.26	114.47	114.21	114.435	2,945	744,893	85,186,472
9/15/16 9:37	114.445	114.65	114.43	114.554	3,938	939,103	107,570,568
9/15/16 9:38	114.569	114.73	114.51	114.61	3,243	773,059	88,624,216
9/15/16 9:39	114.62	114.62	114.3	114.3799	2,071	529,009	60,537,704
9/15/16 9:40	114.37	114.51	114.33	114.36	1,833	450,583	51,565,112
9/15/16 9:41	114.35	114.4	114.1799	114.35	2,638	668,989	76,453,024
9/15/16 9:42	114.36	114.43	114.34	114.34	1,544	356,846	40,817,200
9/15/16 9:43	114.34	114.36	114.19	114.35	2,083	427,851	48,893,000
9/15/16 9:44	114.345	114.36	114.13	114.26	2,261	434,212	49,595,816
9/15/16 9:45	114.26	114.26	114	114.01	2,174	536,700	61,239,476

Date	bin	VOLUME	CUM_VOL	PCT
9/15/16 9:30	1	2,531,717	2,531,717	0.03
9/15/16 9:31	2	1,390,148	3,921,865	0.05
9/15/16 9:32	3	948,012	4,869,877	0.06
9/15/16 9:33	4	640,697	5,510,574	0.07
9/15/16 9:34	5	580,885	6,091,459	0.07
9/15/16 9:35	6	858,753	6,950,212	0.08
9/15/16 9:36	7	744,893	7,695,105	0.09
9/15/16 9:37	8	939,103	8,634,208	0.1
9/15/16 9:38	9	773,059	9,407,267	0.11
9/15/16 9:39	10	529,009	9,936,276	0.12
9/15/16 9:40	11	450,583	10,386,859	0.13
9/15/16 9:41	12	668,989	11,055,848	0.13
9/15/16 9:42	13	356,846	11,412,694	0.14
9/15/16 9:43	14	427,851	11,840,545	0.14
9/15/16 9:44	15	434,212	12,274,757	0.15

1. Decide on data requirements
   - Bar size (1 minute in this example)
   - Number of days of data (often a rolling 30 day window)
2. Acquire historical data
   - Open, High, Low, Close and Volume is fine
3. Prepare the data
   - Convert timestamp to bin (1-390 for US Stock Market)
   - Calculate Cumulative Volume
   - Normalize Cumulative Volume -> Total volume = 1%
4. Fit the data via regression or some other method

A Basic VWAP Algo

Order Placement

1. Target Volume = volume_at_time(q, t) where q = total order quantity and t = time segment

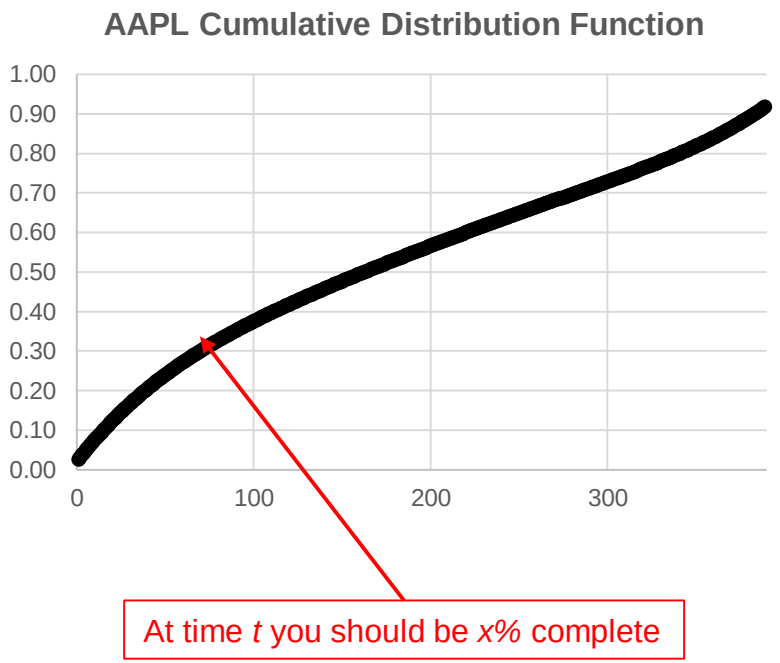
2. Target order size at time *t* = Target volume at time *t* – Executed volume

3. Example Primitive Order Placement Strategy
   - Define Shortfall => Target volume at time *t* – Executed Volume
   - Threshold value is the tolerance for being off target volume distribution
   - Passive if Shortfall <= threshold value
   - Aggressive if Shortfall > threshold value

4. Advanced versions may leverage non-displayed orders, etc. depending on size of shortfall.

Target the Volume Weighted Average Price p_{VWAP}:

$$p_{VWAP} = \frac{\sum(quantity_{@price} \times price)}{total\ quantity}$$



Bar-wise Averaging

Rather than fit a function, create a normalized average curve

Given a selected set of minute bar data:

1. Calculate cumulative “target % complete” as in Method 1
2. Calculate bin-wise average “target % complete” over the population period

A Simple VWAP

VWAP Profile v2.ipynb

VWAP With Tick Data v1.ipynb

VWAP – Other Considerations

- What is your volume profile population?
  - Stock?
  - Sector?
  - Index?
- Special Volume profiles for planned events (FOMC, Expiration, etc.)
- Data filtering options
- Effect of Limit price on performance
  - Limits marketable volume if tight limit
  - How to handle?
- Block prints at run-time: Exclude or include?
- Impact of exogenous events; update volume distribution prediction?

Another Target Function: Implementation Shortfall

$$y = 1 - b^t$$

where

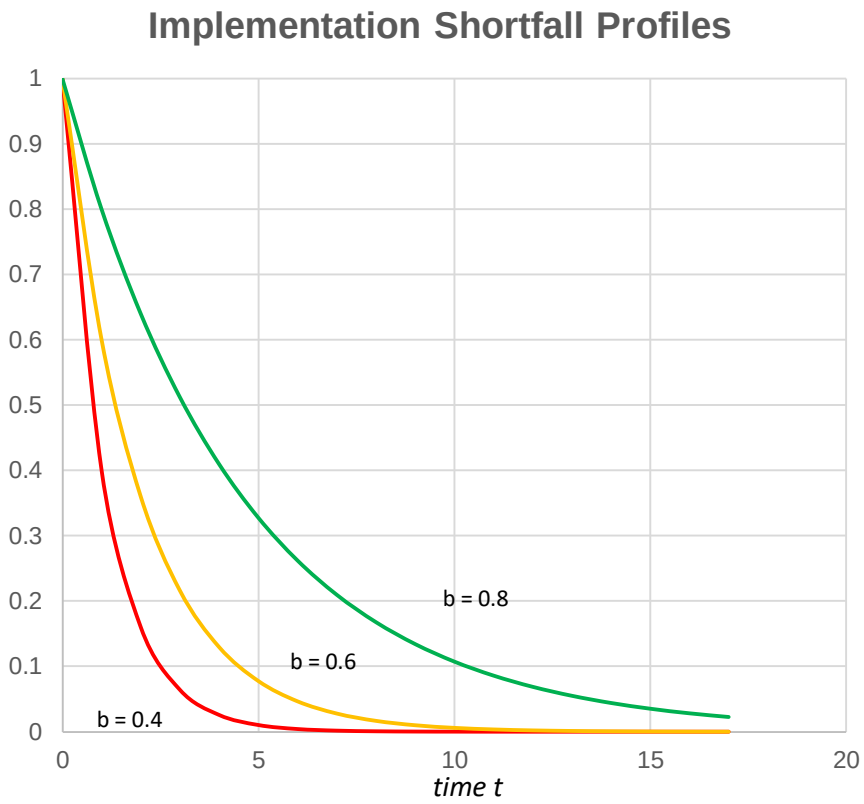
y = % of order complete at time *t*

b = risk tolerance 0 <= b <= 1,

t = minutes from start

Opportunity cost / impact trade off

- Common parameters
  - Volume caps
  - Aggressiveness / patience (tuning the tradeoff)
- Models vs. heuristics
- Cleanup logic



Tolerance Factors, Theoretical Values and Multi-factor Price Models

Tolerance Factors

We have now established the concept of target functions.

The next step is to consider how we will allow ourselves to vary from the deterministic target function over time in order to take advantage of market fluctuations, short term forecasts, etc.

We will define the concept of **tolerance factor.**

Tolerance Factors

Establishing a Tolerance factor:

- How far can you get behind (or ahead)
- Think of this factor as a proxy for **risk tolerance**

Simple intuition:

- Tolerance is high = patient. Capture spread, optimize for cost
- Tolerance is low = aggressive. Stay close to target, pay spread. Risk averse

Threshold model A:

1. Calculate a threshold value, when you are behind by some amount > threshold value then trade aggressively until amount behind < threshold

2. Threshold value = % behind
   % behind = 1- (actual / target)

Our Current Setup

Assume we are buying and consider if we have a model with three signals

Without Tolerance

```
if current_shares_behind <= 0: # we're passive
    buy at bid

else: # we're behind, get aggressive
    buy at offer
```

The Simple Tolerance Function

In the context of our simple VWAP logic, consider this simple tolerance function:

```
# we can be as many as n shares behind
Allowed_behind = n

if current_shares_behind <= allowed_behind:
    # not behind
    place_passive_order() # or do nothing
else:
    # we're behind
    place_aggressive_order()
```

A Work on Order State Management

Now add order state management. When market data ticks:

```
allowed_behind = n
if current_shares_behind <= allowed_behind: # we're passive
        if live_order:
                # do nothing, or maybe check price
                # for simulation purposes let's see if we got filled
                if ( side == 'b' and trade_price <= order_price ) || (
                     side == 's' and trade_price >= order_price ):
                        # we got filled!
                        update_order_quantity()
                        process_trade()
                        check_if_new_order_needed()
        else:
                place_passive_order()
                order_open = True

else: # we're behind, get aggressive
        if order_open:
                cancel_open_order():
        # now we can place our order
        place_aggressive_order()
```

Multiple Factors

But we're smart!

We have a bunch of alpha signals that can make our trading smarter.

How do we manage multiple signals?

Rules and Multiple Signals

Assume we are buying and consider if we have a model with three signals

Signal 1

```
if current_shares_behind <= allowed_behind: # we're passive
        buy at bid

else: # we're behind, get aggressive
        buy at offer
```

Signal 2

```
if momentum_signal > momentum_threshold: # trade aggressively
        buy at offer or sell at bid
```

Signal 3

```
if reversion_signal >= reversion_threshold: # trade passively
        buy at bid
```

Combinations of Factors

Before we get to linear combinations, let's consider multiple factors in a simple "voting" mechanism.

Example Assumptions:

1. Consider a VWAP **buy** order
2. We can only buy on the bid (passive) or the offer (aggressive). Signal must be +0.5 to be aggressive
3. Assume equal weighting of all factors

Case	Schedule	Momentum	Reversion	FV adjustment
1	On schedule: 0	Tick up: +0.5	Neutral: 0	+0.5 aggressive
2	Very behind: +0.5	Tick down: -0.5	Neutral: 0	0.0: passive
3	Very behind: +0.5	Tick down: -0.5	Down: -0.5	-0.5 passive
4	A little behind: +0.2	Tick up (weaker) +0.2	Up (weak): +0.1	+0.5 aggressive

This gets even more interesting for non-execution strategies with bi-directional signals!

Multiple Signals and Theoretical Values

We need to be able to combine multiple factors into one expression of a desire to trade at a given price (passive, aggressive, or anywhere in between)

Applicable to any signal, not just tick-level / microstructure

Tolerance Factors: Model B

- Apply adjustment as a continuous factor
  - Combines with other factors
  - Can support concept of “ahead” as well as “behind” (i.e. if my short term signals are strong, can I trade ahead to some degree?)

Tolerance Factors: Model B

- Use % ahead/behind to adjust target price continuously
  - Start with a mid or "micro" price P_m = midpoint of spread
  - Apply "catchup" factor F_c such that $0.0 < F_c < (Q_{ask} - Q_{bid})$:
  - If I'm buying, and not behind, my catchup factor is 0. buy at bid
  - If I'm buying and sufficiently behind, let's say my catchup factor is 1.0. Buy at offer
  - If I'm only slightly behind, F_c might be 0.2
  - Buy Order Placement: "Fair" price $P_f = P_m – ((Q_{ask} - Q_{bid}) / 2) + F_c$
  - $F_c = k(Q_{ask} - Q_{bid})$
  - k = c * (target - actual) / total shares
  - c = calibration factor – tune based on empirical performance

Borrowing from Arbitrage Pricing Theory

Consider Arbitrage Pricing Theory (APT)

(Ross, 1976), A decent intro at: https://en.m.wikipedia.org/wiki/Arbitrage_pricing_theory

A stock's return can be expressed as a *linear combination* of multiple factors:

$$r_j = a_j + b_{j1}F_1 + b_{j2}F_2 + \cdots + b_{jn}F_n + \epsilon_j$$

Where:

a_j = constant factor for stock j

F_k = factor

b_{jk}= stock *j's* sensitivity to factor F_k

Fair Value

How does this help us?

We can add arbitrarily many factors in linear combination to arrive at a predicted return and a "fair value" or "*FV*" of a stock.

$$FV = s_j + r_j$$

Factors can be

- *signals* such as tick test, order imbalance
- *risk-based*
- or anything else you can think of

Fair Value Factors - Intuition

When we think something is over priced we sell:

Lower FV versus the market

When we think something is under priced, we buy:

Raise FV so stock looks undervalued by the market

Signal Factor Example: Tick Test

How do we use the theoretical value in trading decisions?

1. Consider a simplified conceptual version of a tick test signal T_j
2. Derive a signal from autocorrelation of trade directions:
   - State 1: Next tick predicted up
   - State 2: Next tick predicted down
   - State 3: Next tick indeterminate
3. Assume for the moment we want to trade immediately based on that signal
4. For a stock j, define price sj to be the midpoint within the current quote.

We can then express our three states as:

- State 1: Next tick predicted up (buy): FV = the offer price = midpoint + spread/2
- State 2: Next tick predicted down (sell): FV = the bid price = midpoint – spread/2
- State 3: Next tick indeterminate: FV = midpoint

Signal Factor Example: Tick Test

More generally, we can express this the signal as a continuous function:

$$FV = midpoint_j + b_{jF}F_j$$

Where

b_{jF} = loading of the tick signal factor

F_j = tick signal factor described in spread units in the range [–0.5..0.5]

- State 1: Next tick predicted up (buy): FV = the offer price = midpoint + 0.5
- State 2: Next tick predicted down (sell): FV = the bid price = midpoint – 0.5
- State 3: Next tick indeterminate: FV = midpoint + 0
- States between 1 and 2: FV = midpoint + F_j spread units

Risk-based Factor Example: Position

Consider a proprietary trading strategy in which a variety of signals are working.

In the absence of information you would prefer to *not* have a position:

1. If you are long you would prefer to be flat (sell to close)

2. If you are short, you would prefer to be flat (buy to close)

Risk-Based Factor Example: Position

We can again express this the signal as a continuous function:

$$FV = midpoint_j - b_{jF}F_j$$

Where

b_{jF} = loading of the position factor

F_j = risk adjustment factor described in spread units in the range [–0.5..0.5]

- State 1: Long (sell): FV = the bid price = midpoint – 0.5
- State 2: Short (buy): FV = the offer price = midpoint + 0.5
- State 3: 0 position: FV = midpoint + 0
- States between 1 and 2: FV = midpoint + F_j spread units

Risk-Based Factor Example: Schedule

Consider a schedule based strategy.

"Risk" is defined as the difference between your current position and your target position. The greater the difference (i.e. the greater behind), the greater your risk

If you are buying:

1. If you are behind, you want to buy more: raise FV

2. If you are ahead, you want to buy less: lower FV

3. If you are on schedule, you are neutral. FV at midpoint

Risk-Based Factor Example: Schedule

And again we can again express this the signal as a continuous function:

$$FV = midpoint_j + b_{jF}F_j$$

Where

b_{jF} = loading of the tick signal factor

F_j = schedule factor described in spread units in some range, e.g. [-0.5..0.5]

- State 1: behind (buying): FV = the offer price = midpoint + 0.5 # trade now
- State 2: ahead (buying): FV = the bid price = midpoint – 0.5 # slow down
- State 3: on schedule: FV = midpoint + 0 # no impact
- States between 1 and 2: FV = midpoint + F_j spread units

- Reverse signs for selling

Putting it together for VWAP

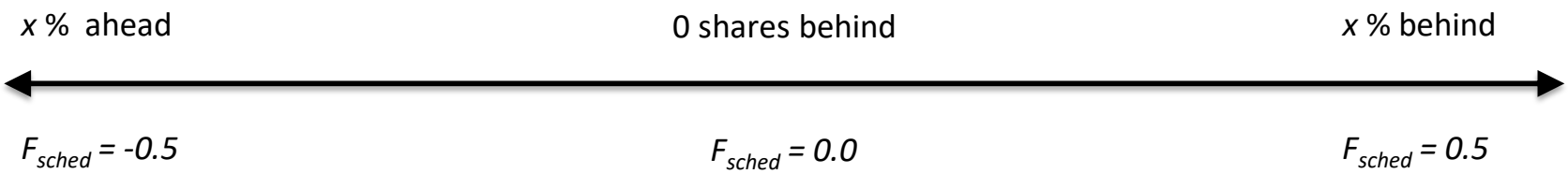
$$FV_j = midpoint_j + b_j F_{tick} + b_j F_{imbal} + b_j F_{schedule}$$

Where for stock *j*:

F_k = factor for each signal or adjustment

b_{jk}= stock *j's* sensitivity to factor F_k

VWAP Schedule Factor Construction



$$F_{schedule} = \frac{q_{pct\_behind}}{q_{\max\_behind}} \times \frac{spread_{avg}}{2}$$

Example 1: If max_behind = 0.02 and average spread = $0.05, then:

At 0.02 behind: $\frac{0.02}{0.02} \times \frac{0.05}{2} = 0.025$

Example 2: If max_behind = 0.02 and average spread = $0.05, then:

At 0.01 behind: $\frac{0.01}{0.02} \times \frac{0.05}{2} = 0.0125$

VWAP Fair Value Calculation

$$FV_j = midpoint_j + b_j F_{schedule}$$

Where:

F_k = factors for each signal or adjustment

b_{jk}= stock *j's* sensitivity to factor F_k

Example 1 continued:

if bid = 100, offer = 100.05, max_behind = 0.02 and avg_spread = 0.05

midpoint = 100.025

$F_{schedule}$ = 0.025

FV = 100.025 + 0.025 = **100.05** **:** this is >= offer price, so aggress

What happens if current spread is < or > average spread??

VWAP Order Placement

$$FV_j = midpoint_j + b_j F_{schedule}$$

It's time to place our order. Even though our FV should now reflect our "behindness", we still may not necessarily want to place the order at that price in call cases.

Why?

Simple placement logic for order_side = BUY:

```
if FV >= offer:
    order_price = offer
else # FV is not sufficiently high to warrant an aggressive order
    order_price = bid
```

Next Week

- Factor – based algorithm – VWAP 2
- Order State Management
- Other factors for execution
- Lee-Ready, MACD, etc.
- Possibly time-series analysis as well.

Questions?