A Framework for Historical Simulation of Trading Behavior

Allison Bishop*

Abstract

"Back-testing" of trading strategies in US equities often comes with the caveat that "transaction costs" are not considered. This may appear to be a fundamental limitation of working with historical data: how can we know what would have happened in an interactive system if some party had behaved differently? Trying to reason about this can feel at first like running up against the Heisenberg uncertainty principle: to interact with the market is to change it, so how can we disentangle the effects of interactions? But this ignores a few favorable conditions: first the kinds of actions that market participants take are classifiable and recognizable, and the list of typical actions is not that long. Second, the data is vast, with many data points across many stocks over many days. What this means is that for most actions we might consider taking as traders under most conditions, there are likely many identifiable examples of past traders taking similar actions in similar conditions. Here, we'll begin to develop a theory of how to turn these rich datasets into robust and actionable insights about trading dynamics.

Introduction

In the context of algorithmic trading, we expect the phrase "back-testing" to mean:"I've never used this code for live trading before, but I'm confident I know what's likely to happen because math." To break this down a bit further, let's dig deeper into the nature of the "code" and of the "math."

From conception to execution, a trade is driven by multiple layers of decisions: what and when to trade at a high level, and where/how/when to trade at a low level. The relevant timescales for higher level decisions span days, months, and perhaps even years, while the relevant timescales at the lower levels are typically hours, minutes, and below.

A high level trading algorithm is basically a set of rules that define the conditions under which an action should be taken. Its initial formulation may stem from a human-relatable hypothesis. For example, you might hypothesize that a stock whose price trends downward on one day is more likely to trend upward the next (perhaps downward price corrections tend to be over-corrections that themselves require corrections). We might turn this hypothesis into pseudocode by making some specific, quantitative choices: when we see a stock price drop by more than 10% during a trading day (as compared to its opening price), then we try to buy 100 shares of it. We hold until the end of the next day and then try to sell in the closing auction.

This kind of high level strategy is amenable to testing with historical data. We can take, say, a year's worth of data about opening, closing, and intraday stock prices and identify each time we would buy 100 shares according to our criteria. However, we immediately face some difficulties:

1. How do we know what exact price we would have paid for each 100 shares?

*Proof Trading, allison@prooftrading.com

2. How do we know what exact price we would have gotten each time we sold 100 shares?

3. How can we check that our dataset is complete? (e.g. includes stocks that appeared or disappeared during our data collection window)

Questions 1 and 2 are potentially challenging because they involve guessing what would have happened if we had interacted with the live financial system at various times when we did not. We can look at the prices that others obtained in the market around those times and assume our prices would have been similar. This assumption starts to be less and less defensible if we want to buy/sell larger quantities of stock at moments when it might be hard to buy/sell. At the modest size of 100 shares, it's probably quite reasonable to assume we could have participated in the closing auction, for instance, without significantly affecting the outcome of that auction. However, if we wanted to know what would have happened if we'd tried to sell in the closing auction when our shares would have represented a significant fraction of the total volume transacted in that auction, it would be less reasonable to suppose we could do so without impacting the price.

Addressing question 3 is by no means trivial - we must be extremely vigilant to route out any sources of sample bias in our data set in order to draw accurate conclusions about how our strategy would have performed. There is naturally no way we can protect against the possibility that the future may behave differently from the past. But there are many ways we can fail to test properly even if the future does behave nearly identically to the past, and we must be careful to avoid those.

Overall though, if we stick to buying and selling relatively modest amounts here, on a time scale of days or longer, and we get a full and accurate dataset for a time period of stock market behavior that is similar to today, we are likely to come out of this process with a reasonable estimate for how our trading strategy will perform. Any systemic differences between the assumed prices and the real prices we obtain in live trading are hopefully small in comparison to the high level effects we are trying to monetize, and we can monitor how closely our live trading conforms to our expectations. This is exactly the kind of scenario where "back-testing" with historical data is primed to succeed.

Of course, most institutional trading does not occur in small 100 share increments. A high level trading algorithm at an institutional firm will more likely decide to trade an amount that represents say $0.1\% - 10\%$ of the average daily trading volume in a given stock. This kind of amount cannot typically be traded instantaneously, and slicing it up into increments that are sprayed thoughtlessly across the market can tip off other market participants that there is a big buyer (or seller) present. This may cause the price to change unnecessarily to the detriment of the institutional trader.

Now let's leave the higher layer of trading decisions for a moment and consider a lower layer. Let's take as a given that we want to trade a large number of shares over a timespan on the order of hours or minutes. We need to decide how to spread out our activity over time and space, and every choice we make to say, send a small order to a particular venue at a particular time, can impact the market conditions we'll face as we navigate the rest of our trading.

It would be useful to subject our choices at this layer to something like "back-testing." If we could use historical data to see what *would have happened* if we had used algorithm A or if we had used algorithm B, then we can improve our decisions without needing to run live experiments, and getting enough data is not too hard because rich historical data is readily available. However, the challenges that arose when designing a back-testing framework for higher level trading decisions are greatly exacerbated at the lower levels, where the prices we will achieve for our individual trades are heavily influenced by our prior activity. In this realm, it is not reasonable to simply take the trading that happened absent our proposed activity as a

proxy for the prices we would have achieved.

The Alternatives

So what can we do in the face of this difficulty? One typical path is to forgo back-testing and instead rely solely on live "AB testing": in an AB test, we start with two different algorithms for trading large orders, and one thing we want to measure, like slippage versus arrival. Tracking "slippage versus arrival" means that we examine the difference between the best price being advertised for the stock at the moment just before we started trading to the actual prices we obtained for our trades. Then we flip a fair coin to decide which algorithm to use each time we have a large order to trade, and we compute the desired metric each time. After collecting a lot of data, we look at the measurements for the times we used each algorithm, and see if one algorithm performed better in our metric on average.

This a perfectly satisfactory and rigorous scientific approach, but there are a few major limitations:

1. This is not a particularly fast way to arrive at a near-optimal algorithm. It doesn't tell us how to design good pairs of algorithms to test, so we may be searching somewhat blindly through the big universe of reasonable-seeming algorithms, just taking two at a time.

2. It takes a lot of experimental data to make each test sufficiently conclusive. It may be hard to understand just how much, as our orders might vary significantly and these natural biases in the two samples may take a long time to average out. Also, the more tests we do, the more chance we have of getting some spurious results, and the more careful we need to be in interpreting the outcomes.

The choice of metric that is used to ultimately select the "better" of algorithms A and B is also fraught with peril. Slippage versus arrival is a reasonable choice, but as it is not normalized to market conditions and trends that may be in progress at the moment of arrival, it likely requires a very large data set in order to average out all such noise in order to draw reliable conclusions. To better account for current market conditions, some people choose slippage vs. VWAP as their metric instead. This compares the prices obtained on trades to the volume-weighted average price obtained by all traders in the market for that symbol over the same period of time. But this introduces a circularity, because the VWAP is calculated over the interval of activity. Hence poor trading decisions will influence *both* the prices attained by the algorithm *and* the prices attained by others that comprise the VWAP. Slippage versus a circular metric like this won't capture the full impact of the algorithm's decisions.

If we want to do more than AB testing, another path is theoretical modeling and simulation. We might try, for example, to understand a few kinds of participants that exist in the market, build code to behave like each of these participants, and then run a simulation where they interact each other. We might learn some interesting things from this, but there are a lot of pitfalls that will likely limit the amount of actionable insights we can gain from this approach. We are unlikely to capture a truly representative sample of the kinds of participants in the market for our simulation, and we are unlikely to simulate real market conditions faithfully enough at this level of abstraction.

Our Approach

Instead, we will try the path of *historical simulation*: we will aim to understand the likely effect of a given action our algorithm may take by matching the current market conditions to a

rich data set of past moments with similar market conditions. To perform this matching, we'll condense the rather amorphous construct of "market conditions" to a few discrete features that we can measure at any moment during a trading day. We'll refer to the combined values of these features as a *market profile.*

Once we have assembled a large (and hopefully representative) set of past data points matching the current market profile, we can start to form clear picture of what we expect to happen next if we *don't act.* We can empirically measure the probability of any particular future event or the distribution of any particular metric over a forward-looking window by using our rich set of matching historical data.

Next we want to understand what is likely to happen if we *do act.* For simplicity for now, let's suppose we are considering an action that is close to atomic (e.g. taking all of the available liquidity at one side of the top of the book over a short period of time). Now we can look at our matching data points and select the subset of them where our proposed action was taken by someone (or by some combination of actors). This will be a much smaller data set, but our hope is that it is still large enough to represent a meaningful sample size. By comparing the behavior of this sample conditioned on the action to the behavior of the more general sample, we can get a sense of the action's impact. For example, maybe the distribution of the VWAP in the market measured 5 to 10 minutes after the action reverts to roughly the same as in the general sample. This might suggest that the action itself exerts an influence that is limited in time to the next 5 minutes. (We would naturally check whether this behavior is similar for other metrics.)

Things get more complicated if we want to study the cumulative effects of a sequence of actions. If we expect the individual actions to be separated enough in time for the impact of each to subside before the next (again, something we can measure by seeing how far out in time the smaller sample distribution takes to converge to the general distribution), then it is reasonable to say the cumulative effects will simply be a sum of the individual effects. However, we may intend to be more active in the market than this pace of dissipation would allow. Also, some more drastics actions may have effects that fail to fully disapate within a relevant time horizon.

To get a sense of the behavior of longer sequences of actions that occur faster than dissipation effects, we might try further sub-sampling our historical data to find matches to the whole sequence of actions we are considering. This, however, is likely to make our matching sample size very small very quickly. And even if we can collect enough data to make this possible, the computational overhead of storing and sifting through it all will be quite prohibitive.

A better approach may be to try to understand the effect of each action on the *distribution of market profiles* at the time just preceding the next action. This is a generalization of the approximate independence we would get when the actions are separated enough to have their individual effects subside. If we can understand how a single action will effect the distribution of market profiles at the next action decision point, we can reduce the problem of simulating a sequence of actions to the problem of understanding the effects of a single action under a *distribution* of descriptive market profiles instead of under a single market profile. This is itself, of course, just a weighted computation on individual market profiles. All of this is promising because we then never have to slice our data thinner than the units prescribed by the combination of a single market profile and a single action.

This does not exempt us, however, from concerns that stringing together too many empirical approximations will eventually lead to unacceptable degradation of our results. The level of noise we measure in our individual distributions for single market profiles and single actions will have to guide us in determining what sequences of actions we can meaningfully simulate.

At every stage of this research process, we will seek to design clear and testable criteria to

guide our choices, so that we can get a good sense of how our individual choices will shape our ultimate outcomes. It is best to do this incrementally to the extent possible, just as it is best to test modules of software incrementally. If we wait until we've made a bunch of low-level decisions and then only measure the end result, we'll wind up with a mess of levers to pull and no real understanding of which ones are the most important. This would ultimately cost us valuable time, as well as burning through valuable data that we could not further use for independent tests.

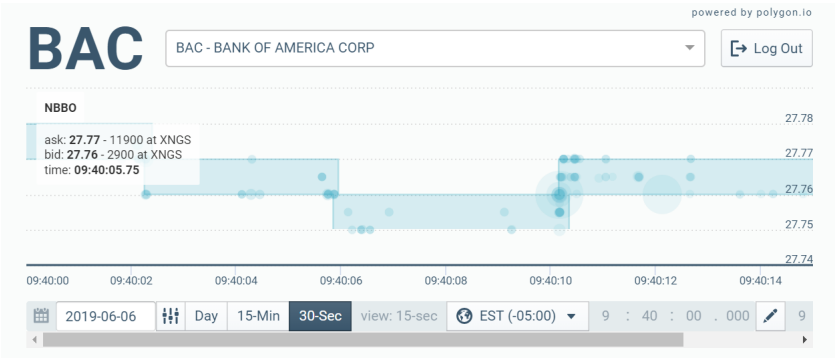
The Market Data

There are two major types of market data that will be most relevant for our purposes: quotes and trades. Quotes contain fields for price, size (how many shares), side (buy/sell), venue (which stock exchange), and time. Depending on the data source, we may have information only on quotes that represent the top of the book (i.e. the highest prices buyers are willing to advertise, and the lowest prices sellers are willing to advertise), or we may have full depth of book (all of the advertised prices).

Trades contain fields for price, size, venue, and time. The identities of the parties involved in the trade are typically not included. The venue field may group many possibilties together, such as only signifying that a trade occured off-exchange without specifying precisely where.

The inter-stitching of trades and quotes (if the timestamps are compatible to a sufficient granularity) can be used to enrich the direct information available about trades with reasonably inferred fields. For example, we can infer which trades occurred at the prevailing midpoint, prevailing bid (the highest price a buyer is willing to advertise), or prevailing offer (the lowest price a seller is willing to advertise). We can also infer which trades were closely followed by changes to the prevailing bid/offer.

Visualizing market data at this level of granularity can help ground our thinking about models and computation on top of it. Our "find my fills" tool, for example, visualizes trades as discrete events along a continuous horizontal axis of time, with the vertical axis representing price. The size of a trade is indicated by the size of the corresponding circle, and the prices are shown in context of the prevailing NBBO (National Best Bid and Offer):



Defining Market Profiles

Before we consider distributions over market profiles, we must first consider possible definitions of market profiles and provide some rationale for choosing between them. Market profiles form

the "points" in our probability space, and our definition of them crucially impacts all of our subsequent analysis.

A market profile will be specific to a particular symbol, though it may include features that capture what is happening more broadly across symbols. A useful market profile should tell us something nontrivial about trading activity in the recent past, and have some predictive power over trading activity in the near future.

At one extreme of granularity, we could choose the full time series of trades and/or quotes over a given time window as our definition of a market profile. This would certainly contain useful trading information and have predictive power (to the extent that predictive power from our time series data exists). However, if we tried to find matches to historical data at this extreme level of granularity, we would be unlikely to find any. Every little segment of time in each stock would behave slightly differently, in meaningless ways that would obscure coarser patterns. Thus, the set of all possible time series of trades and/or quotes over given time intervals is not an effective choice for the probability space underlying our analysis.

Instead, we will need to identify coarser features of time series segments to be our base units of measurement and prediction. This is the part that is often more art than science, and requires a human intuition to identify what really matters about the granular data and what really doesn't. Our primary dimensions here are time, price, and size, so our first task is to identify reasonably coarse units for each of these, hopefully without sacrificing too much meaning.

It's worth taking a moment to think through the various contradictory forces that we must reconcile and how we might try to confirm that our choices are striking the balances we intend. For our descriptive measurements, we want to choose units that are fine-grained enough to differentiate between fundamentally different market behavior. But we also need them to be coarse enough to provide large datasets of samples matching each common profile.

Size and price are probably the simpler dimensions to get a handle on. Whether we are considering trades or quotes, looking at individual events or aggregated over a particular time window, we can likely bucket "size" into a few categories, e.g. "small," "medium," or "large." The exact boundaries of these categories might be absolute or relative to averages, and could be universal or fit specifically to an individual stock. Similarly for price, we might bucket absolute differences in prices or relative differences, for individual events or volume-weighted averages, etc. To some extent, price and size lie on a smooth(-ish) continuum: yes, they are inherently discrete in implementation across the market, but over the course of minutes, hours, etc., the differences between a 200-share trade and a 300-share trade, or between a trade at $50 and a trade at $50.01 probably don't matter overly much.

Time, however, may be a different animal. Trading activity often has inflection points in time, and the durations of bursts and trends may be highly variable. If we simply segment time according to fixed increments (e.g. 1-second or 1-minute buckets), we may lose a lot of meaning in the mismatch between our units and the true patterns of activity. We can try to avoid this by defining windows of time in an event-driven way. For example, we might mark the edges of our windows based on inflection points in trading or quoting activity.

Putting this all together, we can define features that capture various coarse characteristics of trading activity in a given symbol over the recent past. For example, we might use an event-driven definition of a backwards-looking time window, and measure the total volume of trading in that window and the approximate slope of a best fitting line to midpoint of the NBBO[1] over that window. We may also include features that capture similar coarse characteristics of trading activity more broadly (e.g. the same features for SPY at that time).

[1]NBBO is an abbreviation for the "National Best Bid and Offer": the highest price advertised by buyers and the lowest price advertised by sellers across all exchanges. The midpoint is the average of these two prices.

We will judge possible candidates for market profiles according to two criterion that are fundamentally at odds: density of coverage and predictive power. These features can only be defined relative to a probability distribution or data set, so we'll detour for a moment to consider the formation of our data set and then we'll come back to define these criterion.

Probability Distributions over Market Profiles

Once we select a set of features as a reasonable candidate for a market profile, we have a new decision to make: how do we take a historical time series and process it into a sequence of market profiles? Probably the features in our market profile can be computed at *any* point in time during a trading day, but how should we choose what points to sample?

One choice is to sample a new market profile according to a fixed time interval, e.g. every 2 minutes. This may not be a great idea, as our fixed intervals will often mis-align with bursts of market activity.

Another choice is to sample a new market profile at every new trade/quote event. This also might not be an ideal approach, as adjacent market profiles are likely to have highly correlated features, and the bursty nature of events might skew our data samples in unintuitive ways.

A third choice is to somehow split the difference: to do event-driven sampling, but perhaps only resample after a minimum window of time has passed.

For now, we'll lean toward sampling a new market profile at every new market event, just because we'd like to err on the side of starting with the largest potential data set unless there is a compelling reason to do otherwise. We should carefully keep in mind though what this means - that the points in our probability space are mapped one-to-one onto market events. If we equally weight these points, we are therefore weighting the importance of bursts of market activity proportionally to the number of underlying events.

Fixing this sampling strategy, as well as a range of time and symbols, etc., we can collect a fixed historical dataset that will serve as our base "probability space" over market profiles. When we condition on matching a particular market profile, we will take only the points in this space that match that profile, and view them as a new probability space. As we do any computations, we will keep points in data structures that maintain important context like symbol, date, and precise timestamp (even though these are not likely to be part of the market profiles themselves).

We will need this context to answer questions like, "conditioned on matching a particular market profile now, what's the distribution of the change in price over the next 5 minutes?" For this, we would need to find the subset of data points that match our target market profile, and then look forward from their corresponding symbols/times to compute the relevant metric.

Once we have defined our probability space over market profiles (presuming equal weighting of our collected samples for now), we can define and measure "density of coverage" and "predictive power." Density of coverage will consider how many market profiles we need to capture a reasonable portion of the probability space: e.g. how concentrated our probability distribution is. If we imagine sorting all of our market profiles from the most common to the least common (e.g. highest probability mass to lowest probability mass), we can start adding up the mass for the most common profiles first and seeing how many profiles we have to add before we reach a certain target total mass. The fewer profiles it takes to cover a good portion of the total probability mass, the better it is for having healthy-sized data sets that match common profiles.

The counterbalancing concern is predictive power, which will consider how *different* the profiles are in terms of future behavior. For an individual market profile, we can pick a target metric to measure at various time scales after each occurrence of that market profile. We can then compare this to doing the same measurement more generally across our entire dataset.

We would expect that at a long enough time scale, the differences in behavior between the two datasets will dissolve to nothing. Predictive power is higher if the initial difference in behavior is more pronounced and/or persists for longer.

We should also do these calculations to compare market profiles to each other: if different profiles produce similar behavior on the metrics we care about on the time scales we care about, then we should essentially merge them to reduce unnecessary sharding of our dataset. We'll probably want to do this not by merging individual pairs of market profiles in an ad hoc fashion, but by trying to generalize our examples into broader implications for feature selection that lead to us redefining the building blocks of market profiles in order to effectively achieve such merges while keeping our features relatively simple.

This is a typical instantiation of a clustering problem in unsupervised machine learning. In k-means clustering, the goal is to split the data into k clusters such that behavior on a metric of interest within each cluster is similar, whereas behavior across clusters is not. Density of coverage is our way of keeping the value of k to stay effectively small, while predictive power is our version of this clustering goal.

Analyzing Activity

To study the likely impact of a particular action, we need to identify moments in our dataset where such an action was taken. The task of defining each action bears some resemblance to the task of defining a market profile. We want to choose a definition for an action that is sufficiently broad so that we get many matches in our data set, but not so broad that we are grouping together situations that behave fundamentally differently.

One action we might identify is: taking all of the liquidity at the top of the book at once or within a very short window of time. Another action we might identify is taking *more* than the top of the book is quick succession: e.g. blasting through a few levels of depth. As will always be the case, we won't be sure that this activity is carried out by a single actor, as we won't have the identities of the parties to each transaction. This is unlikely to matter too much: the rest of the market is in the dark about the identities too, so in some sense the market's reaction to the activity should be (mostly) independent of whether it is a single actor or not. This is not perfectly true because the parties to each trade will know their own identities of course, and other nuances such as what order types they used, etc.

Once we have a candidate definition of an action, we can search through our historical data for occurrences of that action. We can also take note of the market profiles for each moment preceding an occurrence. If we don't find a healthy number of occurrences matching common market profiles, then we will need to revisit our action definition to make it more general, or we will need to revisit our market profile definition, or both.

At a point where we do have a healthy sample of occurrences of a particular action under each of a fairly comprehensive set of market profiles, we are ready to study the impact of the action. For this, we can look at the market profiles that are produced subsequent to the action, and track the distribution of market profiles as we proceed further out in time/subsequent market events. We can compare this to the same calculation for our larger dataset of moments that match the same initial market profile, but are not conditioned on matching the action. We would expect that these two series of distributions would eventually converge (as we would expect both eventually converge to the generic distribution independent of even the initial market profile over a long enough time scale). The speed and nature of the convergence of distributions for these two data sets (the one matching the initial market profile and action vs. the one just matching the initial market profile) gives us insight into the probable impact of our action under those initial market conditions.

One important challenge will be incorporating the submission of resting orders as an action in this framework, as there is a variable separation between the act of submitting the order and the resulting trades, if any. One clean way of dealing with this might be to define the action as the resulting trade(s), and to think of the actor's decision as sampling a random variable that produces the action with a certain probability distribution over subsequent times.

Recursion

If we next want to understand the probable impact of a short sequence of actions, we first need to specify how the sequence is constructed. For example, do we wait a specified amount of time after the first action and then take the second? Or do we take the first action and then wait for some criterion over market conditions to be satisfied before taking the second?

Let's first consider the case of a fixed time increment. We can start by matching the desired initial market profile and action, and then looking at our projected distribution of likely market profiles at the specified future time increment. For each reasonably likely market profile we get, we can do the analogous computation for the second action.

Basically this sets up a natural recursion: the probable impact of a sequence of actions can be reduced to the problem of understanding the impact for each common combination of a single action under a single initial market profile. If we keep our set of market profiles and our set of actions small, we can hope to precompute these impacts at various time scales and merely assemble them into the desired sequences. Naturally, the more actions we string together, the noisier our computations will become, until they become no more predictive than the generic, unconditioned distribution. We should be careful to monitor this effect so that we are well aware of when our action sequence simulation results are meaningful, and when they are converging to meaningless.

Going from Simulation to Algorithm Development

Let's suppose (quite optimistically!) that all of this works, and we can get meaningful information about the likely effects of individual actions and short sequences of actions that we might want to perform while executing an order on behalf of an institutional client. How can we leverage this to design better trading algorithms? One way is to design various possible algorithms and then test them against each other via simulating the sequences of actions they produce. But it might be sub-optimal to design full algorithms in a vacuum and then only apply this simulation-style knowledge to the testing phase.

Instead, algorithm design problems can be set up as recursions in a framework similar to what we've discussed above. We start with a goal like: buy X shares of stock Y over the next T minutes/hours/days, etc. Our goal is to minimize the cumulative price that we will pay for the stock. Suppose the current market profile in this stock is M. We can define the function F to represent the amount we expect to ultimately pay if we behave optimally, and this is a function of X, Y, T, and M:

$$F(X, Y, T, M) := \text{ expected cost to buy } X \text{ shares of } Y \text{ over time } T, \text{ initial market profile} = M$$

Suppose A represents an action we can choose to take right away (e.g. buy all of the shares currently available at the NBO immediately). If we take action A right now, let's say we'll acquire Z shares, and then suppose we wait t minutes before doing anything else. Let P be the cumulative price of the Z shares we would buy now. This gives us an inequality:

$$F(X, Y, T, M) \leq P + F(X - Z, Y, T - t, M'),$$

where M' is no longer a single value, but is the distribution over market profiles that we expect t minutes after performing action A with initial market profile M.

The goal would then be to use these kind of recursive inequalities to derive an optimal or near-optimal sequence of our available actions. This looks like a dynamic programming problem because the right hand subproblem in the recursion above always has a smaller time or remaining share count. So this should be solvable by computing all the optimal values for goals that can be attained in single actions and then working up to multiple action sequences. It is important to note that our "optimal" decisions here are only optimal relative to our definitions of the available actions. If there is a better action that is simply not on our list of studied actions, this method will not find it. It will simply find the best sequences of the actions we've specified, assuming the sequences do not suffer from too much degradation of our predictive power due to noise in the data.

Discussion

It would be customary at this point in a whitepaper to have a section titled "Conclusion," and for us to wax poetic about how the data has or will ultimately validate our approach. Frankly, this can be asinine. To be clear, real data is the foundation of any responsible development of theory. But people who insist "the data will prove me right" are often trying very hard to shovel words into the data's mouth. It's like when a friend says "some people are saying..." when he just does not want to assume responsibility for his own opinions.

Theories are (and should be!) shaped by data. Otherwise they are never forced to confront reality. And how we interact with and intrepet data is (and must be) shaped by theory. Otherwise there is no structure to our exploration, and we cannot build understanding without structure. This is a more delicate and bi-directional process than we often like to admit. It may strike some as premature or even irresponsible for us to put forth the theoretical framework above without yet validating it on real data. But we take the opposite view. This exact moment, when we have forced ourselves to clarify our thinking sufficiently to write it down but we have not yet fully coded our first round of tests and run them on real data, is the best time to preserve our initial state of mind. This way as our research progresses, we can come back to the document as a starting point, and track how our thinking has evolved over time. Continual documentation of our research framework, including why and how we make changes to it as we interact with real data, gives us a stronger basis for insight and accountability.

And so this is not a "Conclusion." It is a starting point. We will be sharing the insights we learn along the way as our research progresses.