Algorithmic Finance 2 (2013) 151–165
DOI 10.3233/AF-13023
IOS Press

Discovering the ecosystem of an electronic financial market with a dynamic machine-learning method*

Shawn Mankad[†‡], George Michailidis[‡], and Andrei Kirilenko[‡‡]
[‡]*Department of Statistics, University of Michigan, Michigan, USA*
[‡‡]*MIT Sloan School of Management, Cambridge, MA, USA*

Abstract. Not long ago securities were traded by human traders in face-to-face markets. The ecosystem of an open outcry market was well-known, visible to a human eye, and rigidly prescribed. Now trading is increasingly done in anonymous electronic markets where traders do not have designated functions or mandatory roles. In fact, the traders themselves have been replaced by algorithms (machines) operating with little or no human oversight. While the process of electronic trading is not visible to a human eye, machine-learning methods have been developed to recognize persistent patterns in the data. In this study, we develop a dynamic machine-learning method that designates traders in an anonymous electronic market into five persistent categories: high frequency traders, market makers, opportunistic traders, fundamental traders, and small traders. Our method extends a plaid clustering technique with a smoothing framework that filters out transient patterns. The method is fast, robust, and suitable for a discovering trading ecosystems in a large number of electronic markets.

Keywords: trading strategies, high frequency trading, machine learning, clustering

1. Introduction

The words "stock market", "futures market" or "trading pit" used to elicit a mental picture of a chaotic crowd of agitated people wearing brightly-colored jackets, gesticulating wildly and shouting at each other. Yet, a trained human eye would see a great deal of structure behind this frenzy. Some of the people were market makers who stood at certain posts and "made markets" in securities or derivatives that were designated only to them. Some were floor brokers who formed circles around the market makers to get the best prices for a broad range of their customers – from pension funds investing their assets, to banks hedging exposures on their balance sheets. Others were different types of floor traders from scalpers to spreaders to opportunistic position takers, who wandered around the floor looking for opportunities to exploit. The ecosystem of an open outcry market was well-known, visible to a human eye, and rigidly prescribed: traders had designated functions, used common gestures to trade, wore jackets of certain colors, and could be found in specific locations on a trading floor.

The transition to anonymous electronic trading has obfuscated the prescribed ecosystem of roles, relationships, and designations previously clearly visible on a trading floor. Trading floors have been replaced by server farms, prescribed gestures have been replaced by message protocols, and the traders themselves have been replaced by algorithms often operating with little or no human oversight.

While the process of electronic trading is not visible to a human eye, machine-learning methods have been

*The views expressed in this paper are our own and do not constitute an official position of any agency, its management or staff.

[†]Corresponding author: Electronic address: smankad@umich.edu; Fax Number: 734-763-4676; Phone Number: 734-763-3519 Mailing Address: University of Michigan; 269 West Hall, 1085 South University; Ann Arbor, MI 48109-1107

developed to recognize persistent patterns in the data. Even without a formal, regulatory designation, a trader who follows a particular strategy would leave a distinct footprint in the data.

In this study, we present a novel machine-learning method to parse through the footprints of all traders in a highly liquid, anonymous electronic market and find certain common "paths" that they follow, thus, describing the roles and functions of participants who inhabit the new ecosystem of an electronic financial market.

Our method combines a static plaid clustering technique with a dynamic smoothing framework that filters out transient patterns. The plaid clustering technique - a regression-based method to describe empirical regularities in cross-sectional data - was previously used only for a single, static data matrix. Our method extends the plaid model by making use of a time series of data matrices. Our extension, which we refer to as the smooth plaid model, is able to consistently identify categories of traders and trading outcomes that persist through time.

We utilize synthetic data generated from an agent based model (Paddrik et al., 2011; Hayes et al., 2012) that is calibrated on actual E-mini 500 stock index futures contract (E-mini) data made available to us by the CFTC. We originally employed our method on regulatory, transaction-level data for the E-mini - the price discovery vehicle for the broad U.S. stock market. However, due to a Chicago Mercantile Exchange (CME) complaint, there has been a hold by the CFTC on the use and reporting of any regulatory data by the academic community. Using synthetic data allows us to make public our machine-learning methodology and results without concerns associated with the extreme confidentiality of regulatory data. The more methods that are available in the literature for analysis of electronic trading data, the more rigorous the discussion on the role and consequences of electronic markets. We note that results of applying our method on actual regulatory data are broadly similar. Moreover, we were able to designate traders into the same categories that were recovered manually by Kirilenko et al. (2010) in their analysis of the Flash Crash using similar data.

Our results are as follows. Using the smooth plaid model, we assign 6387 traders in the simulated data into five distinct categories: high frequency traders (7 traders), market makers (73), opportunistic traders (2405), fundamental buyers and sellers (1281), and small traders (2849). These traders occupy quite distinct, albeit sometimes overlapping, positions in the ecosystem of the market. High frequency traders, whose data footprint sometimes resembles scalpers on steroids, occupy a very special position in the market. They trade through an enormous number of contracts each day, but carry very little inventory at any point in time. Market makers are in the market all the time; they quickly buy and sell on demand and manage their inventory very tightly. Fundamental traders accumulate directional inventory over long periods of time, often days, presumably to take a longer-term investment view or to hedge their other exposures. Opportunistic traders take on and manage directional bets for minutes or hours at a time, in search of opportunities to profit from the perceived imbalances. Small traders do not exhibit any persistent pattern; they enter the market very infrequently at seemingly random times and trade in trivial quantities.

We believe that the smooth plaid model in particular, and machine-learning methods, more generally, can be effectively used for the analysis of traders and their strategies in electronic financial markets. In an environment where traders do not have formal designations, the smooth plaid model forms a useful first step to separate tens of thousands of trading accounts into manageable trader categories for subsequent analysis - be it a market event like a Flash Crash[1], co-movement of asset prices[2], or the impact of trading strategies on market quality[3].

The paper proceeds as follows. In the next section, we briefly summarize the plaid model (2.1), and discuss our modifications to create the smooth plaid algorithm (2.2). In Section 3, we illustrate the proposed model using a simple set of simulated data. In Section 4, we apply our method to simulated data generated by an agent-based simulation model of an electronic market calibrated to the E-mini. We conclude with a discussion and review of this study (Section 5).

[1] See, Kirilenko et al. (2010). The authors separate their traders into categories manually. They arrive at similar categories as the ones presented in this study.

[2] See, Huang (2011) for an investigation of co-movement of exchange rates. The authors develop a variant of a machine-learning technique with a parametric way to deal with the time-series dimension.

[3] See, Chaboud et al. (2011) and Hendershott et al. (2010). The authors rely on designations given to them by a trading venue. They do not use a machine-learning method to cluster traders into categories.

2. Methods

2.1. Biclustering and the plaid model

Suppose we observe a data matrix $X \in \mathbb{R}^{n\times p}$, where X_{ij} represents the ith sample ($i = 1,\ldots,n$) and jth variable ($j = 1,\ldots,p$). The plaid model, first introduced by Lazzeroni and Owen (2000), aims to decompose the data to reveal the underlying structure. In our setting, the model is trained to discover groups of traders that have similar trading behaviors.

The term biclustering was first used by Cheng and Church (2000) to refer to grouping procedures appropriate when both the samples and variables are of scientific interest. In contrast, clustering methods belong to a closely related topic in machine learning and are concerned with discovering the structure of samples only. Hence, biclustering methods extract groups of samples (rows) and variables (columns) to find homogeneous submatrices in a static data matrix. These methods typically allow samples to be in more than one cluster, or in none at all. This flexibility is also given to variable groups, that is, variables can be defined with respect to only a subset of samples, not necessarily with respect to all of them. Moreover, these flexible models allow for overlapping biclusters.

In our application setting, samples are individual traders and the variables are measures of trading activity for each trader: trading volume, net position, change in inventory, trades per second, and median intertrade duration. A bicluster is then a group of traders and measurements of their trading activity that are similar. With respect to the biclustered variables, a trader is more similar to other traders in the same bicluster than traders outside of the bicluster.

Next, we introduce an important concept to the plaid model, namely that of an additive "layer". A layer is a canonical matrix matching the dimensions of the given data matrix, with zeros everywhere except the biclustered elements. In the plaid model, the data is decomposed into a series of additive layers that capture the underlying structure of the data. As a consequence, layers combine to provide a reconstruction that highlights the main features of the given data matrix.

The plaid model first includes a background layer that consists of *all* traders and variables to account for global effects in the data. In our application setting, the background layer accounts for market trends that affect trading behavior of all traders, such as for example, a major liquidity event. There are in principle many ways to construct the background layer. The simplest approach is to set each element of the background layer to be equal to the global average of the given data matrix. One could also estimate a parametric model that incorporates a priori information about the traders and variables. In our analysis, we set each column of the background layer to the corresponding variable's mean. This is equivalent to standardizing the data, which is necessary since a variable like volume is strictly non-negative, while others like net position can be negative. Subsequent layers represent additional effects corresponding to specific traders and variables that exhibit a strong pattern not explained by the background layer.

Formally the data matrix $X \in \mathbb{R}^{n\times p}$ can be represented as

$$X_{ij} = \mu_0 + \sum_{k=1}^{K} \theta_{ijk} r_{ik} c_{jk}, \tag{1}$$

where $i = 1,\ldots,n$ indexes samples and $j = 1,\ldots,p$ indexes variables, μ_0 captures the background layer and θ_{ijk} describes the bicluster effect; k is a layer index running to the number of biclusters K. The parameters r_{ik} and c_{jk} are indicator variables that combine to identify the bicluster, that is, they denote bicluster membership for, respectively, the traders and variables.

There are several modeling choices for the form of θ_{ijk}, the most common being

$$\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}, \tag{2}$$

where $i = 1,\ldots,n$, $j = 1,\ldots,p$ and $k = 1,\ldots,K$. Each bicluster has a mean, trader, and variable effect. Hence, each bicluster is expressed as a two-way analysis of variance (ANOVA) model. In other words, each trader in a bicluster can be interpreted as following similar strategies (μ_k). Yet, traders in the bicluster may differ slightly due to differences in preference, amount of available capital, and so on. This trader-specific effect is captured by the effect $\{\alpha_{ik}\}$. Similarly, biclustered measures of trading activity can differ by trading strategies $\{\beta_{jk}\}$.

The biclusters are discovered in a sequential fashion. Suppose $K-1$ layers have been estimated in addition to the background layer. The residual data matrix is given by

$$\hat{Z}_{ij} = X_{ij} - \hat{\mu}_0 - \sum_{k=1}^{K-1} \hat{\theta}_{ijk} \hat{r}_{ik} \hat{c}_{jk}. \tag{3}$$

The Kth bicluster is found by minimizing the usual residual sum of squares over all parameters of interest

$$\min_{\{\theta_{ijK},r_{iK},c_{jK}\}} \sum_{i=1}^{n}\sum_{j=1}^{p}(\hat{Z}_{ij} - \theta_{ijK}r_{iK}c_{jK})^2. \quad (4)$$

Estimates of the bicluster memberships ($\hat{r}_{iK}$,$\hat{c}_{jK}$) are obtained with a numerical search. A simple search procedure based on k-means clustering (see Hastie et al., 2001) is presented throughout this paper. More complex strategies and comprehensive discussion can be found in Lazzeroni and Owen (2000) and Turner et al. (2005). When given bicluster memberships, estimates of the bicluster-specific effects ($\hat{\theta}_{ijK}$) are easy to compute, as one can use the usual two-way ANOVA estimators (Turner et al., 2005).

The plaid model estimates the behavior of each trader and then seeks groups of traders that have similar behavior over the biclustered variables. The estimation procedure is an iterative one based on minimizing sum of squares of the data minus estimated layers (pseudocode is given in Algorithm 1). First the background layer is estimated, then bicluster-specific layers are added one at a time. The statistical significance of layers are determined by a permutation test. The algorithm terminates when a significant layer cannot be found.

Next, we provide a brief review of the permutation test discussed starting on page 8 of Lazzeroni and Owen (2000). A comprehensive review can also be found in Turner et al. (2005). The permutation test is intuitively similar to bootstrapping, and relies on resampling of the data to approximate significance of the bicluster. The basic idea is that the data values are independent of biclusters after permuting the rows and columns. Thus, comparing the candidate bicluster against (noise) biclusters obtained after randomizing the data matrix allows one to accept a bicluster only if it is significantly larger than what one would find in noise.

The importance of each bicluster is measured with

$$\sigma_k^2 = \sum_{i=1}^{n}\sum_{j=1}^{p}\hat{r}_{ik}\hat{c}_{jk}\hat{\theta}_{ijk}^2, \quad (5)$$

where $k = 1,\ldots,K$. Let π_r be the permutation of the index set $\{1,\ldots,n\}$, and π_c be the permutation of the index set $\{1,\ldots,p\}$. Then $\tilde{Z}_l = \hat{Z}(\pi_r,\pi_c)$ is the matrix after permuting every row of the residual data matrix $\hat{Z}$ and then permuting every column of the result. The importance of a bicluster obtained from $\tilde{Z}_l$ is measured with

$$\hat{\sigma}_{n_l}^2 = \sum_{i=1}^{n}\sum_{j=1}^{p}\tilde{r}_{il}\tilde{c}_{jl}\tilde{\theta}_{ijl}^2, \quad (6)$$

where $\tilde{r}_{il},\tilde{c}_{jl}$ are bicluster memberships estimated from $\tilde{Z}_l$. The candidate bicluster is rejected if any of the noise biclusters are more important. The selection of the total number of noise biclusters L is discussed further in Section 2.3. The permutation test is given in Algorithm 3.

Next, we will discuss an extension of plaid models to detect persistent patterns when given a sequence of data matrices.

Algorithm 1 The plaid model estimation procedure for static data.

Input: Matrix $X \in \mathbb{R}^{n\times p}$

Output: Sequentially discovered biclusters $\{\hat{r}_{ik},\hat{c}_{jk},\hat{\theta}_{ijk}\}_{k=1}^{K}$

$\hat{\mu}_0 = \frac{1}{n}1_{n\times n}X$
$\hat{Z} = X - \hat{\mu}_0$
$K = 1$ (bicluster counter)
repeat
  $\{\hat{r}_{iK},\hat{c}_{jK},\hat{\theta}_{ijK}\} = estimateBicluster(\hat{Z})$ (see Algorithm 2)
  $b = permuteTest(\{\hat{r}_{iK},\hat{c}_{jK},\hat{\theta}_{ijK}\})$ (see Algorithm 3)
  if $b = 0$ **then**
    $\hat{Z}_{ij} = X_{ij} - \hat{\mu}_0 - \sum_{k=1}^{K}\hat{\theta}_{ijk}\hat{r}_{ik}\hat{c}_{jk}$
    $K = K + 1$
  end if
until $b = 1$
return $\{\hat{r}_{ik},\hat{c}_{jk},\hat{\theta}_{ijk}\}, i = 1,\ldots,n, j = 1,\ldots,p, k = 1,\ldots,K$

Algorithm 2 estimateBicluster

Input: Matrix $\hat{Z} \in \mathbb{R}^{n \times p}$
Output: Bicluster $\{\hat{r}_{iK}, \hat{c}_{jK}, \hat{\theta}_{ijK}\}$

Apply k-means (k=2) to rows of $\hat{Z}$. Set $\hat{r}_{iK}$ to the smaller cluster.
Apply k-means (k=2) to columns of $\hat{Z}$. Set $\hat{c}_{jK}$ to the smaller cluster.
repeat
  $\hat{\theta}_{ijK} = \text{argmin}_{\{\theta_{ijK}\}} \sum_{i=1}^{n} \sum_{j=1}^{p} (\hat{Z}_{ij} - \theta_{ijK} \hat{r}_{iK} \hat{c}_{jK})^2$
  for i=1,...,n **do**
    if $\sum_{j=1}^{p} (\hat{Z}_{ij} - \hat{\theta}_{ijK} \hat{c}_{jK})^2 < \sum_{j=1}^{p} \hat{Z}_{ij}^2$ **then**
      $\hat{r}_{iK} = 1$
    else
      $\hat{r}_{iK} = 0$
    end if
  end for
  for j=1,...,p **do**
    if $\sum_{i=1}^{n} (\hat{Z}_{ij} - \hat{\theta}_{ijK} \hat{r}_{iK})^2 < \sum_{i=1}^{n} \hat{Z}_{ij}^2$ **then**
      $\hat{c}_{jK} = 1$
    else
      $\hat{c}_{jK} = 0$
    end if
  end for
until $\hat{r}_{iK}, \hat{c}_{jK}$ converge or maximum iteration number attained
return $\{\hat{r}_{iK}, \hat{c}_{jK}, \hat{\theta}_{ijK}\}, i = 1, \ldots, n, j = 1, \ldots, p$

Algorithm 3 permuteTest

Input: Bicluster $\{\hat{r}_{iK}, \hat{c}_{jK}, \hat{\theta}_{ijK}\}$
Output: $\{0, 1\}$

$\sigma_K^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} \hat{r}_{iK} \hat{c}_{jK} \hat{\theta}_{ijK}^2$
$\hat{Z}_{ij} = X_{ij} - \hat{\mu}_0 - \sum_{k=1}^{K} \hat{\theta}_{ijk} \hat{r}_{ik} \hat{c}_{jk}$
for l=1,...,L **do**
  $\pi_r = permutation(\{1, \ldots, n\})$
  $\pi_c = permutation(\{1, \ldots, p\})$
  $\tilde{Z}_l = \hat{Z}(\pi_r, \pi_c)$
  $\tilde{r}_{il}, \tilde{c}_{jl}, \tilde{\theta}_{ijl} = estimateBicluster(\tilde{Z}_l)$
  $\hat{\sigma}_{n_l}^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} \tilde{r}_{il} \tilde{c}_{jl} \tilde{\theta}_{ijl}^2$
end for
if $\hat{\sigma}_K^2 > \max\{\hat{\sigma}_{n_1}^2, ..., \hat{\sigma}_{n_L}^2\}$ **then**
  return 0
else
  return 1
end if

2.2. *Smooth plaid models for multidimensional time-series*

Suppose we observe a time series of matrices with the rows consisting of individual traders and the columns representing various measures of their trading activity at different points in time. Formally, we have $\{X_{ij}^{(t)}\}_{t=1}^{T}$, where t is a time index and $i = 1, \ldots, n, j = 1, \ldots, p$. Since each row corresponds to a trader, we have a total of n traders that transact at least once in the data. For each trader, at each point in time, we observe the same p variables that measure different aspects of trader behavior. Then we can represent the data matrix $X^{(t)}$ at time t as

$$X_{ij}^{(t)} = \mu_0^{(t)} + \sum_{k=1}^{K} \theta_{ijk}^{(t)} r_{ik}^{(t)} c_{jk}^{(t)}, \tag{7}$$

where $t = 1, \ldots, T$. The expression and indicator parameters ($\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}$) reflect whether a bicluster is active in a given time period t. The total number of biclusters K is fixed for all time periods, since additionally allowing K to vary with time creates identifiability and implementation challenges.

Visually, the data can be organized as a three dimensional array, shown in Figure 1, with traders arranged in the rows, trading features in the columns, and time as the 3rd 'depth' dimension.

Note that a direct analysis may proceed by collapsing the temporal dimension and working with a

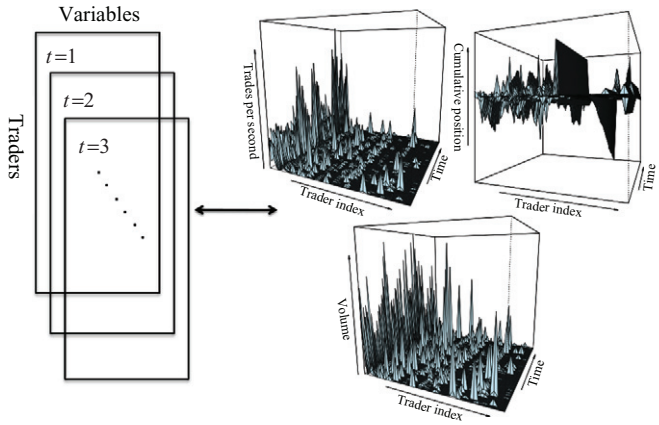

Fig. 1. For our study, we construct a series of matrices, one for each period of time, consisting of approximately six thousand rows (one for each trader) and several columns of trading measures. Once the data is organized, we are working with a three dimensional array with traders arranged in the rows, trading measures in the columns, and time as the 3rd dimension.

single two dimensional matrix with traders arranged in the rows and variables that have been combined over time in the columns. However, collapsing the time dimension would aggregate away a significant amount of valuable time-series information present in the data. To illustrate, Figure 2 shows the general structure that remains after integrating over time. As shown in the stylized plot[4] of net position vs. volume/number of trades, a distinct group of high frequency traders emerges holding a very small open position at the end of the trading day, together with fundamental traders holding large positive or negative positions. However, there is a very large number of significant traders that are not allocated to an interpretable group.

A potential remedy for this is to analyze each data set at each point in time separately. However, this would ignore the potentially important time component of trading strategies, while becoming dominated by transient patterns. Moreover, if we repeatedly and directly apply a method like the plaid model, the estimated groupings can change for each data set when a temporally stable structure is more appropriate.

In this study, we design and employ a dynamic method that is in between these two direct approaches. A penalized optimization framework accounts for auto-correlation by effectively averaging the groups over a rolling window of time. Such an approach helps mitigate the effects of transient patterns, while enhancing structural regularities in the data.

[4]Plots of aggregate data are used throughout this work to protect confidentiality.

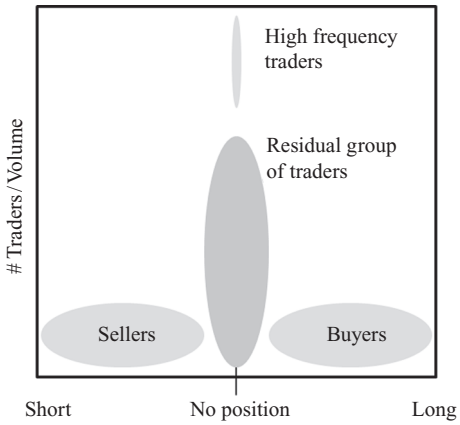

Fig. 2. The general structure that remains after collapsing the temporal dimension. High frequency traders, buyers, and sellers are prominent and easily detectable. However, there are a large number of residual traders that cannot be easily interpreted.

Formally, we consider the search for the Kth layer over the interval $t = T - W, \ldots, T$. The search is initialized with starting values for $\{\hat{r}_{iK}^{(t)}\}_{t=T-W}^{T}$ and $\{\hat{c}_{jK}^{(t)}\}_{t=T-W}^{T}$, which denote whether the candidate bicluster was detected in the previous W time periods. The objective function is given next.

$$\min_{\{\theta^{(T-W)},\ldots,\theta^{(T)}\}} \sum_{t=T-W}^{T}\sum_{i=1}^{n}\sum_{j=1}^{p}(\hat{Z}_{ij}^{(t)} - \theta_{ijK}^{(t)}\hat{r}_{iK}^{(t)}\hat{c}_{jK}^{(t)})^2 + \lambda \sum_{t=T-W+1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{p}(\theta_{ijK}^{(t)}\hat{r}_{iK}^{(t)}\hat{c}_{jK}^{(t)} - \theta_{ijK}^{(t-1)}\hat{r}_{iK}^{(t-1)}\hat{c}_{jK}^{(t-1)})^2, \quad (8)$$

where λ is a tuning parameter and W is a parameter determining the number of previous time periods to consider. Following the notation in Equation 3, $\hat{Z}^{(t)}$ is the residual data matrix at time t

$$\hat{Z}_{ij}^{(t)} = X_{ij}^{(t)} - \hat{\mu}_0^{(t)} - \sum_{k=1}^{K-1} \hat{\theta}_{ijk}^{(t)} \hat{r}_{ik}^{(t)} \hat{c}_{jk}^{(t)}, \tag{9}$$

where $i = 1, \ldots, n, j = 1, \ldots, p$ and $t = 1, \ldots, T$.

For given λ and W, we solve the optimization problem through coordinate descent, which has been developed and implemented for such objective functions by Friedman et al. (2010). This optimization approach achieves large improvements in computational efficiency over other minimization approaches, and allows our framework to be feasible for large-size data problems.

The basic steps of the algorithm are given below in Algorithm 4 and illustrated on a toy example in Figure 3. The main idea behind the algorithm is to:

1. Use results from previous time steps to form candidate biclusters for the current time period, then apply the penalization framework and permutation test to discover significant and stable biclusters (see Algorithm 5).
2. After candidate biclusters from previous times have been exhausted, a final search is performed with the penalization framework and permutation test for new biclusters that were not captured in the previous results (see Algorithm 6).
3. Return all significant biclusters discovered in previous steps 1 and 2.

We note that when $t = T, \{\theta_{ijk}^{(t)}\}_{t=T-W}^{T}$ are simultaneously estimated. After that, $t = T + 1$, and $\{\theta_{ijk}^{(t)}\}_{t=T-W+1}^{T+1}$ are estimated independently. As easily seen, these two sets are highly overlapping. We use the most recent estimate as the final estimator due mainly to its simplicity. This strategy is similar to using a rolling window smoother. In principle, other methods that combine the overlapping estimates could be employed. Though in practice, more complex strategies can sometimes complicate implementation without fundamentally changing the final estimator.

2.3. Implementation issues

Our implementation is performed in R (version 2.15), with all auxiliary functions supported in the basic distribution (R Core Team, 2012). R code is available at `www.stat.lsa.umich.edu/~smankad/`. Numerical results presented in the following sections are obtained using the code specification above on a Linux platform. Next, we discuss the permutation test used in the stopping criterion for the smooth plaid algorithm, and selection of the parameters λ and W.

Permutation Test. The permutation test utilized in Algorithm 3 is modified to accommodate the additional structure between data matrices. Specifically, matrix observations at different times should be permuted separately, so that global time effects are maintained. Also, the importance of bicluster k is measured over the time interval, instead of at a single time: $\sigma_k^2 = \sum_{t=T-W}^{T} \sum_{i=1}^{n} \sum_{j=1}^{p} \hat{r}_{ik}^{(t)} \hat{c}_{jk}^{(t)} \hat{\theta}_{ijk}^{(t)2}$. Algorithm 8 shows the smooth plaid permutation test.

Algorithm 4 Smooth plaid models estimation procedure.

Input: Matrices $\{X^{(t)} \in \mathbb{R}^{n \times p}\}_{t=T-W}^{T}$, M biclusters from previous times $\{\hat{r}_{im}^{(t)}, \hat{c}_{jm}^{(t)}\}_{m=1}^{M}$

Output: Biclusters $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,\ t=T-W}^{K,T}$

1: $\hat{\mu}_0^{(t)} = \frac{1}{n} 1_{n \times n} X^{(t)}, t = T - W, \ldots, T$

2: $\hat{Z}^{(t)} = X^{(t)} - \hat{\mu}_0^{(t)}, t = T - W, \ldots, T$

3: $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K',T} = searchPrevBCResults(\{\hat{Z}^{(t)}\}_{t=T-W}^{T}, \{\hat{r}_{im}^{(t)}, \hat{c}_{jm}^{(t)}\}_{m=1}^{M})$ (see Algorithm 5)

4: $\hat{Z}_{ij}^{(t)} = \hat{Z}_{ij}^{(t)} - \sum_{k=1}^{K'} \hat{r}_{ik}^{(t)} \hat{c}_{jk}^{(t)} \hat{\theta}_{ijk}^{(t)}$ for $t = T - W, \ldots, T, i = 1, \ldots, n, j = 1, \ldots, p$.

5: $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=K'+1,t=T-W}^{K,T} = searchNewBC(\{\hat{Z}^{(t)}\}_{t=T-W}^{T})$ (see Algorithm 6)

6: **return** $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K,T}$

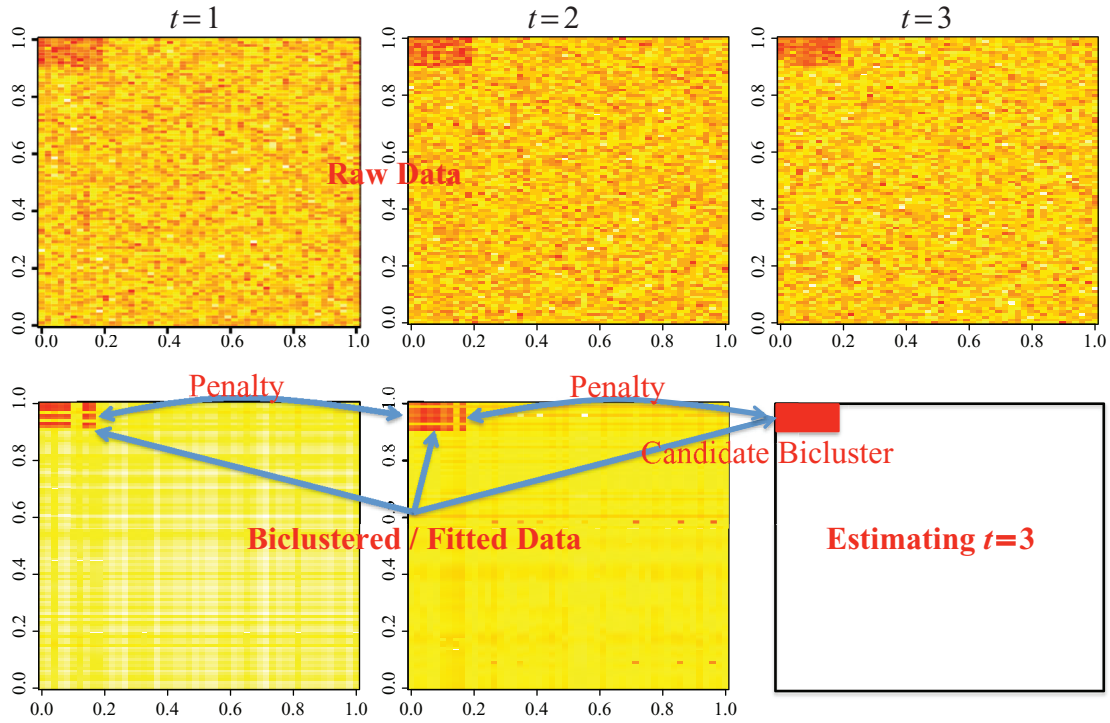

Fig. 3. The smooth plaid algorithm on a toy example. The first row contains raw data and the second row shows the fitted values. We are interested in estimating $t = 3$. We combine the imperfectly detected bicluster from time steps $t = 1$ and $t = 2$ to form the initial condition or candidate bicluster for $t = 3$. The double arrows denote the penalty.

Algorithm 5 searchPrevBCResults

Input: Matrices $\{\hat{Z}^{(t)} \in \mathbb{R}^{n\times p}\}_{t=T-W}^{T}$, M biclusters from previous times $\{\hat{r}_{im}^{(t)}, \hat{c}_{jm}^{(t)}\}_{m=1}^{M}$

Output: Biclusters $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K,T}$

$K = 1$ (bicluster counter)
for $m = 1, \ldots, M$ **do**
  $\hat{r}_{im}^{(T)} = \min(1, \sum_{t=T-W}^{T-1} \hat{r}_{im}^{(t)}), i = 1, \ldots, n$
  $\hat{c}_{jm}^{(T)} = \min(1, \sum_{t=T-W}^{T-1} \hat{c}_{jm}^{(t)}), j = 1, \ldots, p$
  $\{\theta_{ijm}^{(t)}\}_{t=T-W}^{T} =$ argmin of Equation 8
  $b = permuteTest2(\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T}, \{\hat{Z}^{(t)}\}_{t=T-W}^{T})$ (see Algorithm 7)
  if $b = 0$ **then**
    $\hat{Z}_{ij}^{(t)} = \hat{Z}_{ij}^{(t)} - \hat{r}_{im}^{(t)}\hat{c}_{jm}^{(t)}\hat{\theta}_{ijm}^{(t)}$ for $t = T - W, ..., T$.
    $K = K + 1$
  end if
end for
return $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K,T}$

It is argued in Lazzeroni and Owen (2000) that, since after permuting rows and columns the data values are independent of row and column labels, the approximate probability of accepting k or more false biclusters is $(L + 1)^{-k}$, where L is the total number of noise biclusters. The authors suggest four or fewer noise biclusters for each permutation test. Though, this is highly dependent on the size of the data and available computing power (costs are proportional to the number of noise biclusters). With the large sized

Algorithm 6 searchNewBC

Input: Matrices $\{Z^{(t)} \in \mathbb{R}^{n\times p}\}_{t=T-W}^{T}$

Output: Biclusters $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K,T}$

$K = 1$ (bicluster counter)
repeat
  $\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T} = estimateSmoothBicluster(\{\hat{Z}^{(t)}\}_{t=T-W}^{T})$ (see Algorithm 7)
  $b = permuteTest2(\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T}, \{\hat{Z}^{(t)}\}_{t=T-W}^{T})$ (see Algorithm 8)
  if $b = 0$ **then**
    $\hat{Z}_{ij}^{(t)} = X_{ij}^{(t)} - \hat{\theta}_{ijk}^{(t)}\hat{r}_{ik}^{(t)}\hat{c}_{jk}^{(t)}, t = T-W, \ldots, T$
    Set $K = K + 1$
  end if
until $b = 1$
return $\{\hat{r}_{ik}^{(t)}, \hat{c}_{jk}^{(t)}, \hat{\theta}_{ijk}^{(t)}\}_{k=1,t=T-W}^{K,T}$

Algorithm 7 estimateSmoothBicluster

Input: Matrices $\{\hat{Z}^{(t)} \in \mathbb{R}^{n\times p}\}_{t=T-W}^{T}$

Output: Biclusters $\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T}$

Apply k-means (k=2) to rows of $\hat{Z}^{(T)}$. Set $\{\hat{r}_{iK}^{(t)}\}_{t=T-W}^{T}$ to the smaller cluster.
Apply k-means (k=2) to columns of $\hat{Z}^{(T)}$. Set $\{\hat{c}_{jK}^{(t)}\}_{t=T-W}^{T}$ to the smaller cluster.
repeat
  $\{\theta_{ijK}^{(t)}\}_{t=T-W}^{T} = \text{argmin}$ of Equation 8
  for t=T-W,...,T **do**
    for i=1,...,n **do**
      if $\sum_{t=T-W}^{T}\sum_{j=1}^{p}(\hat{Z}_{ij}^{(t)} - \hat{\theta}_{ijK}^{(t)}\hat{c}_{jK}^{(t)})^2 < \sum_{t=T-W}^{T}\sum_{j=1}^{p}\hat{Z}_{ij}^{(t)2}$ **then**
        $\hat{r}_{iK}^{(t)} = 1$
      else
        $\hat{r}_{iK}^{(t)} = 0$
      end if
    end for
    for j=1,...,p **do**
      if $\sum_{t=T-W}^{T}\sum_{i=1}^{n}(\hat{Z}_{ij}^{(t)} - \hat{\theta}_{ijK}^{(t)}\hat{r}_{iK}^{(t)})^2 < \sum_{t=T-W}^{T}\sum_{i=1}^{n}\hat{Z}_{ij}^{(t)2}$ **then**
        $\hat{c}_{jK}^{(t)} = 1$
      else
        $\hat{c}_{jK}^{(t)} = 0$
      end if
    end for
  end for
until $\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}\}$ converge or maximum iteration number attained
return $\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T}$

data encountered in our application, we set $L = 3$. This parameter can be adjusted by the user to balance accuracy and computational ease.

Choosing λ. The effect of λ is to create smoother paths over time for each bicluster. Specifically, larger penalization levels force biclusters to have similar estimated values as in neighboring time steps.

A systematic way to choose λ is through cross-validation. The idea behind cross validation is to use a random subset of the data to fit the model, and the rest

Algorithm 8 permuteTest2

Input: Biclusters $\{\hat{r}_{iK}^{(t)}, \hat{c}_{jK}^{(t)}, \hat{\theta}_{ijK}^{(t)}\}_{t=T-W}^{T}$, Matrices $\{Z^{(t)} \in \mathbb{R}^{n\times p}\}_{t=T-W}^{T}$
Output: $\{0, 1\}$

$\sigma_K^2 = \sum_{t=T-W}^{T}\sum_{i=1}^{n}\sum_{j=1}^{p}\hat{r}_{iK}^{(t)}\hat{c}_{jK}^{(t)}\hat{\theta}_{ijK}^{(t)2}$.
for l=1,...,L (number of noise layers) **do**
  for t=1,...,T **do**
    $\pi_r^{(t)} = permutation(\{1,\ldots,n\})$
    $\pi_c^{(t)} = permutation(\{1,\ldots,p\})$
    $\tilde{Z}_l^{(t)} = \hat{Z}^{(t)}(\pi_r^{(t)}, \pi_c^{(t)})$
  end for
  $\tilde{r}_{il}^{(t)}, \tilde{c}_{jl}^{(t)}, \tilde{\theta}_{ijl}^{(t)} = estimateSmoothBicluster(\{\tilde{Z}_l^{(t)}\})$ (see Algorithm 7)
  $\hat{\sigma}_{n_l} = \sum_{t=T-W}^{T}\sum_{i=1}^{n}\sum_{j=1}^{p}\tilde{r}_{il}^{(t)}\tilde{c}_{jl}^{(t)}\tilde{\theta}_{ijl}^{(t)2}$.
end for
if $\hat{\sigma}_K^2 > \max\{\hat{\sigma}_{n_1}^2, ..., \hat{\sigma}_{n_L}^2\}$ **then**
  return 0
else
  return 1
end if

of the data to assess model accuracy. Different values of λ are cycled over and the one that corresponds to the lowest test error is chosen.

In particular, suppose one is given a sequence of potential λs. Cross-validation divides the samples into G groups. Then for each potential λ, Equation 8 is minimized G times, once with each of the groups omitted. The coefficients from each estimation are used to predict the omitted group. The error is accumulated, and average error and standard deviation over the G groups is computed. Finally, the λ corresponding to the lowest mean squared error is chosen. A full algorithmic description, including selecting the initial λ sequence, with code can be found in Friedman et al. (2010).

Under this framework, the 'optimal' value of λ (denoted by λ^*) can change each time we minimize the penalized objective function. Thus, for the same bicluster, λ^* over time periods $T-W,\ldots,T$ may be different than the λ^* chosen for $T-W+1,\ldots,T+1$. Further, λ^* can change for different biclusters over the same time window. This flexibility is ideal given that different trader groups may follow different dynamics, and those dynamics may be time-varying.

Choosing W. The parameter, W, controls the window width for smoothing, e.g., the number of previous time steps to include in the smoothing. Larger values of W mean that the model has more memory so it incorporates more observations for estimation. This risks missing sharper changes in the data and only detecting the most persistent patterns. On the other hand, small values of W make the fitting more sensitive to sharp changes, but increase variance due to smaller number of observations. We find setting $W = 1$ (penalizing over adjacent matrices) is sufficient for filtering out most noisy expressions. Other values could be used if external information is known, or if additional smoothing is needed.

3. An illustrative example

Before applying our model to the E-mini S&P 500 Futures Contract, we illustrate and validate our methodology with simulated data.

We generate a time-series of data matrices, where each matrix has 100 rows and columns, with embedded biclusters that evolve through time. In particular, we have $X^{(t)} \in \mathbb{R}^{100\times 100}$, where $X_{ij}^{(t)} \sim N(\mu_k^{(t)}, 1)$. The background layer has mean $\mu_0 = 0$, and there are two biclusters with means shown in Figure 4. One bicluster has constant mean, while the other oscillates with time. The size of each bicluster is 16% of the size of X, and is constant throughout time.

We use only a mean effect for the bicluster effect for simplicity, that is, $\theta_{ijk} = \mu_k$. Figure 5 shows the estimated values for different levels of penalization. We see that if λ is too large, only the most persistent pattern is detected. On the other hand, the λ from cross-validation yields a nearly complete

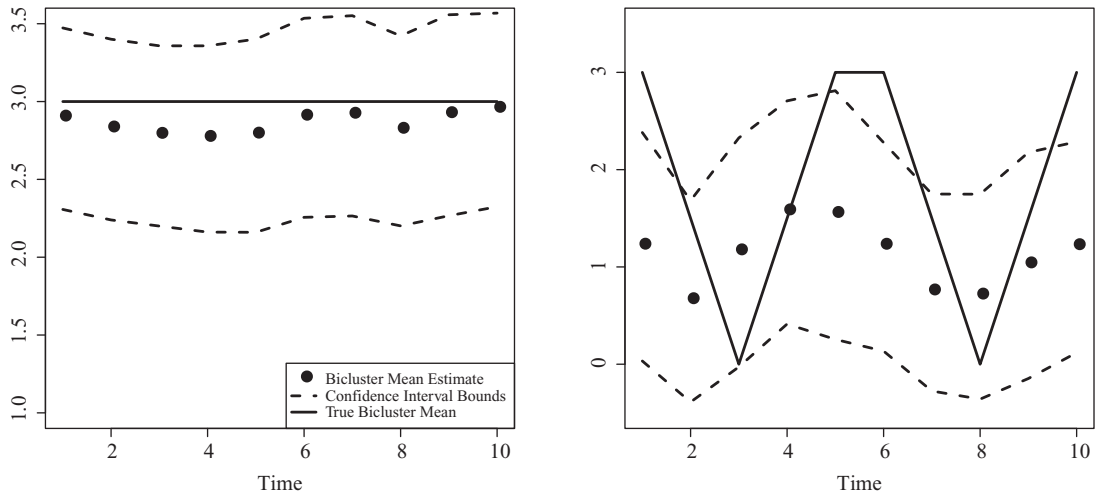

Fig. 4. True and estimated bicluster means in simulated data. One bicluster has constant mean, while the other oscillates. The estimates are obtained with λ chosen through cross validation.

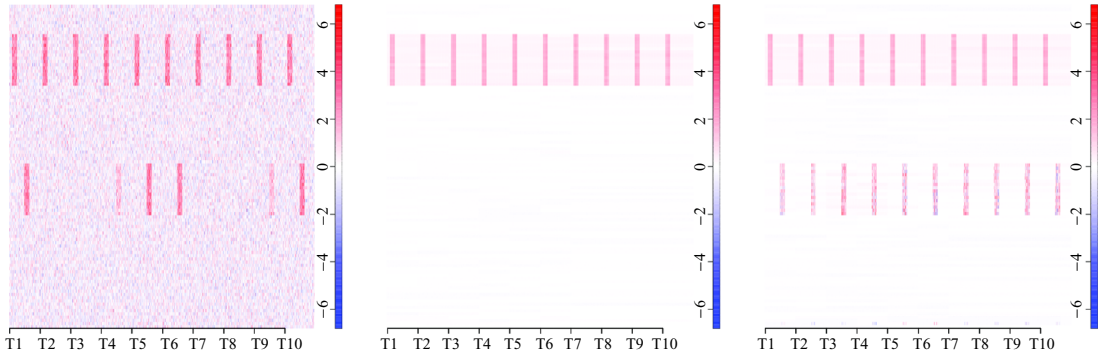

Fig. 5. The left panel shows an example of the raw data, unfolded (repeatedly concatenated) to be a 2-D array $[X^{(1)}|X^{(2)}|...|X^{(T)}]$. The central panel shows the estimated values with $\lambda = \infty$. The right panel shows the estimated values when choosing λ via cross validation.

perfect detection of the biclusters, while smoothing the expression patterns over time.

We compare the proposed smooth plaid model with the more direct approach of applying the static plaid model to each time step separately. The percentage of biclusters detected and false positive results are presented in Table 1. We see that the smooth plaid procedures perform favorably, since it does a significantly better job at detecting the dynamic bicluster, while maintaining a negligible number of false positives and the detection of the stable bicluster.

Panel B of Table 1 shows that the recovered bicluster contributions to explained variance are relatively small. This highlights that the proposed approach can be advantageous when finding 'needles in a haystack', and is closely related to anomaly detection.

Table 1

Simulation Results from the Illustrative Example. % Variance Explains is defined as $1 - \sum_t ||\hat{X}^{(t)} - X^{(t)}||_F^2 / \sum_t ||X^{(t)}||_F^2$.

Panel A: Detection Accuracy

Algorithm	% Stable Bicluster Detected	% Dynamic Bicluster Detected	% False Positive
Smooth Plaid	93.7	87.9	1.6
Static Plaid	89.1	40.6	1.7

Panel B: Estimated Smooth Plaid Bicluster Statistics

Bicluster	Number Rows	Number Columns	% Variance Explained
Bicluster 1	16	16	16.6
Bicluster 2	16	16	2.9
Overall			19.5

In summary, the smooth plaid procedures perform favorably in this synthetic setting by discovering the true, underlying biclustering structure and evolution.

4. Applying smooth plaid models to transaction-level data

We now apply the smooth plaid model to transaction-level data generated by an agent-based simulation model calibrated to the E-mini S&P 500 futures contract. The E-mini trades electronically on the CME Globex trading platform, a fully electronic limit order market. Limit orders are submitted by traders wishing to buy and sell a certain number of contracts up to a certain price (or at the market price for market orders). Submitted orders are matched by a matching algorithm. The number of outstanding E-mini contracts is created directly by buying and selling interests. There is no limit on how many contracts can be outstanding at any given time. The CME Globex matching algorithm for the E-mini offers strict price and time priority. Specifically, orders to buy at higher prices or sell at lower prices are placed in queues ahead orders to buy at lower prices or sell at higher prices. Orders that offer to buy or sell at the same price arranged in the order that they have arrived into the Globex matching engine.

We use simulated, transaction-level data generated by an agent-based model (Paddrik et al., 2011; Hayes et al., 2012) that retains key attributes of actual E-mini 500 stock index futures data observed by the exchanges and regulators. The model of Paddrik et al. (2011) and Hayes et al. (2012) simulates traders who submit orders to a limit order book according to different combinations of trading styles. Orders are matched according to strict price and time priority like with the CME Globex matching algorithm. The six different trading styles are based on the findings of Kirilenko et al. (2010) and consistent of fundamental buyers and sellers, market makers, opportunistic and high frequency traders. The different trader types are calibrated to match key aspects of behavior found in real E-mini 500 stock index futures market data, such as trading speed, market volume share, and position limits. For each simulated transaction, we know the buyer and the seller, the price and quantity at which they traded, and the time of execution.

We use the following variables to cluster traders into groups: trades per second, trading volume (total number of contracts traded), cumulative inventory/net position (reset to zero for each trading account at the end of each trading day), change in inventory, and median duration for each trader. Each of the variables is calculated for a preset time period. We define the time period as 600 transactions (trades). Our results are robust with respect to different sampling schemes. Though, if too small of a time period is used, then most traders will not have participated in any transactions and the data matrices will contain many zeros. Such sparsity can mask the slower groups of traders. A brief description of each variable follows.

Trades per second are computed by dividing the total number of transactions that a trader makes in a given time period by the total number of seconds in that time period. Trades per second is indicative of the decision horizons and execution strategies for different traders.

Trading volume is computed for each trader by summing up the total number of contracts transacted in each time period. Trading volume is indicative of the overall trading activity of a particular trader.

Change in inventory is computed by subtracting the number of contracts sold during a particular period from the number of contracts bought during the period. Change in inventory is indicative of a risk exposure of a particular trader accumulated during a period of time.

Cumulative net inventory is calculated by accumulating a trader's inventory from the beginning of the day to the end of the current time period. Cumulative net inventory indicates the direction (long or short) and size of the risk exposure of a trader accumulated from the beginning of the day.

Lastly, intertrade duration is defined as the time (in seconds) until the next trade. Specifically, for each transaction involving a given trader, we compute the time, in seconds, until the next transaction between any two traders. We then compute the median intertrade duration for a trader during a sample period.

Each trading variable measures different aspects of how much, in which direction, and how quickly each trader transacts. Once organized, the data contains 6387 rows (traders), 5 columns, and 792 time periods. Each day contains between approximately 30 to 50 time periods, depending on the number of transactions per day.

After we apply our algorithm to the data, we use additional filtering on the fitted values to separate traders into five broad groups. The additional grouping consists of variable thresholds that separate traders into groups and is based on characteristics that measure a trader's strategic profile, such as, among

others, the rate of a trader's mean reversion of fitted accumulated inventory. Thus, we employ the smooth plaid method as a temporal filter to facilitate trader classifications. Our method improves on the direct approaches by cleaning the temporal noise, allowing the additional classification of thousands of traders into interpretable categories.

The main benefit of employing the smooth plaid method is in separating market makers, opportunistic, and small traders. The more direct approaches struggle with these traders since they have strategies that can appear very similar statistically when the time dimension is aggregated.

Altogether, we find 7 high frequency traders (HFT's), 73 market makers, 2405 opportunistic traders, 1281 fundamental position traders, and 2849 small/residual traders.

HFTs occupy a distinctive niche in the ecosystem of modern electronic markets. They trade through an enormous number of contracts each day, but carry very little inventory at any point in time. Market makers have a footprint qualitatively similar to HFTs, but significantly smaller volume-wise. Fundamental traders are primarily characterized by large positive or negative cumulative net positions at the end of a trading day. The use of temporal information is quite important in identifying this group, since some of its members accumulate directional positions by executing many small-size orders, while others execute a few larger-size orders, thus trying to disguise their behavior so as not to be taken advantage by the market. This feature is to a large extent lost to an analysis ignoring the temporal dimension. The same holds true for the group of small traders that trade infrequently at random points in time, hence lacking any persistent pattern. The final group consists of opportunistic traders that have a persistent presence in the market, but their trading behavior bifurcates between fundamental positioning and market making.

Figures 6, 7, and 8 illustrate that the five groups exhibit different trading signatures. Fundamental traders accumulate either a large positive or negative net imbalance. On the other hand, all other groups have on average zero net position. Opportunistic traders net positions vary more than market makers, which vary more than high frequency traders. These trader groups are conceptually similar to the bicluster in the illustrative example with oscillating mean structure, and as we saw, the smooth plaid model has superior performance for such dynamic behavior by conditioning on previous time points.

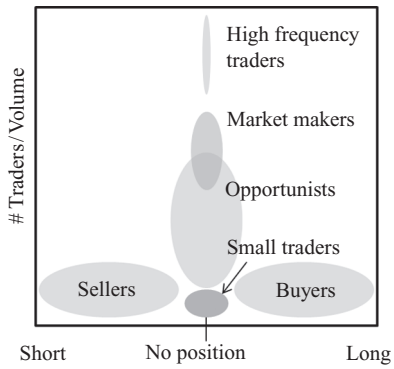

Fig. 6. Stylized representation of the net position (x-axis) versus volume/number of trades. After applying the smooth plaid procedure, we can additionally classify market makers, opportunists and small traders.

5. Conclusion

In this study, we present a dynamic machine-learning method that designates traders in a liquid financial market into five persistent categories based on their footprint in the data. Our method is based on a plaid clustering technique enhanced by a smoothing framework that filters out transient patterns. The method performs extremely well on regulatory, transaction-level data for the E-mini S&P 500 stock index futures contract, the price discovery vehicle for the broad U.S. stock market. However, in order to preserve confidentiality of the regulatory data, the results we present employ simulated data generated by an agent-based simulation model of an electronic market calibrated to the E-mini.

For comparison, Table 2 shows that our classification of traders is consistent with the study by Kirilenko et al. (2010), which classified trader behavior using similar E-mini futures data three days before and during the Flash Crash of May 6, 2010. While investigating the triggering event of the Flash Crash, Kirilenko et al. (2010) manually designated trading accounts that traded in the E-mini on May 6, 2010 into the same six distinct categories. The categorization was essentially based on the dynamics of two characteristics: end of day holdings and intraday trading volume for each trading account.

The similarity in our groupings validates and demonstrates the usefulness of our method, since these previous reports manually classified each trader through an exhaustive and labor intensive procedure. Our biclustering algorithm was able to detect similar groups and

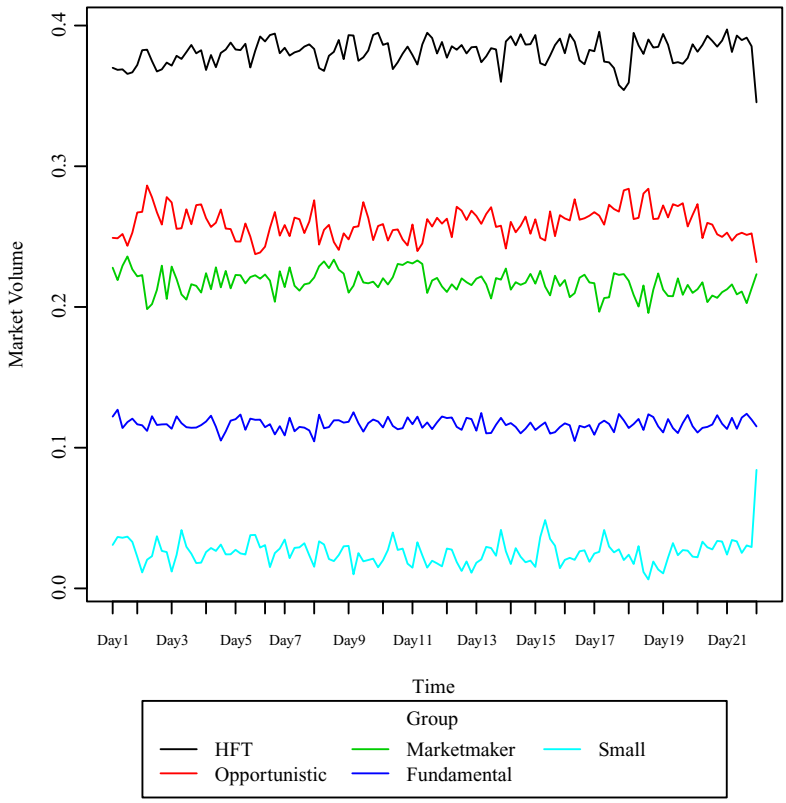

Fig. 7. Volume-based market share for each group of trader over each day of data. The high frequency traders persistently trade a significantly large number of contracts.

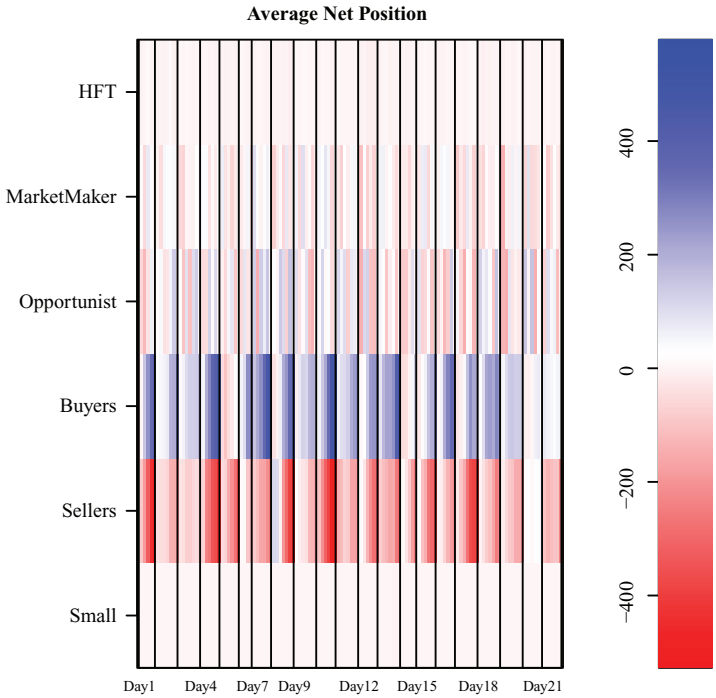

Fig. 8. This heatmap portrays average net position for each group of trader over each day of the month. The different trader types have different signatures in the data.

Table 2
Grouping traders in the E-mini S&P 500 futures contract. Our analysis is performed on simulated E-mini data, while Kirilenko et al. (2010) analyze regulatory data for the May 2010 E-mini S&P 500 futures contract.

Trader Type	Smooth Plaid	Kirilenko et al. (2010)
HFT	7	16
Market maker	73	179
Opportunistic	2405	5808
Fundamental Buyers/Sellers	1281	2539
Small	2849	6880

the relevant variables that consistently separate them over time using a novel machine-learning methodology.

We argue that the smooth plaid model can be effectively used for the analysis of traders and their strategies in electronic financial markets. In an environment where traders do not have formal designations, the smooth plaid model forms a useful first step to separate tens of thousands of trading accounts into manageable trader categories for subsequent academic, policy and regulatory analysis.

We also expect our method to be useful in other applications where one is given a time-series of matrices, such as examining traders across different markets or analyzing macroeconomic variables for different entities over time.

6. Acknowledgments

Shawn Mankad and George Michailidis were supported by grant DMS 1228164.

The authors also thank an anonymous referee and the Editor Philip Maymin for their valuable comments and suggestions to improve the quality of the paper. The authors are also extremely grateful to Mark Paddrik for sharing the transaction-level data generated by the agent-based simulation model that he and his co-authors have developed.

References

Chaboud, A., Hjalmarsson, E., Vega, C., Chiquoine, B., 2011. Rise of the machines: Algorithmic trading in the foreign exchange market. *SSRN eLibrary* http://ssrn.com/abstract=1501135 [Accessed July 11, 2013].

Cheng, Y., Church, G.M., 2000. Biclustering of expression data. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, AAAI Press, Menlo Park, CA, pp. 93–103. ISBN 1-57735-115-0.

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. J. Stat. Softw. 33 (1), 1–22. http://www.jstatsoft.org/v33/i01/ [Accessed July 11, 2013].

Hastie, T., Tibshirani, R., Friedman, J.H., 2001. The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations. Springer-Verlag, New York.

Hayes, R.L., Paddrik, M.E., Todd, A., Yang, S.Y., Beling, P., Scherer, W., 2012. Agent based model of the E-Mini S&P 500 future: Application for policy making. Proceedings of the 2012 Winter Simulation Conference. http://ssrn.com/abstract=2112139 [Accessed July 11, 2013].

Hendershott, T., Jones, C.M., Menkveld, A.J., 2010. Does algorithmic trading improve liquidity? J. Finance. 66, 1–33. http://ssrn.com/abstract=1100635 [Accessed July 11, 2013].

Huang, Q.-H., 2011. Discovery of time-inconsecutive co-movement patterns of foreign currencies using an evolutionary biclustering method. App. Math. Comput. 218 (8), 4353–4364. ISSN 0096-3003. doi: 10.1016/j.amc.2011.10.011. http://www.sciencedirect.com/science/article/pii/S0096300311012410.

Kirilenko, A.A., Kyle, A.S., Samadi, M., Tuzun, T., 2010. The Flash Crash: The impact of high frequency trading on an electronic market. http://ssrn.com/abstract=1686004 [Accessed July 11, 2013].

Lazzeroni, L., Owen, A., 2000. Plaid models for gene expression data. *Stat. Sin.* 12, 61–86.

Paddrik, M.E., Hayes, R.L., Todd, A., Yang, S.Y., Scherer, W., Beling, P., 2011. An agent based model of the E-Mini S&P 500 and the Flash Crash. SSRN eLibrary. http://ssrn.com/abstract=1932152 [Accessed July 11, 2013].

R Core Team. 2012. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/. ISBN 3-900051-07-0 [Accessed July 11, 2013].

Turner, H., Bailey, T., Krzanowski, W., 2005. Improved biclustering of microarray data demonstrated through systematic performance tests. *Comput. Stat. Data Anal.* 48 (2), 235–254. ISSN 0167-9473. doi: 10.1016/j.csda.2004.02.003. http://www.sciencedirect.com/science/article/pii/S0167947304000295.