转移置换熵：一种度量金融时间序列动态复杂性的新方法

Permutation transition entropy: a new method for characterizing the dynamical complexity of financial time series

中文摘要

转移置换熵可以刻画时间序列的动态复杂性。通过量化相邻置换模式的马氏状态转移过程，可以获取时间序列状态变化的轨迹。不同于传统的度量静态复杂度的方法，置换转移熵可以识别时间序列在暂态结构变化中的动态复杂性。本文将转移置换熵方法应用于金融时间序列，通过数值实验说明转移置换熵能揭示时间序列一些新的信息，而这些信息是置换熵不能展现出来的。首先，将转移置换熵方法用于分析正弦周期性时间序列，随着嵌入维度的增加，将会有更多的状态转移模式，因此时间序列的内在复杂程度增加，转移置换熵的值变得更大，但是这个值会比纯随机时间序列的转移置换熵的值小很多。我们又将转移置换熵的方法用于分析 Logistic 时间序列，通过分析，我们发现转移置换熵可以有效地区分混沌时间序列和纯随机时间序列。我们还将转移置换熵方法用于金融时间序列，而且发现了纳斯达克市场的每日收盘价时间序列中动量效应的存在。并且该指标的动态复杂性低于纯随机时间序列的动态复杂性，因此即将发生的每日收盘价的状态会比较规律，可以对其进行预测。而该市场的对数收益率时间序列的转移置换熵的值和纯随机时间序列的转移置换熵值比较相似，因此它的动态复杂性较高，难以直接进行预测。

关键词：转移置换熵，动态复杂性，马氏状态转移，时间序列分析，多尺度分析

ABSTRACT

In this paper, we apply the method of Permutation Transition Entropy, which quantifies the Markov states transition between adjacent permutations, to measure the dynamical complexity of non-stationary time series. This method can capture the change of states trajectory of the underlying system by quantifying the Markov states transition between adjacent permutations. Unlike many traditional methods which usually measure the static complexity, the new method of permutation transition entropy(PTE) is able to identify the dynamical complexity with respect to the temporal structure change of the time series. By numerical analyses, we show that the PTE can give new information while other methods, like the permutation entropy (PE), cannot. We apply the PTE method to the financial time series, and find the existence of the momentum effect in the daily closing price and the daily trading volume of NASDAQ Composite Index. It indicates that the dynamical complexity of the index is lower than that of the purely random time series. The forthcoming state of the daily closing price is, therefore, a bit regular, which can be used for prediction. While the logarithm return shows very similar PTE values with those of the purely random time series, which correspond to high dynamical complexity. Furthermore, with multiscale analysis towards PTE, it turns out that under the same embedding dimension, the series with higher time-scale are more deterministic and represents more obvious momentum effect.

KEYWORDS: Permutation Transition Entropy, Dynamical complexity, Markov state transition, Time series analysis, multiscale analysis

目　　录

1 引言

如何定义系统的复杂程度是一个很重要的问题，也有很多研究尝试去解答这个问题[1-3]。一种方法是度量系统的时间序列的不规则性，或者说是多样性[4-5]。如果该系统特别不规则而且有很多状态，所以去解释这些状态变化将需要大量的信息，同时，该时间序列有着很高的静态结构复杂度。

系统的复杂程度也可以在动态的结构变化下得到解释。如果一个系统的状态不能简单地由过去已发生过的状态来预测，那么要描述该系统新的状态就需要大量的信息去实现，而且这个系统有着很高的复杂度。然而，若一个系统只有很少的几种状态，那么根据之前的状态，将相对容易地预测该系统的下一状态，因此该系统的复杂程度较低。

时间序列的动态复杂度与其时序结构关系紧密，它表现了数据相对于时间的内在结构模式。时序结构不仅描述了邻近数据点之间的联系，而且能包含远距离数据点对复杂度的影响。比如自相关函数是一种描述线性相关性的简单方法。对时序结构的分析可用于刻画复杂系统的内在行为模式，可以监控系统状态的变化，并且预测系统的变化轨迹。在生理学中，通过分析心跳时序结构来区分健康和处于病态的心跳，并且这种区分结果是具有鲁棒性的[6]。在金融中，股票时间序列的时序结构可用来量化股票市场的有效程度并且可用于区分股票市场发展从新兴到趋于成熟的各个阶段[7]，对于系统的时间序列来说，如果时间序列的时序结构非常有规律，或者在任何时候的时序状态都与之前的状态紧密关联，那么这个时间序列将更可能被预测到。这意味着要产生新的数据点，需要的信息将会更少，因此该时间序列的动态复杂度是较低的。

已经有很多的研究来探索单一时间序列的复杂性，而且也有很多相应的方法，比如李亚诺夫指数[8]，分型理论[9]，熵理论[10-11]等。很多现有的方法都是基于分析静态复杂度的。这意味着这些方法都仅仅是度量系统状态的多样性，而不是度量系统状态变化的多样性。置换熵是信息理论中度量系统复杂程度的方法，它的优点是形式简单，便于计算，并且具有鲁棒性。但是使用置换熵这种方法并不能刻画系统的动态复杂程度。也有很多学者利用置换熵来探索时间序列的动态变化特征[12]。但他们是利用滚动窗口期的方法来计算特定周期内数据的置换熵。这种方法的前提是要设置窗口期大小这一参数。而且这种方法不能识别时间序列从一种状态立刻转移到下一状态时的动态变化，因为要计算一段时间序列的置换熵，窗口期不能设置得太小。

本文将通过转移置换熵来刻画单一时间序列的动态复杂程度，系统的状态可以由状态向量的置换模式来表示。本文使用转移置换熵来量化排序模式发生变化的不确定性。不同于很多传统的方法，像近似熵和样本熵比较两个远距离状态向量的相似性和差异性，置换转移熵能够分辨马氏链中的一步转移过程。对一个动态复杂度较低的系统来说，其转移置换熵的值较低。而且转移置换熵可以反映时间序列的可预测性，因为动态复杂度较低的时间序列的可预测概率更高。

首先，我们先给出置换熵的计算方法，然后介绍转移置换熵。我们将讨论新方法的一些特征，然后我们将转移置换熵方法用于模拟时间序列的研究，然后将其应用到金融时间序列的研究中，最后我们给出结论。

2. 转移置换熵方法

2.1 置换熵

为了计算信息熵和统计复杂度，首先需要构建一个时间序列的概率分布。Bandt[13]提出了一种符号化时间序列的方法，运用该符号化方法不但可以保留时间序列的时间结构，即数据间的相对大小关系，而且可以构建不同排序模式，即符号串组合的概率分布。

(i) 在 t 时刻，我们用状态向量X_t来表示系统的状态：

$$X_t = \{x_t, x_{t+\tau}, \dots, x_{t+(m-1)\tau}\} \quad (1)$$

τ和 m 分别表示时滞项和相空间中的嵌入维度。在不同时间点的状态向量矩阵

$$X=[X_1, X_2, \dots, X_{T-(m-1)\tau}]^T \quad (2)$$

(ii) 通过将向量中的相邻值进行比较，每一个状态向量X_t都分别有一个独特的排序模式$\pi\left(X_t\right)$，比如，向量$X_t = \{3,9,4,5\}$的排序模式是$\pi\left(X_t\right) = 0312$，因为$X_t(0) < X_t(2) < X_t(3) < X_t(1)$. 在确定嵌入维度 m 后，最多有 m!种排序模式。

(iii) 对所有类型的排序模式进行计数，并分别归类到π_i, i=1,2,…,m!, 对每一个$\pi\left(X_t\right)$, $\pi\left(X_t\right) \in \{\pi_i | i = 1, \dots, m!\}$。$\pi_i$发生的概率通过其发生的频率求得：

$$P\left(\pi_i\right) = \frac{n\{X_t | X_t \text{的排序模式为} \pi_i\}}{T-(m-1)\tau} \quad (3)$$

$1 \le t \le T-(m-1)\tau$,n 表示该集合中元素的个数。

(iv) 包含在π_i中的信息和 $P\left(\pi_i\right)$可以用$log_2[1/p\left(\pi_i\right)]$量化，所有排序模式的平均信息量由期望$H_m$给出：

$$H_m = -\sum_{i=1}^{m!} P\left(\pi_i\right) log_2[P\left(\pi_i\right)] \quad (4)$$

H_m就是计算得到的置换熵，置换熵量化了数据的复杂程度和不规则程度。H_m的范围为$[0, log_2(m!)]$。置换熵的值越大表示$\{x_t\}$的复杂程度越高。当所有排

序模式都服从均匀分布时。置换熵的值达到最大，此时 $\mathrm{P}\left(\pi_i\right)=1/m!$；当一种排序模式以概率 1 发生，而其他的排序模式几乎不出现时，置换熵的值最小。通过标准化后，置换熵的值的范围为[0,1]。$h_m=H_m/[log_2(m!)]$

为了确保 m!≪ T，嵌入维度 m 通常设置为 2，3，4，5，6，7，时滞项通常设为 1.

2.2 转移置换熵的定义

置换熵可以度量排序模式发生的概率，但是不能刻画邻近排序模式之间的关系。我们将进一步分析时间序列的时序结构，探究排序模式的动态转移概率，因此提出新的度量时间序列动态复杂性的方法：置换转移熵。

对时间序列$\{x_t\}$,t=1,…,T，在时间 t 时的状态向量由X_t表示，其独特的排序模式为 $\pi(X_t),\ \pi\left(X_t\right)\in\{\pi_i|i=1,\ldots,m!\}$. 我们通过量化邻近排序模式的改变来测算动态复杂性。这是一个一步马氏状态转移过程。在给定的排序模式下，如果之后的置换模式更加具有确定性，那么系统将以一种更加确定的方式进行演化，这样，系统的复杂程度较低。

假设$\pi\left(X_t\right)=\pi_i$，$\pi\left(X_{t+1}\right)=\pi_j$，t=1, …, T−(m − 1)τ, i,j=1, …, m!

由π_i到其邻近的排序模式π_j的一步马氏转移概率为

$$\mathrm{P}\left(\pi_i\to\pi_j\right)=P\left(\pi_j\middle|\pi_i\right)=\frac{P(\pi_i,\pi_j)}{P(\pi_i)} \tag{5}$$

条件概率$P\left(\pi_j\middle|\pi_i\right)$表示邻近置换模式的转移概率。如果一种排序模式之后邻接的排序模式更加确定，那么这种置换转移模式将更容易同时出现，那么该系统的确定性也更高。因此该系统的动态复杂程度较低，并且具有较高的可预测性。另一方面，如果一种排序模式后邻接的排序模式在不同的时间点都是不同的，那么这个系统显得更加的随机，这表明该系统的动态复杂性较高，而且可预测程度较低。

转移置换熵是通过条件熵定义的，通过统计所有排序模式，由联合概率密度函数和归一性得:

$$T_m=-\sum_{i=1}^{m!}\sum_{j=1}^{m!}P(\pi_i,\pi_j)log_2[P(\pi_j|\pi_i)]=-\sum_{i=1}^{m!}\sum_{j=1}^{m!}P(\pi_i,\pi_j)log_2\frac{P(\pi_i,\pi_i)}{P(\pi_i)}$$

$$=H_m\left(\pi_i,\pi_j\right)-H_m(\pi_i)$$

T_m即为置换转移熵，$H_m\left(\pi_i,\pi_j\right)$是联合熵，$H_m\left(\pi_i\right)$是置换熵。在给定前一状态$\pi_i$的条件下，置换转移熵量化了描述排序模式$\pi_j$所需要的信息量。

我们首先考虑 m=2 的情形，此时仅存在两种排序模式，分别是 01 和 10。01 表示该排

序模式递增，而 01 表示递减的排序模式。在这两种排序模式中，一共有 4 种可能的状态转移模式（见图 1）：

(i) 01→ 01, (ii) 01→ 10, (iii) 10→ 01(iv) 10→ 10.

在这四种转移模式中，(i)和(iv)表示模式状态没有变化，而(ii)和(iii)则表示模式发生变化。在图一中，我们在二维平面展现这种排序模式的变化。在对角线下方，x_t总是大于x_{t+1}，故表示 01 这种排序模式。 当 m≥ 3 时，m 维空间将被分成 m!个子空间，而且在每一对排序模式之间将会有$(m!)^2$种可能的转移模式。

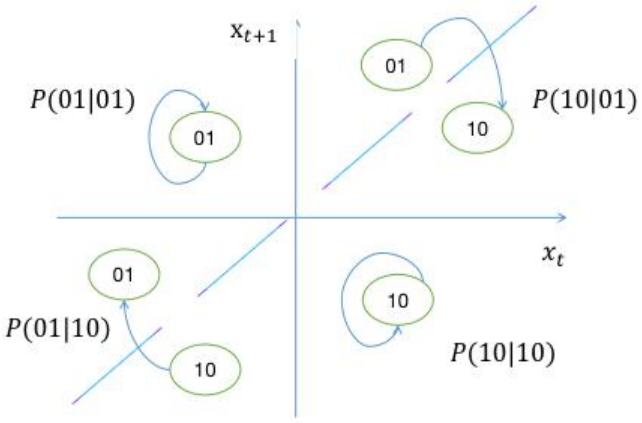


图 1:在排序模式 01 和 10 下的四种置换转移模式

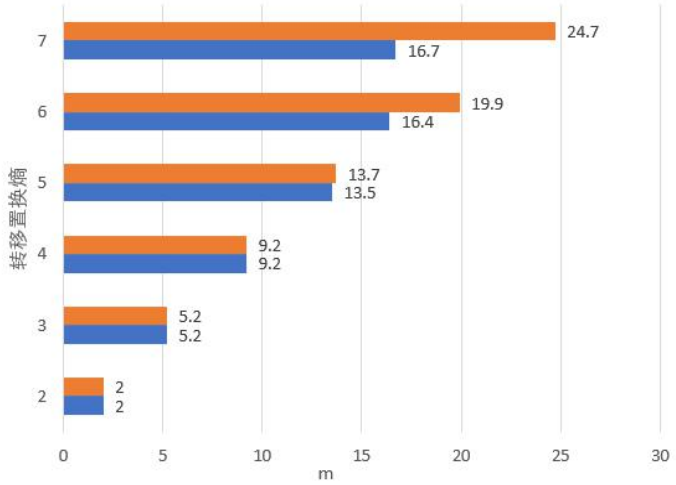
2.3 转移置换熵方法的性质

(1) 转移置换熵的范围

根据定义，转移置换熵的值是非负的，即$T_m \geq 0$.当时间序列的动态复杂性最低时，转移置换熵的值趋近于 0.这意味着该系统种只存在很少的几种模式转移，在单增或者单减的时间序列，以及存在某些规律特征的时间序列种会出现这种情况。另一方面，当所有状态变化都服从均匀分布时，即每一种状态都在下一时刻随机转变为另一种状态时，转移置换熵的值达到最大。在这种情况下，$P\left(\pi_i\right) = 1/(m!)$，$P\left(\pi_i, \pi_j\right) = 1/(m!)^2$，$P\left(\pi_j \middle| \pi_i\right) = 1/(m!)$ 所以转移置换熵T_m的最大值为 2log$_2$(m!)，标准化后，PTE 的范围为 $0 \leq T_m/[2log_2(\mathrm{m!})] \leq 1$.

(2) 有限样本效应

在计算置换熵的时候，对样本数据量的要求较低，只要满足 m!≪T 就可以了。但是计算转移置换熵时需要满足$(m!)^2 \ll T$. 因此，在计算时间序列转移置换熵时，通常设置 m=2, 3, 4, 5. 在图二中, 我们生成一组长度 T=10^6的高斯白噪声序列, 并分别设 m=2, ..., 7. 转移置换熵的理论值通过$T_m = 2log_2(m!)$得到，然后与转移置换熵的实际值进行比较。当 m=2，3，4，5 时，理论值和实际值拟合得很好，但是由于有限样本效应，当 m=6 时理论值和实际值的偏差开始产生，在 m=7 时偏差增大。当样本数据量增大时，偏差的发生将会滞后。



图二：数据总量为10^6的高斯白噪声序列的转移置换熵的理论值为 2log_2m!，与实验值相比的结果，在 m=6 和 m=7 时由于有限样本效应，将会出现偏差。m=6 时会出现 518400 种置换转移模式，在10^6的数据集下，有一些置换转移模式没有体现出来。

(3)错误地统计状态转移模式

如果我们不能正确地设置时滞项τ，将会发生错误的模式转移统计方式。比如，对高斯白噪声序列，在 m=3，$\tau = 1$ 时，第一个状态向量是$X_1 = [x_1, x_2, x_3]$，第二个向量是$X_2 = [x_2, x_3, x_4]$.如果X_1的排序模式是 012，这表示$x_1 < x_2 < x_3$，因此X_2的排序模式就不能是 210，102 或者是 201，因为$x_2 < x_3$，因此，我们可能人为地在邻近向量之间引入了某些联系从而导致错误的置换转移模式发生。现有两种方法来解决这个问题

(i) 第一种方法是设置时滞项$\tau \geq 2$，以$\tau = 2$ 为例，第一个状态向量$X_1 =$

$[x_1, x_3, x_5]$，第二个向量为$X_2 = [x_2, x_4, x_6]$，这样，在X_1和X_2之间没有人为加入的联系，尽管X_1和$X_3 = [x_3, x_5, x_7]$之间都包含x_3, x_5，这样是合理的，因为我们只考虑$\pi(X_1)$到$\pi(X_2)$的一步转移模式。

(ii) 第二种方法是设置$\tau = 1$, 当 m=3 时, $X_1 = [x_1, x_2, x_3]$, $X_2 = [x_2, x_3, x_4]$, $X_3 = [x_3, x_4, x_5]$，$X_4 = [x_4, x_5, x_6]$.不同于考虑从$\pi(X_1)$到$\pi(X_2)$的模式转移，我们考虑从$\pi(X_1)$到$\pi(X_4)$的模式转移，然后考虑$\pi(X_2)$到$\pi(X_5)$的模式转移。这样，我们引入了转移模式滞后项，这保证了邻近的时序模式被映射到不含有重叠项的状态向量。

3 数值实验

3.1 将转移置换熵应用于周期性时间序列

对于纯随机的时间序列，转移置换熵的理论值为$T_m = 2log_2(m!)$，这也是转移置换熵的上界。对于单增或单减的时间序列, 只存在一种排序模式, 因此转移置换熵的值$T_m = 0$，这对应了下界。对于其他情况，$0 < H_m < 2log_2(m!)$

我们首先考虑简单的周期性时间序列:

$$x_t = A_1 \sin\left(2\pi\phi_1(t)\right)$$

$t = 1, \ldots, T$. 该时间序列的动态复杂度决定于取样频率 1/ $\phi_1(t)$，但是与振幅A_1是独立的。通常来说，如果我们增大取样频率，动态复杂度将会降低。最极端的情况就是采样频率趋于无限大，这样时间序列就趋于单增或是单减，那么状态向量的排序也趋于单增或是单减的模式。在状态转移模式很少的情况下，系统的动态复杂度将会很低。在图三中，我们设置$\phi_1(t) = t/N$，其中 N 分别取 10，10^2，10^3，10^4.在嵌入维度 m 一定的情况下，转移置换熵的值理论上会随着 N 的增加而减少。由于 m 增大时，将会有更多的状态转移模式出现，因此转移置换熵的值会随 m 增大而增大。在大部分情况下，转移置换熵的值都远小于纯随机时间序列的转移置换熵的值。

sin time series' PTE

m N	10	100	1000	10000
2	1.722289235295665	1.1416948028073173	1.0209457604626824	0.0005774244324753903
3	2.6403784761133666	1.307006180881141	1.0417234698599054	0.0005776556334185062
4	3.597364243857003	1.4596705928377645	1.0625096895075483	0.0005778870195816172

图三：N=10，10^2，10^3，10^4下的正弦函数序列的转移置换熵的值，随着 N 的增加，转移置换熵值减小，动态复杂度降低

sin time series' PE

m N	10	100	1000	10000
2	0.9999926015559886	0.9999995375979904	0.9999995375979904	-0.0
3	1.8576013558349205	1.153721377961066	1.0207769737298586	-0.0
4	2.7767704409087743	1.3069519341120044	1.0415629399533404	-0.0

图四：N=10，10^2，10^3，10^4下的正弦函数序列的置换熵的值

将置换熵和转移置换熵应用于正弦函数序列时，从结果可以看出，在扩大正弦波函数的采样频率时，其动态复杂度会降低，极端情况是采样频率取无限大时，正弦波函数的排序模式只有增加或减少的趋势，由于缺少排序模式的转移，则该时间序列的动态复杂度会非常低。另外，在嵌入维度一定时，转移置换熵的值随着 N 的增加而减少，因为排序模式的转移相对减少。而转移置换熵的值随嵌入维度的增加而增加，因为嵌入维度增加时会出现更多的置换转移模式。在嵌入维度一定时，将正弦波函数的转移置换熵的值比纯随机序列的转移置换熵的值更小。在嵌入维度和 N 一定时，由于转移置换熵衡量的是系统的动态复杂程度，理论上其置换熵的值小于转移置换熵的值，从数值实验中也可看出这一命题是成立的。

3.2 将转移置换熵应用于 logistic 时间序列

我们用转移置换熵来研究 logistic 时间序列

$$x_{t+1} = \mu x_t\left(1 - x_t\right) \qquad \text{对任意 t, } x_t \in [0,1]. \qquad (9)$$

x_t可以用来表示当前人口数与最大环境容纳量的比值。参数μ 的区间为[0,4]. 这个时间序列中包含着复杂且混乱的排序模式。当μ 超过 3.56995 时，该时间序列将表现出混沌时间序列的特征。因此我们分别研究$\mu = 3.6, 3.7, 3.8, 3.9$ 时的情况，并且设样本数据总

量 T=10^5，初始值x_0设为 0.5，通过改变参数μ 的值，我们可以观察到该时间序列的不同特征。

m	2	3	4	5
u=3.6	4	12	16	30
u=3.7	4	15	38	99
u=3.8	4	16	48	148
u=3.9	4	18	62	319
Random	4	36	569	14379

表 1：u=3.6,3.7,3.8,3.9 时的 logistic 序列的置换转移次数和纯随机时间序列的置换转移次数，嵌入维度 m 设为 2，3，4，5. 当 m=5 时，随机时间序列的置换转移数为 14386，略小于理论值 14400，这是由于有效样本效应引起的。

在表 1 中我们分别计算了 logistic 序列在$\mu = 3.6, 3.7, 3.8, 3.9$ 时的不同排序模式转移的数量，同时我们也计算了纯随机时间序列在相同样本量下的不同排序模式的转移数量。嵌入维度 m 分别设为 2，3，4，5. 时滞项$\tau = 2$. 纯随机时间序列的所有可能的转移模式数量为$(m!)^2$. 在实际数据中，转移模式总量并没有达到最大可能值，因此，实验中的实际序列表现出“禁模式”，这意味着一些置换转移模式并没有表现出来。当 m=2 时，所有不同的置换转移模式在每一种情况下都是 4。但是当 m=3,4,5 时，相对于 logistic 序列，随机时间序列的不同置换转移模式数量更多。并且，当μ增加时，不同置换转移数量也在增加。

在图五中，我们给出 logistic 序列和纯随机时间序列的转移置换熵的值。在 m=2 时，两种时间序列的转移置换熵的值差别不大，但是当 m=3,4,5 时，相比于随机时间序列，logistic 序列的转移置换熵的值要小很多。因此，这意味着 logistic 序列的动态复杂程度要比纯随机时间序列的动态复杂程度小很多。因为 logistic 时序结构更加规律，更加确定。同时，这表明转移置换熵可以有效的区分时间序列的混沌性和随机性，而且，在μ增加时，logistic 序列的转移置换熵的值增加，其表现的随机性增强。

logistic time series' PTE

m u	3.6	3.7	3.8	3.9	Random
2	1.0000044268946735	1.2379270822630084	1.4244820429887182	1.5014421699063965	1.9194508828777108
3	2.1244643774629166	2.5439279293735226	2.7074706495472407	2.7932589589618932	4.0853827885989995
4	2.2510877016682542	3.0036314341950696	3.243509168282017	4.056293933937476	6.505549897508927
5	3.3449734343321142	3.9246453689852783	4.1206569859159705	4.977293439369472	9.155717447824829

图五：u=3.6,3.7,3.8,3.9 时的 logistic 序列的转移置换熵的值和随机时间序列的转移置换熵，嵌入维度设为 2，3，4，5.

当 m=3,4,5 时，logistic 的转移置换熵的值显著小于纯随机序列的值，这表明 logistic 时间序列的动态复杂度要小于纯随机序列的动态复杂度。并且与纯随机时间序列不同，logistic 时间序列的时序结构存在一定的规律性，因此转移置换熵可以有效地区分时间序列的混沌特征和随机性特征。

3.3 将转移置换熵应用于金融时间序列

在股票交易中，投资者主要关系股票价格的上升或者下跌。股票价格的波动可以由邻近交易日价格的排序模式反映出来。比如，排序模式为 01 时表示股票价格在上涨，而 10 则表示股票价格下跌。012 表示在两个交易日内股票价格持续上涨。在该部分的数值实验中，将根据以下四个问题来讨论：

(1) 时间序列中哪一种排序模式出现得最频繁，是否存在一种主要的排序模式？

(2) 在时间序列所有排序模式中包含了多少股票市场多少信息？

(3) 时间序列内部的各个排序模式的转移模式是怎样的？在一种确定的排序模式之后最有可能发生哪一种排序模式？

(4) 在时间序列所有的排序转移模式中包含了多少信息？

前两个问题可以由置换熵解决，但是后两个问题是置换熵解决不了的。我们尝试用置换转移熵来解决后两个问题。

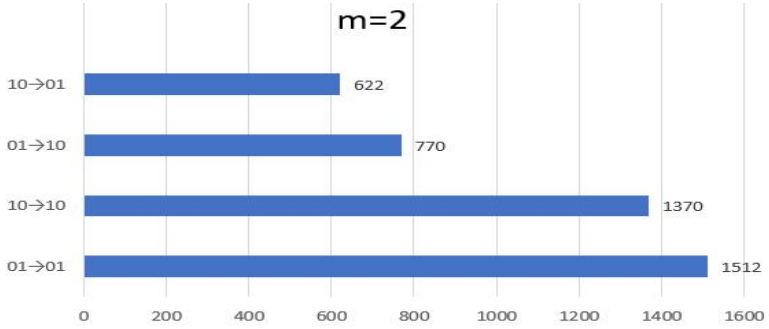
在金融学中，存在着两种著名的投资理论，一种是动量效应，另一种是均值回归效应。动量效应指实际中观察到的资产价格的上涨趋势会持续进行，价格下跌趋势也会持续下去。比如，研究表明，在过去表现强劲的股市（牛市）将会在接下来的一个周期中持续这种强劲态势，并且平均每个月的平均收益率要比表现较差的股市的平均收益率高出大约 1%[14]. N.Jegadeesh 使用买入过去表现较好的股票，卖出过去表现较差的股票，将会在未来 3-12 月的持有周期中获得很可观的收益。另一方面，均值回归效应是指股票价格无论高于或低于长期以来的均值，都会以很高的概率向均值回归的趋势。也有关于股票交易的研究深入调查了市场中存在的均值回归效应[15]. 我们将利用转移置换熵的方法来探究股票市场中是否存在动量效应和均值回归效应。

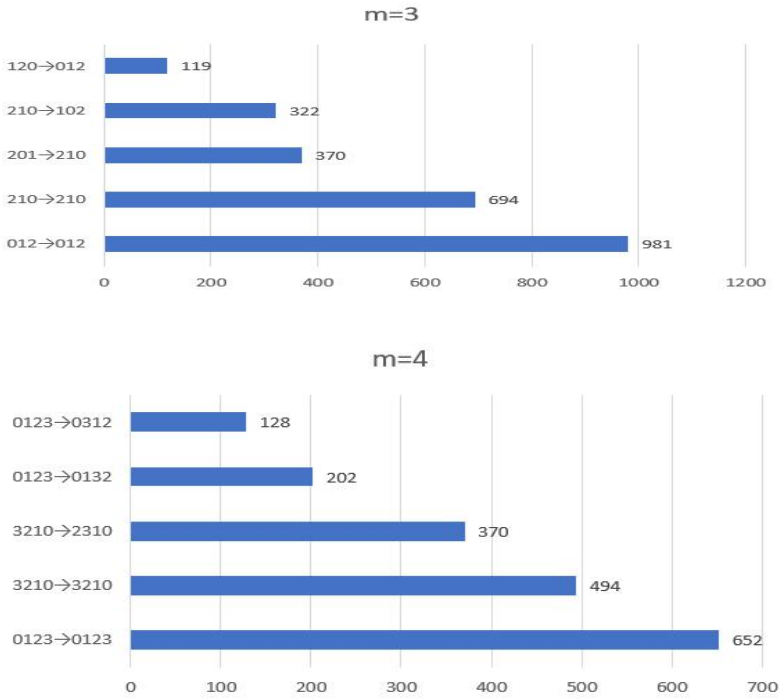
我们分析美国的纳斯达克指数，数据是从 2000.1.3 到 2016.12.30 共 4278 个交易日获取的。

对数收益率的计算公式为

$$\Delta \ln(\mathrm{p}) = \ln\left(p_t\right) - \ln\left(p_{t-1}\right)$$

我们分别将静态置换熵和转移置换熵的方法应用到收盘价和对数收益率的计算来分析纳斯达克指数。





图七：在 m=2,3,4 下纳斯达克指数收盘价序列中发生频率最高的转移置换模式

Nasdaq's PE and PTE

m type	PE	PTE
2	0.9967008712559409	1.9931977335010584
3	2.4935329823618586	3.9703892345400935
4	4.281308028419514	6.030301474410646
5	6.265984542177385	8.149481431947034

LnNasdaq's PE and PTE

m type	PE	PTE
2	0.9991469144390385	1.916169621219617
3	2.5822895643731894	4.066245726339582
4	4.576077131020167	6.477378491298014
5	6.876659780033617	9.032649987204714

图八：纳斯达克指数收盘价和对数收益率的置换熵和转移置换熵的值

从以上结果中可以发现，在嵌入维度为 2 时，01（代表股票价格下跌），10（代表股票价格上涨）几乎是一样多，所以在 m=2 时置换熵的值几乎等于纯随机时间序列对应的置换熵的值。这表明通过前一天的收盘价来预测收盘价的上涨或下跌基本是不可能的，而在 m=3,4,5 时，对数收益率的时序结构比收盘价的时序结构更复杂，动态复杂度

更高，可预测性更低。从静态置换熵的计算中，我们发现在收盘价和对数收益率序列中不存在占据主导地位的排列模式，并且在两个系统中包含的信息接近于纯随机序列包含的信息，因此不可能从之前的收盘价或是对数收益率预测到未来的对应值。因此，通过静态置换熵的方法，我们不能以充足的理由拒绝市场有效性假说。

我们用动态置换熵来研究纳斯达克指数收盘价和对数收益率的时序结构。在 m=2 时，01→01（发生次数为 1512 次）和 10→10（发生次数为 1370 次）模式的出现频率要比 10→01（622 次），01→10（770 次）发生频率高很多。另外，在 m=3 时，置换转移模式发生最多的是 012→012（981 次）而像 012→210 的发生次数只有 19 次，210→012 发生次数只有 23 次。图七具体的反映了以上提到的现象。这表明在纳斯达克指数 2000.1.3-2016.12.13 的收盘价序列中，动量效应体现得比均值回归效应体现得更为突出。利用转移置换熵的方法进行分析后，我们拒绝在纳斯达克指数的收盘价序列中的市场有效性假说，这表明转移置换熵可以揭示时间序列更多的信息，而这些信息是不能被置换熵揭示的。

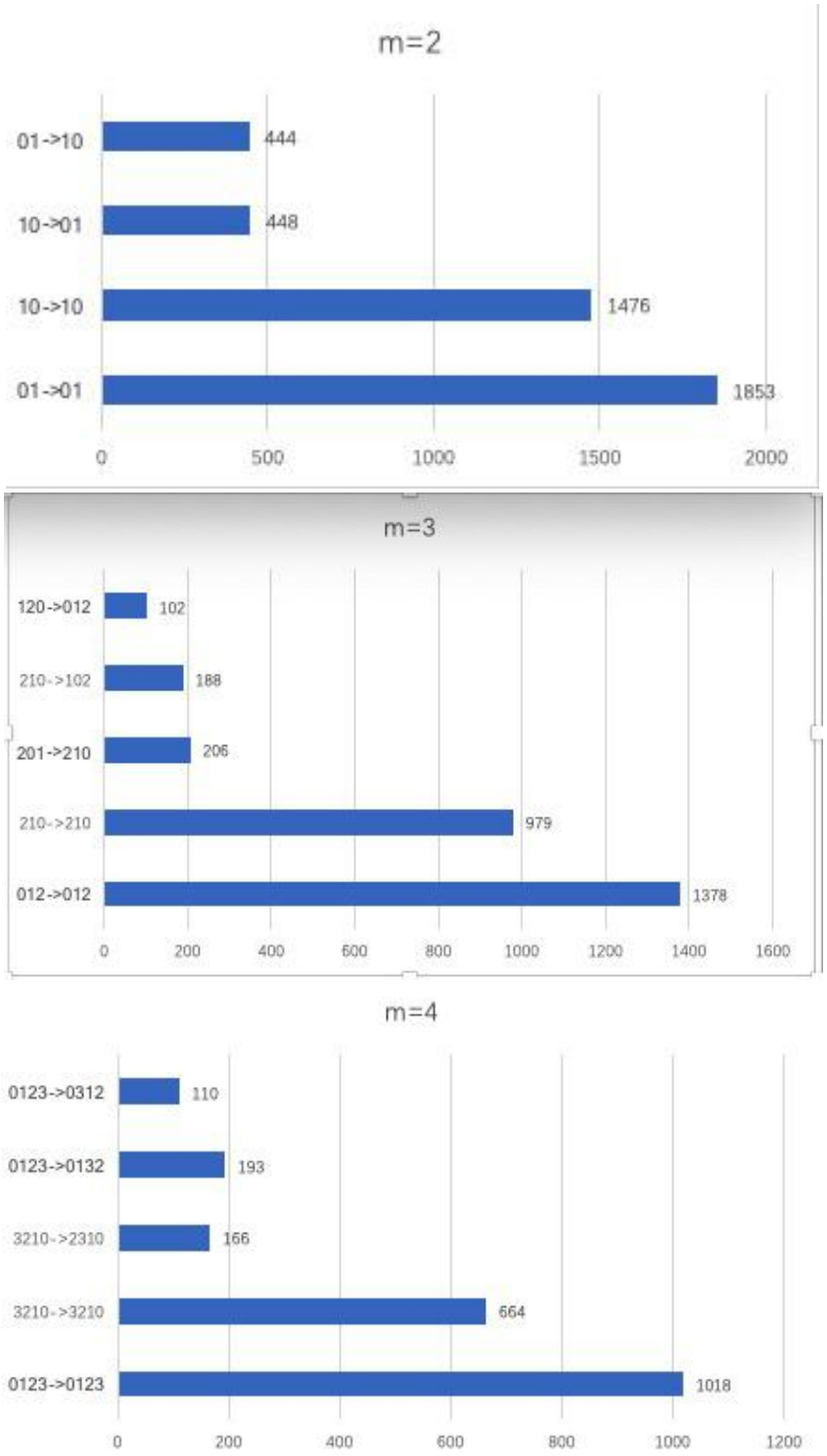
3.4 将多尺度分析应用于转移置换熵方法

将多尺度分析[16]和转移置换熵方法结合起来，具体的，将粗粒化过程应用于转移置换熵的计算中。对于原始时间序列 $\{x_t\}, t = 1, \ldots, T,$ 非重叠的粗粒化后的在尺度 s 下的时间序列为：

$$y_j(s) = \frac{\sum_{i=(j-1)s+1}^{js} x_i}{s}, \text{ i} \leq \text{ j} \leq {}^{\text{T}}\!/_{\text{S}}$$

在尺度为 1 时，时间序列 y 就是原始时间序列，粗粒化过程后的时间序列的长度和原始时间序列的长度在经过尺度因子 s 分解后的长度是相等的，然后可以计算粗粒化后的时间序列的转移置换熵的值。

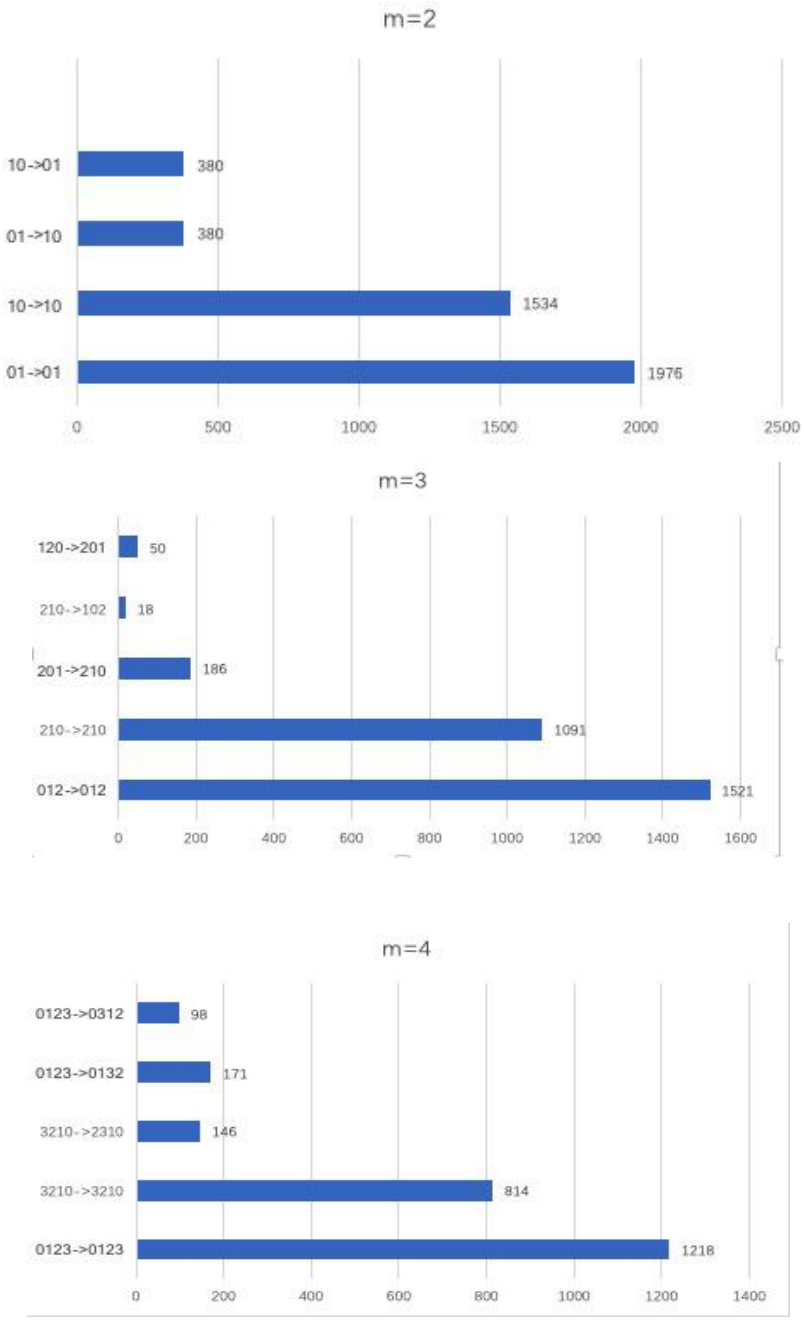
在该部分，我们将多尺度下的转移置换熵方法应用于分析纳斯达克指数的收盘价时间序列，首先，分析粒度为 3 的情况：



图九：在 m=2,3,4，粒度为 3 下纳斯达克指数收盘价发生频率最高的转移置换模式

m type	PE	PTE
2	0.9928150575383796	1.8103516025890602
3	2.0660239812981183	3.126054492753825
4	3.306072403574799	4.515638428222819
5	4.668541004922868	5.9599153883233695

图十：粒度为 3，时滞项为 3 时纳斯达克指数收盘价的置换熵和转移置换熵的值



图十一：在 m=2,3,4，粒度为 4 下纳斯达克指数收盘价发生频率最高的转移置换模式

m type	PE	PTE
2	0.9922956340492695	1.7588045151815477
3	1.9829065599847566	2.9559855285494483
4	3.1081312991088126	4.203161902112904
5	4.324245307974481	5.4672218735707645

图十二：粒度为 4，时滞项为 4 时纳斯达克指数收盘价的置换熵和转移置换熵的值

从以上结果可以发现，在考虑无重叠项之间的转移时，粒度为 4 下的转移熵和转移置换熵的值都分别大于粒度为 3 下的转移熵和转移置换熵的值，这说明在更大的粒度下，收盘价序列的确定性更高，即包含的信息量更大，可预测性更高。在不同的嵌入维度下，粒度更大的收盘价序列表现出的动量效应更加明显，比如嵌入维度为 2 时，粒度为 3 时的 01->01 转移为 1853 次，10->10 转移为 1476 次，而粒度为 4 时的 01->01 转移为 1976 次，10->10 转移为 1534 次。这也表明在粗粒化过程后，收盘价时间序列的确定性增加，且粒度越大，确定性越强，可预测性越高。

4 结论与展望

本文主要利用转移置换熵方法来揭金融示时间序列的动态复杂度。转移置换熵的方法包含了置换熵的符号化分析和条件熵的概念。在转移置换熵的方法中，本文探究了其上界和下界，时间序列中存在的有效样本容量效应，和方法中可能发生的错误置换转移模式统计。然后我们将转移置换熵方法用于分析正弦时间序列和 logistic 时间序列。发现转移置换熵方法能够区分混沌行为和纯随机行为。

在金融时间序列分析中，我们利用较为成熟的纳斯达克指数的收盘价和对数收益率序列进行分析，我们对比了置换熵和转移置换熵方法，发现转移置换熵方法可以解决很多关于动态复杂度的问题，而这些方法是置换熵方法不能解决的。比如在嵌入维度为 2 时，01（代表股票价格下跌），10（代表股票价格上涨）几乎是一样多，所以在 m=2 时置换熵的值几乎等于纯随机时间序列对应的置换熵的值。这表明通过前一天的收盘价来预测收盘价的上涨或下跌基本是不可能的，而在 m=3,4,5 时，对数收益率的时序结构比收盘价的时序结构更复杂，动态复杂度更高，可预测性更低。从静态置换熵的计算中，我们发现在收盘价和对数收益率序列中不存在占据主导地位的排列模式，并且在两个系统中包含的信息接近于纯随机序列包含的信息，因此不可能从之前的收盘价或是对数收益率预测到未来的对应值。因此，通过静态置换熵的方法，我们不能以充足的理由拒绝市场有效性假说。而用动态置换熵来研究纳斯达克指数收盘价和对数收益率的时序结构时，在 m=2 时，01->01（发生次数为 1512 次）和 10->10（发生次数为 1370 次）模式的出现频率要比 10->01（622 次），01->10（770 次）发生频率高很多。另外，在 m=3 时，置换转移模式发生最多的是 012->012（981 次）而像 012->210 的发生次数只有 19 次，210->012 发生次数只有 23 次。即存在占主导地位的排序模式转移。这表明在纳斯达克指数 2000.1.3-2016.12.13 的收盘价序列中，动量效应体现得比均值回归效应体现得更为突出。利用转移置换熵的方法进行分析后，我们拒绝在纳斯达克指数的收盘价序列中的市场有效性假说，这表明转移置换熵可以揭示时间序列更多的信息，而这些信息是

不能被置换熵揭示的。我们发现在纳斯达克指数中存在动量效应，而对数收益率序列的特征和纯随机时间序列的特征非常相似。之后，结合多尺度分析方法，将原始时间序列粗粒化，将转移置换熵应用于分析粗粒化后的时间序列。发现在更大的粒度下，收盘价序列的确定性更高，即包含的信息量更大，可预测性更高。在不同的嵌入维度下，粒度更大的收盘价序列表现出的动量效应更加明显，比如嵌入维度为 2 时，粒度为 3 时的 01->01 转移为 1853 次，10->10 转移为 1476 次，而粒度为 4 时的 01->01 转移为 1976 次，10->10 转移为 1534 次。这也表明在粗粒化过程后，收盘价时间序列的确定性增加，且粒度越大，确定性越强，可预测性越高。转移置换熵方法不需要对时间序列做出任何假设，它可以应用到非平稳和非线性时间序列的分析中，因此，除金融时间序列的研究外，转移置换熵的方法可以应用到更多的领域中。

参考文献

[1] Chen Weiting, Zhuang Jun, Yu Wangxin, Wang Zhizhong. Measuring complexity using FuzzyEn, ApEn, and SampEn.[J]. Medical engineering & physics, 2009, 31(1).

[2] Ke Da-guan. Unifying complexity and information.[J]. Scientific reports,2013,3.

[3] Moulder Robert G,Daniel Katharine E,Teachman Bethany A,Boker Steven M. Tangle: A New Measure of Time Series Complexity.[J]. Multivariate behavioral research,2019.

[4] Richman J S,Moorman J R. Physiological time-series analysis using approximate entropy and sample entropy.[J]. American journal of physiology. Heart and circulatory physiology, 2000, 278(6).

[5] Xiaojun Zhao, Pengjian Shang. Permutation complexity and dependence measures of time series.[J]. Europhysics Letters, 2013, 102(4).

[6] Costa Madalena, Goldberger Ary L, Peng C-K. Multiscale entropy analysis of complex physiologic time series.[J]. Physical review letters, 2002, 89(6).

[7] Luciano Zunino, Massimiliano Zanin, Benjamin M. Tabak,Darío G. Pérez,Osvaldo A. Rosso. Forbidden patterns, permutation entropy and stock market inefficiency[J]. Physica A: Statistical Mechanics and its Applications, 2009, 388(14).

[8] Wolf Alan, Swift Jack B., Swinney Harry L., Vastano John A.. Determining Lyapunov exponents from a time series[J]. North-Holland, 1985, 16(3).

[9] Forsythe A, Nadal M, Sheehy N, Cela-Conde C J, Sawey M. Predicting beauty: fractal dimension and visual complexity in art.[J]. British journal of psychology (London, England : 1953), 2011, 102(1).

[10] Pincus S M. Approximate entropy as a measure of system complexity.[J]. Proceedings of the National Academy of Sciences of the United States of America, 1991, 88(6).

[11] Batty Michael,Morphet Robin,Masucci Paolo,Stanilov Kiril. Entropy, complexity, and spatial information.[J]. Journal of geographical systems, 2014,16(4).

[12] Jianan Xia, Pengjian Shang, Jing Wang, Wenbin Shi. Permutation and weighted-permutation entropy analysis for the complexity of nonlinear time series[J]. Communications in Nonlinear Science and Numerical Simulation, 2016, 31(1-3).

[13] Bandt Christoph,Pompe Bernd. Permutation entropy: a natural complexity measure for time series.[J]. Physical review letters, 2002, 88(17).

[14] Narasimhan Jegadeesh, Sheridan Titman. Profitability of Momentum Strategies: An Evaluation of Alternative Explanations. 2001, 56(2): 699-720.

[15] Shuangjie LI, Xuefeng HU, Liming WANG. Could the Stock Market Adjust Itself? An Empirical Study Based on Mean Reversion Theory[J]. Journal of Systems Science and Information, 2020, 8(02):97-115.

[16] Xiaojun Zhao, Yupeng Sun, Xuemei Li, Pengjian Shang. Multiscale transfer entropy: Measuring information transfer on multiple time scales[J]. Communications in Nonlinear Science and Numerical Simulation, 2018, 62.

附　　录

附录 A　Python 程序代码

```
# encoding=utf8
from functools import reduce
from itertools import permutations

import math
import numpy as np
import prettytable as pt
import xlrd


def CalculatePE(stocks, m):  # 计算静态置换熵
    numbers = [i for i in range(m)]
    permutation = list(permutations(numbers))  # 将 m 个数全排列得到列表
    count = [0] * len(permutation)  # 用来计每种置换模式数的列表
    for i in range(len(stocks) - m):  # 开始计数
        x = np.array(stocks[i:i + m])
        count[permutation.index(tuple(np.argsort(x)))] += 1
    P = [i / (len(stocks) - m) for i in count]  # 计算概率
    # for i, c in enumerate(count):  # 展示每种模式的数量
    #     print(permutation[i], ":", c)
    PE = -reduce((lambda x, y: x + y), [i * math.log2(i) for i in P if i != 0])  # 计算
PE
    return PE


def CalculatePTE(stocks, m):  # 计算动态置换熵
    numbers = [i for i in range(m)]
    permutation = list(permutations(numbers))
    count = [0] * len(permutation) ** 2
    for i in range(len(stocks) - m - 1):  # 开始计数
        x = np.array(stocks[i:i + m])
        y = np.array(stocks[i + 1:i + m + 1])
        count[permutation.index(tuple(np.argsort(x))) * len(permutation) +
permutation.index(tuple(np.argsort(y)))] += 1
    P = [i / (len(stocks) - m) for i in count]  # 计算概率
    # for i, c in enumerate(count):  # 展示每种模式的数量
```

```
    #             print(permutation[i // len(permutation)], "->", permutation[i %
len(permutation)], ":", c)
    PTE = -reduce((lambda x, y: x + y), [i * math.log2(i) for i in P if i != 0])  # 计
算 PE
    return PTE


def generate_sin(N):  # 生成 sin 时间序列对应的列表
    number_points = 2500  # 产生点的数量，可自定义
    series = [math.sin(2 * math.pi * i / N) for i in range(number_points)]
    return series


def generate_logistic(u):  # 生成 logistic 时间序列对应的列表
    T = 10 ** 5  # 产生点的数量，可自定义
    series = [0] * T
    series[0] = 0.5
    for i in range(T):
        if i == 0:
            continue
        series[i] = u * series[i - 1] * (1 - series[i - 1])
    return series


def display_sin():  # 展示 sin 时间序列的 PT 与 PTE 表格
    print()
    print("sin time series' PE")
    tb = pt.PrettyTable()
    tb.field_names = ["m N", "10", "100", "1000", "10000"]
    for i in range(2, 5):
        tb.add_row(
            [i,                    CalculatePE(generate_sin(10),                    i),
CalculatePE(generate_sin(100), i),
             CalculatePE(generate_sin(1000),                    i),
CalculatePE(generate_sin(10000), i)])
    print(tb)

    print()
    print("sin time series' PTE")
    tb = pt.PrettyTable()
    tb.field_names = ["m N", "10", "100", "1000", "10000"]
    for i in range(2, 5):
        tb.add_row(
```

```
            [i,                 CalculatePTE(generate_sin(10),                 i),
CalculatePTE(generate_sin(100), i),
             CalculatePTE(generate_sin(1000),                                  i),
CalculatePTE(generate_sin(10000), i)])
    print(tb)


def display_logistic():  # 展示 logistic 时间序列的 PT 与 PTE 表格
    print()
    print("logistic time series' PE")
    tb = pt.PrettyTable()
    tb.field_names = ["m u", "3.6", "3.7", "3.8", "3.9"]
    for i in range(2, 5):
        tb.add_row(
            [i,                 CalculatePE(generate_logistic(3.6),            i),
CalculatePE(generate_logistic(3.7), i),
             CalculatePE(generate_logistic(3.8),                               i),
CalculatePE(generate_logistic(3.9), i)])
    print(tb)

    print()
    print("logistic time series' PTE")
    tb = pt.PrettyTable()
    tb.field_names = ["m u", "3.6", "3.7", "3.8", "3.9"]
    for i in range(2, 5):
        tb.add_row(
            [i,                 CalculatePTE(generate_logistic(3.6),           i),
CalculatePTE(generate_logistic(3.7), i),
             CalculatePTE(generate_logistic(3.8),                              i),
CalculatePTE(generate_logistic(3.9), i)])
    print(tb)


def display_stock(name, stock):  # 展示某一股票的 PT 与 PTE 表格
    print()
    print(name + "'s PE and PTE")
    tb = pt.PrettyTable()
    tb.field_names = ["m type", "PE", "PTE"]
    for i in range(2, 6):
        tb.add_row(
            [i, CalculatePE(stock, i), CalculatePTE(stock, i)])
    print(tb)
```

```
if __name__ == "__main__":
    amounts = 1218
    workbook = xlrd.open_workbook("test.xlsx")
    booksheet = workbook.sheet_by_index(0)  # 用索引取第一个 sheet
    stocks = {}
    for i in range(booksheet.ncols):
        if i == 0:  # 跳过第一列（日期）
            continue
        elif i == 1:
            continue
        elif i == 2:
            continue
        elif i == 3:
            continue
        elif i == 4:
            continue
        stocks[booksheet.col_values(i)[0]] = booksheet.col_values(i)[1:]
    #display_sin()  # 展示 sin 时间序列的 PT 与 PTE 表格
    # display_logistic()  # 展示 logistic 时间序列的 PT 与 PTE 表格
    display_stock('LnNasdaq', stocks['LnNasdaq'])
```