机器学习研究杂志 24 (2023) 1-36 10/21提交；12/22修改；3/23发表

通过功能修剪CUSUM统计的快速在线变化点检测

加埃塔诺-罗曼诺 G.ROMANO@LANCASTER.AC.UK
兰卡斯特大学数学和统计系、
Lancaster, United Kingdom, LA1 4YF
伊德里斯-A-埃克雷 I.ECKLEY@LANCASTER.AC.UK
兰卡斯特大学数学和统计系、
Lancaster, United Kingdom, LA1 4YF
保罗-费恩黑德
兰卡斯特大学数学和统计系、 P.FEARNHEAD@LANCASTER.AC.UK
Lancaster, United Kingdom, LA1 4YF
Guillem Rigaill
巴黎-萨克雷大学、法国国家科学研究中心、法国国家农业研究院、埃弗里大学、巴黎-萨克雷植物科学研究所（IPS2）、 GUillem.rigaill@inrae.fr
91405, Gif-sur-Yvette, 法国
和
Eval d'Esson大学，CNRS，INRAE、
Laboratoire de Math´ematiques et Mod´elisation d'Evry,
91000, Evry, France

编辑：Zaid Harchaoui

摘要

许多在线变化点检测的现代应用需要处理高频观测的能力，有时可用的计算资源有限。检测平均数变化的在线算法通常涉及使用移动窗口，或指定预期变化的大小。这样的选择会影响到算法对哪些变化有最大的检测能力。我们介绍了一种算法，即功能性在线CuSUM（FO- CuS），它相当于对所有大小的窗口或所有可能的变化值同时运行这些早期方法。我们的理论结果对FOCuS的每次迭代的预期计算成本给出了严格的约束，这与观察值的数量成对数关系。我们展示了FOCuS如何应用于一些不同的平均变化情况，并通过其在检测计算机服务器数据中的异常行为的最先进的性能来证明其实际效用。**关键词：**断点；变化点；CUSUM；在线；实时分析；FPOP。

1. 简介

在过去的十年中，我们见证了变化点算法的复兴，现在可以看到它们在许多现实世界的应用中发挥着作用。大多数的

目前的文献着重于*事后*分析，在观察了一系列的数据之后。这种方法通常被称为*离线*变化点检测。然而，随着技术的发展，近年来一些领域对*在线*变化点检测程序的需求急剧增加。例如，但不限于IT和网络安全（Jeske等人，2018；Tartakovsky等人，2012；Peng等人，2004）；检测天文学中的伽马射线暴（Fridman，2010；Fuschino等人，2019）；检测地震震感（Popescu和Aiordˇachioaie，2017；Xie等人、2019年）；工业过程监测（Pouliezos和Stavrakakis，2013年）；检测不良健康事件（Clifford等人，2015年）；以及监测飞机的结构完整性（Alvarez-Montoya等人，2020年；Basseville等人，2007）。

在线设置带来了离线变化点检测中所没有的计算挑战。一个程序需要是连续的，也就是说，我们应该在观测结果可用时对其进行处理，在每次迭代时，我们应该根据迄今为止的信息来决定是否标记一个变化点。该程序还需要能够在有限状态机上运行无限量的迭代，也就是说，在内存中是恒定的。最后，一个程序最好能够处理观察结果，至少在平均水平上，在观察结果到达时能够迅速处理。许多在线变化点应用设置有高频率的观察，而且有些也有有限的计算资源。例如，2015年PhysioNet挑战中的心电图数据（Clifford等人，2015）的采样频率为240Hz，而探测伽马射线暴的方法（Fuschino等人，2019）需要处理高频观测数据，并且能够在微型卫星上的小型计算机上运行。在线变化点算法的一个挑战是，在满足这种编译限制的同时，仍然具有接近最佳的统计特性。本文正是在这种情况下考虑单变量平均变化问题。

目前具有线性计算成本的在线变化点方法包括Page（1955）的方法，该方法假定对变化前和变化后的平均值都有了解；或者移动窗口方法，如MOSUM。假设变化前均值是已知的，这在许多应用中是合理的，因为会有大量的数据来估计这个均值（不过见G¨osmann等人，2019年的讨论）。然而，如果假设的变化大小是错误的，Page（1955）的方法会失去力量。例如，这种方法对检测小于假设变化大小的一半的变化几乎没有能力。同样，如果窗口的大小与变化的大小不合适，移动窗口方法也会表现得很差。例如，小的窗口尺寸会导致检测小变化的能力不足，而太大的窗口会导致检测较大变化的延迟。关于这些问题的例子见第2.1节。

另一种具有更稳健统计特性的方法是，例如，应用一个移动窗口，但考虑所有可能的窗口大小。在已知变化前均值的情况下，这被称为Page-cusum方法（Kirch和Weber，2018），也是Yu等人（2020）在未知变化前均值情况下的方法。Yu等人（2020）的理论结果证明了这种方法的优秀统计特性。然而，目前这个想法的精确实现，其每次迭代的计算成本与观测值的数量成线性关系，因此总体上有一个二次计算成本。推导出一个能够解决PAGE-CUSUM统计的有效实现的问题可以追溯到上个世纪。Basseville和Nikiforov（1993）指出："[.....]GLR[广义似然比--Page的方法]算法在计算上是很复杂的。

plex"，然后提出了一些近似值。最近，Yu等人（2020）评论了开发具有良好统计保证的快速算法的挑战。我们介绍了一种以有效方式精确计算GLR的算法，我们称之为功能在线CuSum（FOCuS）。我们开发的FOCuS用于检测高斯模型下单变量数据中均值的变化，它可以应用于变化前均值已知或未知的情况下。FOCuS递归地更新一个分片二次方，其最大值为变化的测试统计量。我们表明，解决递归的摊销成本在每次迭代中是恒定的。最大化函数的计算成本与片断二次函数中的分量数量成正比，而且分量的平均数量随着观测值数量的对数增加。我们可以通过在必要时将最大化限制在固定数量的分量上，获得每次迭代计算量不变的算法。我们开发了一个这样的FOCuS的近似版本，并根据经验表明，它的统计性能与精确实现几乎相同。在未知变化前均值的情况下，FOCuS实现了Yu等人（2020）的方法，而我们对FOCuS的精确实现可以在普通个人计算机上以不到一秒的时间分析100万个观测值。

许多关于在线变化点方法的研究都关注如何实施这些方法，以使它们在无变化的无效假设下具有良好的特性表现。有两个不同的标准来量化一个方法在无效假设下的行为，一个是平均运行长度（Reynolds, 1975），这是我们检测到变化之前的预期观测次数。另一个是显著性水平--如果该方法在无限长的数据上运行，检测到变化的概率。在实践中，这两个标准会影响检测方法的阈值选择。如果我们希望控制显著性水平，那么我们需要一个随着观测值的增加而增加的阈值（例如，见Kirch和Kamgaing，2015），而如果我们希望控制平均运行长度，我们可以使用一个固定的阈值。FOCuS算法可以使用这两种方法--但是为了简单起见，我们在本文中只使用一个固定的阈值。

本文的大纲如下。在第2节中，我们考虑了在假设变化前均值已知的情况下检测单变量均值变化的挑战。我们提出了FOCuS算法，该算法可以被看作是对所有可能的变化大小同时执行Page(1955)的程序。我们的主要理论结果表明，FOCuS在实现这一目标时，每次迭代的平均计算成本与数据点的数量成对数关系。我们对每次迭代的平均成本的约束是很严格的，处理第100万个观测值的成本大致相当于评估15个二次方的成本。在第3节中，我们研究了变化前平均值未知的情况下的FOCuS。这个算法可以被看作是实现Yu等人（2020）的统计检验。有趣的是，他们的算法要么是精确的，但每次迭代的成本是线性的，要么是近似的，成本是迭代次数的对数，而FOCuS算法既是精确的，又有每次迭代的平均成本是对数的。我们没有介绍FOCuS的任何统计理论，因为这在Yu等人（2020）中已经涉及。在第4节中，我们介绍了FOCuS算法对检测异常值存在时的变化情况的扩展。在第5节中，我们展示了FOCuS在一些AWS Cloudwatch服务器实例上的监控应用。最后，本文以讨论结束，其中包括评估在多个数据流上独立运行FOCuS的用途，然后结合结果来检测可能共同影响多个数据流的变化。

实现FOCuS的软件和我们的模拟研究的代码可在以下网站获得 https://github.com/gtromano/FOCuS。

2. 已知变化前的平均值

2.1 问题的设置和背景

考虑检测单变量数据中均值变化的问题。我们将让x_t表示时间t的数据，$t = 1, 2, \ldots\ldots$。我们对在线检测感兴趣，也就是说，在观察了每个新的数据点之后，我们希望决定是否标明发生了变化。我们首先假设变化前的平均值是已知的。通常情况下，下面的方法在实践中是通过使用从训练数据中计算出来的变化前平均值的插件估计器来实现的。

虽然有许多不同的方法来在线检测变化（见Veeravalli和Banerjee，2014，一些例子），但一个常见的方法（见Kirch和Weber，2018）是使用分数统计的累积和，也被称为基于CUSUM的程序。假设我们将数据建模为来自密度为$f(x;\mu)$的参数模型，并将变化前的平均值表示为μ_0。将观测值x的得分统计量定义为

$$H(x, \mu) = \frac{\partial \log f(x; \mu)}{\partial \mu}.$$

那么如果在时间n之前没有变化

$$\mathrm{E}\left(H(X_i, \mu_0)\right) = 0 \text{，对于} i = 1, \ldots, n.$$

因此，在时间n之前的变化证据可以通过监测这些分数统计的部分和的绝对值来获得，我们将其表示为

$$S(s, n) = \sum_{i=s+1}^{n} H(x_i, \mu_0)。$$

我们的想法是，如果没有变化，这些部分之和应该接近于0，反之则是
如果有变化，则偏离零。

为了便于表述，并使想法具体化，在下文中我们将考虑我们有一个单位方差的高斯模型的数据的情况。在这种情况下，$H(x, \mu) = (x-\mu)$。另外，由于我们假设μ_0是已知的，那么在不丧失一般性的情况下，我们可以设定$\mu_0 = 0$。

一直以来，我们可以监测的部分和的选择有很多不同。对于检测观察x后的变化$_n$，Kirch和Weber（2018）强调了以下统计数据：

$$\text{CUSUM} \qquad C(n) = \sqrt{\frac{1}{n}} |S(0, n)|; \tag{1}$$

$$\text{MOSUM} \qquad M_w(n) = \sqrt{\frac{1}{w}} |S(n - w, n)|; \tag{2}$$

$$\text{mMOSUM} \qquad M_k^{(m)}(n) = \sqrt{\frac{1}{nk}} |S(n - \lfloor kn \rfloor, n)|; \tag{3}$$

$$\text{页码-CUSUM} \qquad P(n) = \text{最大}_{0 \le w < n} \sqrt{\frac{1}{w}} |S(n - w, n)|。 \tag{4}$$

每种情况下的比例因子都是为了使累积总和$S(-, -)$正常化，目的是使其方差标准化。在每种情况下，我们都会将时间n的统计量与一些适当的阈值进行比较，如果统计量高于阈值，则检测n之前的变化。正如介绍中所讨论的，阈值的选择会影响到无效情况下测试的属性。可以选择恒定的阈值或随着n的增加而增加的阈值（Kirch和Weber，2018），但为了简单起见，我们将全程使用恒定的阈值。

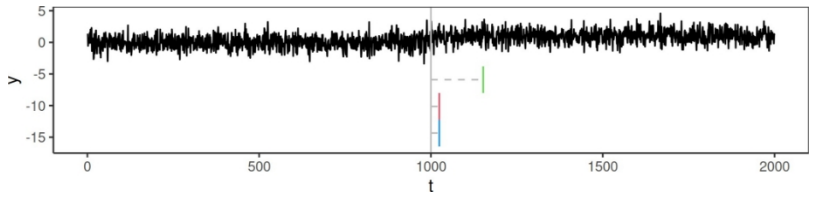
对于MOSUM程序（Eiauer和Hackl，1978；Chu等人，1995）和mMOSUM程序（源于Chen和Tian，2010），我们需要指定一个调整参数。MOSUM方法使用最近的$w>0$观察值的窗口上的部分和，而mMOSUM固定了一些比例$0<k<1$，并使用最近的观察值比例k的部分和。这三个统计量都是在线的，在我们处理每一个新的数据点时，只有一个$O(1)$的统计量更新。Page-CUSUM在所有可能的部分总和上实现最大化，在时间n结束时。

为了说明CUSUM、MOSUM和PAGE-CUSUM之间的区别，我们实现了这些方法，以检测在时间t对某一平均数μ_1，不同的t和μ_1。我们计算了在我们的模型下不同的测试统计量，其中数据是独立高斯的单位方差。在图1中，我们比较了每个统计量的检测延迟。在图1a中，我们模拟了1000次观测后的变化数据。选择变化的大小是为了给MOSUM程序的窗口大小提供高功率。MOSUM和PAGE-CUSUM倾向于快速检测出一个变化。在第二个例子中，如图1b所示，我们减少了变化的幅度。在这里，MOSUM测试大大降低了功率，未能检测到变化。虽然增加窗口可以检测到变化，但在第一个例子中，使用更大的窗口尺寸会增加检测延迟。在我们的最后一个例子，图1c中，我们有与第一个例子相同的变化大小，但是，现在的变化发生在8000个观测值之后。在这种情况下，我们看到CUSUM统计量的表现很差。这是因为CUSUM统计量必须对变化后的数据与变化前的所有数据进行平均，这就降低了测试统计量的力量。当变化之前有大量数据时，情况尤其如此。MOSUM和PAGE-CUSUM的表现都与第一个例子相同。虽然我们没有显示mMOSUM在这些例子中的表现，但是它对窗口比例k的选择具有类似MOSUM对窗口大小的敏感性。

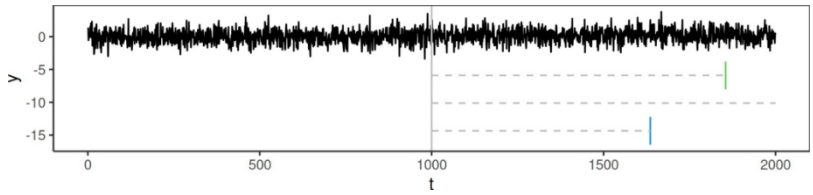
Pag-CUSUM方法试图避免在MOSUM方法中选择窗口大小的问题，并等同于在w上最大化MOSUM统计量。事实上，计算$\max_{0\le s<n} |S(s, n)|$的计算成本随$n$线性增加，导致$O(n^2)$的计算复杂性。

$_t$检测变化的另一种方法（Page, 1954, 1955）是基于在变化后均值的假设值下依次应用似然比测试，即μ_1。在我们的高斯模型下，变化前的均值为0，方差为1，如果数据的均值为μ_1，而非均值为0，则单个数据点的对数似然比统计量的贡献是对数似然比差异的两倍：

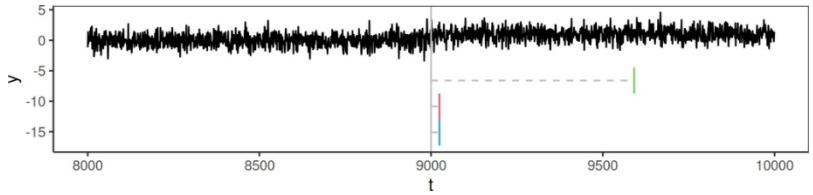
$$LR(x_t, \mu_1) = 2\mu_1 \left(x_t - \frac{\mu_1}{2}\right)$$



(a)



(b)



(c)

图1：CUSUM（绿色）、MOSUM（红色，w=50）和PAGE-CUSUM（蓝色）对三个序列的检测延迟。这些序列是以下列方式产生的：(a)一个由2000个观察值组成的序列，在1000处有一个大小为1的变化；(b)类似于
(a)但平均数变化为0.2；(c)再次类似(a)，但在序列开始时增加了8×10^3 的观测值。阈值根据第2.3节的模拟研究进行了相应的调整。灰色实线是指真实的变化点位置，虚线是指程序的检测延迟。

在这种情况下很常见，我们将使用对数似然比统计量的一半的测试统计量（尽管显然使用这种测试统计量等同于使用对数似然比测试统计量）。因此，在时间n，我们对时间s的变化的测试统计量是$t = s + 1, \ldots\ldots$的$LR(x_t, \mu_1)$项之和的一半。由于我们不知道变化的时间，我们对s进行最大化：

$$Q^{n,\mu_1} = \max_{0\le s\le n} \sum_{t=s+1}^{n} \mu_1 \left(x_t - \frac{\mu_1}{2} \right),$$

其中我们使用的惯例是，从s=n+1到n的总和为0。 我们将这个统计量称为*顺序-页统计量*。

虽然$Q_{n,\mu}$ 1涉及到n个项的和，但佩奇（1954）表明，我们可以计算出
$Q_{n,\mu}$ 1在恒定时间内递归为：

$$Q_{n,\mu_1} = \max\left(0, Q_{n-1,\mu_1} + \mu_1 x_n - \frac{\mu_1}{2}\right). \quad (5)$$

顺序-页统计的一个问题是需要指定μ_1，如果对μ_1选择不当，会大大降低检测变化的能力。这与MOSUM的窗口大小的选择类似。为了部分克服这个问题，在这两种情况下，我们可以针对窗口大小或μ_1的网格，多次实施这些方法。显然，这也会增加计算成本。

Lorden（1971）提出了一个扩展Page方法的程序，以测试所有大于最小变化规模的变化。为了便于阐述，假设我们希望测试一个正的变化，类似的方法适用于一个负的变化。让μ_1是最小变化规模。Lorden (1971)使用的统计量是所有大于μ_1的变化的顺序-Page统计量的最大值。这个程序是基于对μ_1运行Page的方法，并记录重置时间，也就是顺序-Page统计量为0的时间。然后，Lorden（1971）利用这样一个属性：在任何时间t，顺序-Page统计量的最大值将是时间s变化的一半对数似然比统计量的最大值，我们在所有大于或等于最近的重置时间的s上进行最大化。在时间t上的计算成本与上次重置后的时间点数量成正比。

这种想法与我们接下来介绍的FOCuS^{0}算法有一些相似之处，即它试图巧妙地选择要进行的测试。但是FOCuS^{0}不需要指定最小变化大小；而且FOCuS^{0}在决定计算哪些测试方面更有效率，因此它的计算效率更高（除非Lorden算法的最小变化大小在0.5左右或者更高，见附录C.1）。

2.2 FOCuS^0：解决所有μ的Page递归问题$_1$

我们的想法是对变化后平均值的所有值同时解决顺序-页递归。为此，我们用变化后均值μ的函数$Q_n(\mu)$来重写(5)。然后我们有$Q_0(\mu) = 0$，对于$n = 1, \ldots,$

$$Q_n(\mu) = \max\left\{0, Q_{n-1}(\mu) + \mu x_n - \frac{\mu}{2}\right\}. \tag{6}$$

然后我们将使用最大$_\mu Q_n(\mu)$作为我们的测试统计量。可以直接看到，对于任何μ_1，$Q_n(\mu_1) = Q_{n,\mu_1}$。因此，如果我们能够有效地计算函数$Q_n(\mu)$，那么我们的测试统计量就等同于在所有可能的变化后平均值的选择上的序贯-页统计量的最大值。此外，下面的命题（例如见Basseville和Nikiforov, 1993中的例2.4.3）表明，这个测试统计量等同于Page-CUSUM统计量（4），或者等同于MOSUM统计量（2）在所有可能窗口上的最大值。

命题1 *$Q_n(\mu)$的最大值满足以下条件*

$$\max_\mu Q_n(\mu) = \frac{1}{2}P(n)^2 = \frac{1}{2}\max_w M_w(n)^2,$$

其中$P(n)$是Page-CUSUM统计量，$M_w(n)$是窗口大小为w的MOSUM统计量。

这方面的证明可以在附录A中找到。

算法1：FOCuS^0（一次迭代）。

数据：x_n 时间n的数据；$Q_{n-1}(\mu)$ 前次迭代的成本函数。

输入：$\lambda > 0$

$Q_n(\mu) \leftarrow \max\left\{0, Q_{n-1}(\mu) + \mu x_n - \frac{\mu^2}{2}\right\}$ ；// 算法2：摊销的$O(1)$

$Q_n \leftarrow \max_\mu Q_n(\mu)$ ； // 定理4：平均$O(\log(n))$

如果$Q_n \geq \lambda$ **那么**

  返回n*作为一个停止点*；

结束

返回$Q_n(\mu)$，*用于下一次迭代。*

在算法1中给出了对所产生的在线变化点检测算法的描述。我们将其称为功能性在线CuSUM（FOCuS）算法。为了能够将这个假设有已知变化前均值的版本与我们在下一节介绍的版本区分开来，我们将算法1称为FOCuS^0。FOCuS^0 算法只有在计算效率高的情况下才有用，特别是如果我们能够有效地实现步骤1和2。这些步骤对应于解决(6)中的递归，从$Q_{n-1}(\mu)$中获得函数$Q_n(\mu)$，然后使函数$Q_n(\mu)$最大化。下面，我们依次描述这些步骤，并介绍其平均计算成本的结果。

2.2.1 步骤1　更新区间和四分法

:

对于算法1的第1步，我们建议分别更新$\mu>0$和$\mu<0$的函数$Q_n(\mu)$。这些可以以相同的方式更新，所以我们将只描述$\mu>0$的更新。我们将利用以下事实：(6)将分片二次函数映射到分片二次函数，因此$Q_n(\mu)$将是分片二次函数（见Maidstone等人，2017，关于类似想法）。并可将其作为μ的有序区间的列表，连同μ的系数一起存储。

在该区间上对$Q_n(\mu)$的二次方。让$S_t = \Sigma^t_{j=1} x_j$ 是前t个数据点的总和。

在时间n，在迭代τ引入的二次方将是这样的形式

$$\mu \sum_{t=\tau+1}^{n} x_t - (n-\tau)\frac{\mu}{2} \;\; != \;\; \mu\,(S_n - S_\tau) - (n-\tau)\frac{\mu}{2}\,. \tag{7}$$

因此，如果在时间n，我们知道n和S_n，如果我们存储τ和S_τ，它的系数就可以计算出来。这种为二次元存储的信息不需要在每次迭代时更新。

因此，在任何时间t，我们能够通过为定义片断二次方$Q_t(\mu)$的每一个k_t二次方存储一组三联体(τ_i, s_i, l_i)，来产生一个函数$Q_t(\mu)$的紧凑摘要。这些条目是τ_i，即第1个二次方程被引入的时间，$s_i = S_\tau$ i，即截至τ_i的观测值之和，以及l_i，即第1个二次方程为最优的μ区间的左手点。二次方程的排序是这样的：$0 = l_1 < \cdots < l_k$ t，因此对于$i < k_t$，第1个二次方程对于$\mu \in [l_i, l_{i+1})$是最优的，第k_t个二次方程对于$\mu \in [l_k t, \infty)$是最优的。此外，如果$l_i < l_j$，那么$\tau_i < \tau_j$，因为最近引入的二次方程比最近引入的二次方程对较大的μ值是最优的。(这一点可以证明，因为所有的二次方程都经过原点，第1个二次方程的μ^2 系数将大于*第*j个二次方程的系数）。

现在，考虑一个迭代，从函数$Q_n(\mu)$计算出函数$Q_{n-1}(\mu)$。这在算法2中给出。我们目前将储存与定义$Q_{n-1}(\mu)$的四元数相关的k_{n-1}三元数。我们将递归分成两部分。首先我们计算中间函数$Q^*(\mu) = Q_{n-1}(\mu) + \mu(x_n - \mu/2)$。这将更新定义$Q_{n-1}(\mu)$的$k$个$_{n-1}$四元数中的每个系数。然而，由于我们是根据数据点之和的汇总统计来定义四元数的（7），这将通过更新所有数据点之和$S_n = S_{n-1} + x_n$来实现。

其次，我们计算$\max\{0, Q^*(\mu)\}_n$。要做到这一点，我们首先添加一个对应于零线的二次方。这将有三重（n, S_n, l），对于一些l，使Q^*_n（l）=0，即我们需要计算的。

一个关键的观察是，在迭代τ时引入的二次元之间的差异和零线给出，零线在区间上的效果更好

$$2\sum_{t=\tau+1}^{n}\frac{x_t}{(n-\tau)},+\infty \;=\; 2\frac{S_n - S_\tau}{(n-\tau)},+\infty \quad . \tag{8}$$

考虑到所有的τ，我们可以得到，零线比其他所有的线都好。

$$2\max_{\tau}\frac{S_n - S_\tau}{(n-\tau)},+\infty \text{ 。}$$

因此，我们可以看出，定义零线的三联体的最终成分是

$$l = 2\max_{\tau}\frac{S_n - S_\tau}{(n-\tau)}\text{。} \tag{9}$$

如果我们能计算出l，那么马上就会发现任何$l_k > l$的二次方都可以被删除。而任何$l_k < l$的二次函数将不受影响（它们的最优区间可能会被改变，但需要存储二次函数的三联体将不受影响）。通过使用上面提到的二次方程的排序，我们通过依次比较每个二次方程的零线来找到l，从第k_{n-1}个二次方程开始，按递减的顺序逐个进行。如果我们考虑的是第1个二次方，我们检查$Q^*(l_i)$是否<0。这相当于检查在l_i，第i个二次方是否小于0。如果是，我们就删除这个二次方，并移到第$(i\ 1)$个二次方（如果$i = 1$，则停止）。如果$Q^*(l_i) > 0$，那么$l > l_i$，我们找到l作为μ的正值，使第i个二次函数等于0。

图2显示了该算法的图示。l的条件与随机行走$S_t, t = 1, \ldots$的斜率有关如图3中的一个例子所示，我们保留的四边形与这个随机行走的下凸壳的顶点的一个子集有关（这个属性在定理4的证明中被正式证明），这个属性在约束算法的计算复杂度方面将是有用的。此外，算法2与Melkman的寻找一组点的凸壳的算法密切相关（Melkman，1987）。

我们可以证明，算法2的摊销后的每次迭代成本为$O(1)$。直观的理解是，每个二次元都被添加一次和删除一次，其他方面没有变化。因此，每次迭代的平均成本基本上是添加和删除一个二次方的成本。

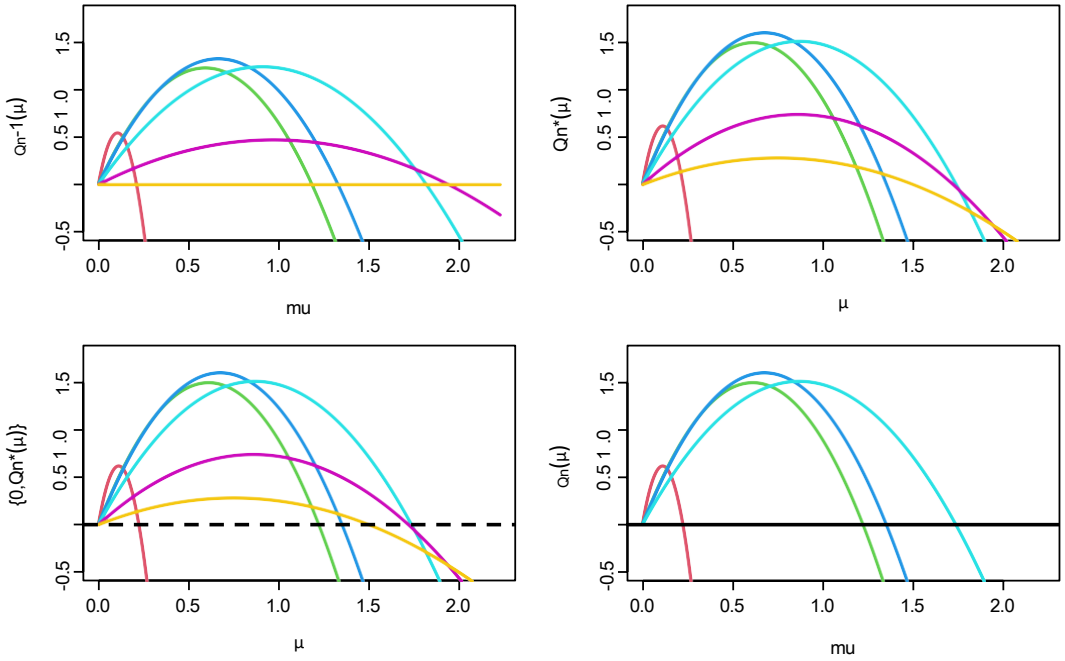


图2：FOCuS的一次迭代实例[0]。(左上)时间n-1的输出是函数$Q_{n-1}(\mu)$，它是一组四元数的最大值。在这个例子中，它是6个二次方程的最大值，其中一个是零线。在该算法中，每个二次方由一个三联体（τ, s, l）表示，其元素是二次方被引入的时间、该时间的观测值之和，以及它是最优的最小的μ值。一个关键的属性是，这些都是有序的，所以早期引入的二次方程具有较高的μ系数[2]，因此对于较低的μ值来说是最优的。（右上）函数$Q^*(\mu) = Q_{n-1}(\mu) + \mu(x_n - \mu/2)$。这仍然是六个二次方程的最大值，但系数与左上图相比有所变化。在实践中，这涉及到不需要更新四元组的三元组。(左下角）我们引入零线（黑色虚线$_n$），这是在时间n引入的二次方程。我们将零线与当前的每个二次方程进行比较，看哪个在二次方程最优的最低μ值时较大，从最近的二次方程开始（即黄色，然后是紫色等）。如果该二次方程低于零线，它就不再是最优的，可以被删除。对于第一个高于零线的二次函数，我们计算二次函数和零线相交处的非零值μ。这个值和0的最大值是最小的μ值，对它来说，零线是最优的。(右下角）去除两个不再是最优的二次函数后的函数$Q_n(\mu)$。

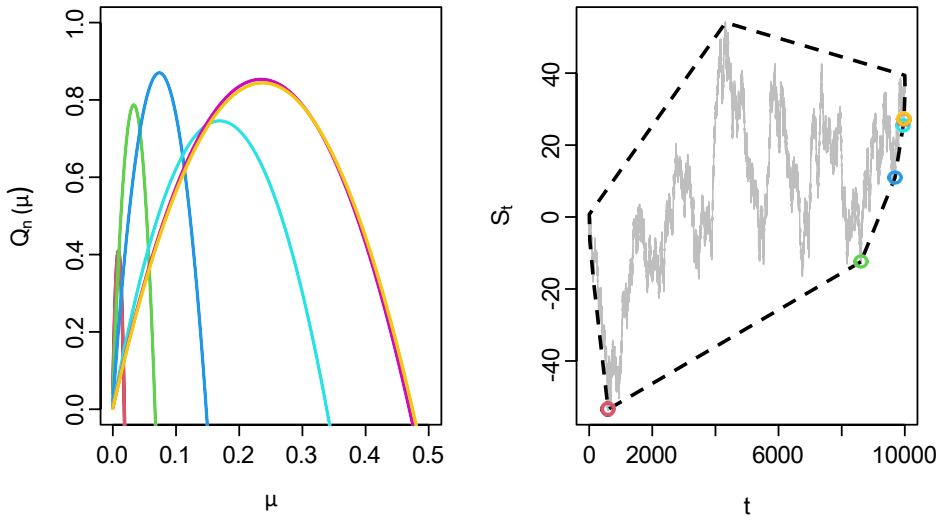


图3：显示由FOCuS^0 保持的四边形的例子，是在时间上引入的。与随机行走S的凸壳的顶点有关$_t = \Sigma^t$ x_i 。 (左)图中的定义Q的四次方$_n$（μ）。(右图）S_t 作为t（灰色）的函数的图，以及凸的点(t, S_t)的壳(虚线)。我们圈出了与左图中每个二次函数的引入时间相关的所有点（t, S_t）（用同样的颜色），这些点对应于凸壳的顶点。所有顶点都位于下凸小体上，因为我们考虑的是平均值的正变化。由于变化后的均值大于变化前的均值0的条件，我们只保留与凸壳左侧的顶点相关的四分法，因此改变变化前的均值只影响下层小数上的哪些顶点对应于被FOCuS保留的四分法。^{0}

算法2：最大 $\{0, Q_{n-1}(\mu) + \mu(x_n - \mu/2)\}$为$\mu > 0$的算法

数据：$Q_k^+(\mu) = Q$ 一个有序的三联体集合$\{q_i = (\tau_i, s_i, l_i) \forall\ i = 1, \ldots, k\}$，$x_n$ 和 S_{n-1}

1 $S_n \leftarrow S_{n-1} + x_n$ ； // 更新累积总和
2 $q_{k+1} \leftarrow (\tau_{k+1} = n, s_{k+1} = S_n, l_{k+1} = \infty)$ ； // 新的二次方程
3 $i \leftarrow k$ ；
4 **而**$2(s_{k+1} - s_i) - (\tau_{k+1} - \tau_i) l_i \le 0$*且*$i \ge 1$**做**
5 $i \leftarrow i - 1$ ；
6 **结束**
7 $l_{k+1} \leftarrow \max\{0, 2(s_{k+1} - s_i)/(\tau_{k+1} - \tau_i)\}$ ； // 更新新的边界
8 **如果***i* $= k$ **，那么**
9 $Q \leftarrow Q \setminus \{q_{i+1}, \ldots, q_k\}$ ； // 修剪旧的二次方程
10 **结束**
11 **返回**$\{Q, q_{k+1}\}$、 S_n

定理*2 对于任何数据*$x_1, \ldots, x_T$ *是*$O(T)$*，其每次迭代的摊销复杂度是*$O(1)$。

证明：在迭代n时，让k_n 是输入的四元数，让c_n 是while语句被评估的次数。让C_1 是步骤1到3和7到11的成本，C_2 是步骤4到6的一组一评估的成本。那么算法2的一次迭代的计算成本是$C_1 + c_n \times C_2$ 。

关键的观察是，$k_{n+1} = k_n - (c_n - 1) + 1$。也就是说，如果我们将步骤4到6重复$c_n$ 次，那么我们将在步骤9中删除$c_n - 1$的二次函数，并在步骤11中增加一个二次函数。此外，$k_1 = 0$，k_{T+1} 是$Q_T(\mu)$的二次函数的数量。因此，总的计算成本是

$$\sum_{n=1}^{T} (C_1 + c\, C_{n2}) = C_1\, T + C_2 \sum_{n=1}^{T} c_n = C_1\, T + C_2 \sum_{n=1}^{T} (2 + k_n - k_{n+1}) \text{ 。}$$

由于伸缩和中的抵消以及$k_1 = 0$的事实，我们有

$$\sum_{n=1}^{T} c_n = 2T - k_{T+1} \le 2T$$

因此该定理成立 □

由于整体计算成本在T中是线性的，因此每次迭代的预期成本必须是常数。该证明给出了算法2中操作的开销形式。在实践中，相对于算法1第2步的成本，即最大化$Q_n(\mu)$，这个成本可以忽略不计。

2.2.2 第2步 最大化

:

为了实现算法1的第2步，我们首先使用琐碎的观察，如果$x_n > 0$，则

$Q_n(\mu) < Q_{n-1}(\mu)$ 对于所有的$\mu < 0$。因此要检查$\max_\mu Q_n(\mu)$ 是否$\ge \lambda$，我们只需要检查这个

同样，如果$x_n < 0$，那么我们只需要检查$\mu < 0$。为了进行检查，我们只需循环检查所有存储的$\mu > 0$或$\mu < 0$的二次方程，并对每个二次方程检查其最大值是否大于λ。对于一个存储有三联体（τ, s, l）的二次方程，这涉及到检查是否

$$(S_n - s)^2 \geq 2\lambda(n - \tau)。 \quad (10)$$

如果我们在时间n上标记一个变化，那么我们也可以输出与最大的二次元对应的τ值，这将是对变化时间的一个估计。因此，计算成本与存储的二次方程的数量成正比，并且可以用下面的结果来约束。

定理3 *让$x_1, \ldots, x_T, \ldots$是过程$X_i = \mu_i + \epsilon_i$的实现，其中ϵ_i是均值为0的同向分布连续随机变量。让数*

在迭代T时，FOCuS^{0}存储的$\mu>0$的二次方程的$\#^{l0}_{1:T}$。那么，如果μ_i是常数

$$E(\#^{l0}_{1:T}) \leq (\log(T) + 1)、$$

而如果μ_i在T之前有一个变化，那么

$$E(\#^{l0}_{1:T}) \leq 2(\log(T/2) + 1)。$$

这方面的证明可以在附录B中找到。关键的想法是使用一个一对一的关系------。
我们需要保留的每个二次元与凸壳的顶点之间的响应关系
的随机行走$S_t = \Sigma^t \quad x_i$，如图3所示。标准结果（Andersen、1955）给出了随机行走的凸壳顶点的界限，其中的增量
是可以交换的。

根据对称性，对于$\mu<0$的情况下存储的四元数也有同样的结果。数据生成机制的条件很弱--因为只要噪声是独立的，噪声的分布可以是任何连续分布。该定理表明，FOCuS^{0}在时间T的预期每次迭代时间和内存复杂度为$O(\log T)$。此外，如果没有变化，每次迭代的预期成本基本上等于检查(10) $\log(n) + 1$次，如果有一个尚未检测到的变化，则为$2(\log(n/2) + 1)$次。一个固定大小的变化在$O(1)$次迭代中被检测出来，因此，对于大的T来说，总体计算时间将被变化点之前的迭代成本所支配。对于100万大小的数据，四元数的约束是小于15。对于FOCuS^0，这个约束可以得到改善。该约束是基于所有
储存的四边形对应于随机的凸小数上的顶点。
走$S_t = \Sigma^t \quad x_i$，但如图3所示，并不是所有凸小体上的顶点都是
对应于被存储的二次方程。事实上，一个简单的对称性论证表明
只有一半的人这样做。这一点在附录C.2中得到了经验上的验证。

FOCuS^{0}算法不是严格意义上的在线算法，因为每次迭代的成本不受限制。但是引入一个在线的近似算法是很简单的。假设我们有一个约束条件，即我们每次迭代最多可以找到P个二次方程的最大值。一个简单的近似方法是引入一个点的网格$\pm m_p$，用于$m_p \in \mathrm{R}^+$，$p=1, \ldots P$。然后有两种自然的方法。一种是，如果我们有P+1个存储的二次元，我们通过删除第一个区间不包含网格点的二次元来修剪到P个二次元。或者，我们可以保留所有的二次方程，但只找到其中的最大值。

其区间包含一个网格点的四次方。后一种方法的优点是，它避免了对函数$Q_n(\mu)$的任何近似，这种近似可能会传播到$t>n$的函数$Q_t(\mu)$的未来值。使用对μ_1值使用相同网格的顺序-Page方法，这两种方法都会占优势。例如，如果$Q_n(\mu)$表示使用第一种方法对$Q_n(\mu)$的近似值，那么对于我们网格中的所有μ_1，我们有$Q_n(\mu_1) = Q_{n,\mu_1}$。因此，对于任何一种方法，由FOCuS评估的四分法的最大值[0]，将等于或大于网格点的sequential-Page统计数据的最大值。与sequential-Page一样，对网格点使用几何比例$\pm m_p$是明智的（见下文）。

2.3 仿真研究

我们研究了FOCuS[0]程序及其近似值（在第2.2节末尾介绍）、方程（5）中的顺序PAGE-CUSUM统计以及方程（2）中的MOSUM程序在已知变化前平均值为0的情况下的平均运行长度和检测延迟。

该研究的结构如下：对于Page递归，我们采用了一个几何网格（正如Chen等人所建议的，2022年）。我们使用10点网格，因为这相当于在十万个观测序列中存储在FOCuS[0]的预期间隔数。为了看到使用更细的网格的潜在好处，我们还使用了20点网格。我们称这两种方法为Page-20p和Page-10p。我们在一组20个窗口尺寸上评估MOSUM，这些尺寸以几何级数增加，并选择让MOSUM在PAGE-20p所使用的网格指定的变化大小上具有最大的能力。我们还测量了FOCuS[0]在用于PAGE-10p的10点网格上的近似性能。我们称这种近似为FOCuS[0] -10p。

对于这些方法中的每一种，我们首先根据没有变化的、长度为200万个观测值的数据来估计运行长度作为阈值的函数。我们在100个不同的复制中对结果进行平均，并在图4中进行总结。对于每一种方法，我们都选择了平均运行长度为1×10^6的阈值，并在不同变化幅度的范围内评估该阈值的检测延迟。为了产生有变化的剖面图，我们在以前的100个空剖面图上叠加了一个变化幅度为1×10^5的片状恒定信号。为了简单起见，我们只显示幅度为正数的变化，然而研究也延伸到了负数的变化。

在图5中，我们报告了成对方法的平均检测延迟与不同变化幅度的对数比。最突出的比较是在FOCuS[0]和PAGE-10p之间。这些方法的相对性能取决于变化的大小与PAGE-10p使用的网格的匹配程度。如果这些完全匹配，那么PAGE-10p的平均检测延迟略小，因为它的阈值较小（见图4）。然而，当我们转向与网格点不同的变化时，PAGE-10p方法相对于FOCuS[0]失去了力量，而后者检测变化的速度会大大加快。一旦我们将网格大小增加到20，我们就会有类似的定性模式，但是现在性能的定量差异很小，就像FOCuS[0]和FOCuS[0] -10p之间的差异。一般来说，相对于基于网格的方法，FOCuS[0]的最大收益出现在小于最小跳变大小，或者大于最大跳变的变化上

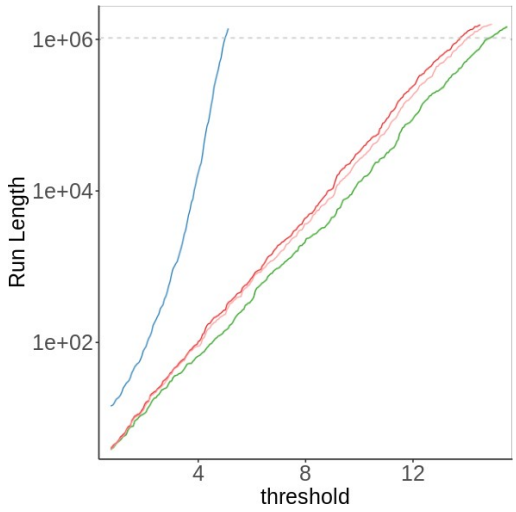


图4：FOCuS^0 和FOCuS^0 -10p（两者相同，以绿色显示）、Page-20p（粉色）、Page-10p（红色）和MOSUM（蓝色）的平均运行长度与阈值的关系。结果是100次模拟的平均数。我们在Y轴上使用对数尺度。

网格的大小。最后，我们看到MOSUM明显是所有方法中性能最差的。

3. 变化前的平均值未知

假设我们在观察一连串的观察结果$x_1, \ldots, x_n$，在任何变化之前分布为$N(\mu_0, \sigma)$，在变化之后分布为$N(\mu_1, \sigma)$，σ已知，$\mu_1 \neq \mu_0$。如前所述，在不丧失一般性的情况下，我们将假设$\sigma = 1$。正如Yu等人（2020）所建议的，我们可以根据似然比统计量来检验变化。

$$LR_n = \max_{\tau \in \{1,\ldots,n-1\},\ \mu_0, \mu_1 \in R} \left(-\sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \sum_{t=\tau+1}^{n} (x_t - \mu_1)^2 \right) - \max_{\mu \in R} \left(-\sum_{t=1}^{n} (x_t - \mu)^2 \right). \quad (11)$$

Yu等人（2020）提出了有限样本的结果，证明了这种测试的统计优化。他们还提出了评估这种测试统计的算法。他们最快的算法避免了任何近似，每次迭代的计算复杂度为$O(n)$，而存储量为$O(n)$，这使得他们的方法在真正的在线环境中不可行。在本文的其余部分，我们将把该算法称为Yu-CUSUM。

为了与我们在变化前均值已知的情况下计算检验统计量的方法相一致，并显示其联系，我们引入以下函数、

$$Q_{\tau,n}(\mu_0, \mu_1) = -\frac{1}{2}\sum_{t=1}^{\tau} (x_t - \mu_0)^2 - \frac{1}{2}\sum_{t=\tau+1}^{n} (x_t - \mu_1)^2$$

这是一个模型的对数可能性，在τ处从平均数μ_0变化到μ_1。如果$\mu_1 = \mu_0$这就减少了数据的对数可能性，其平均值为μ_0。

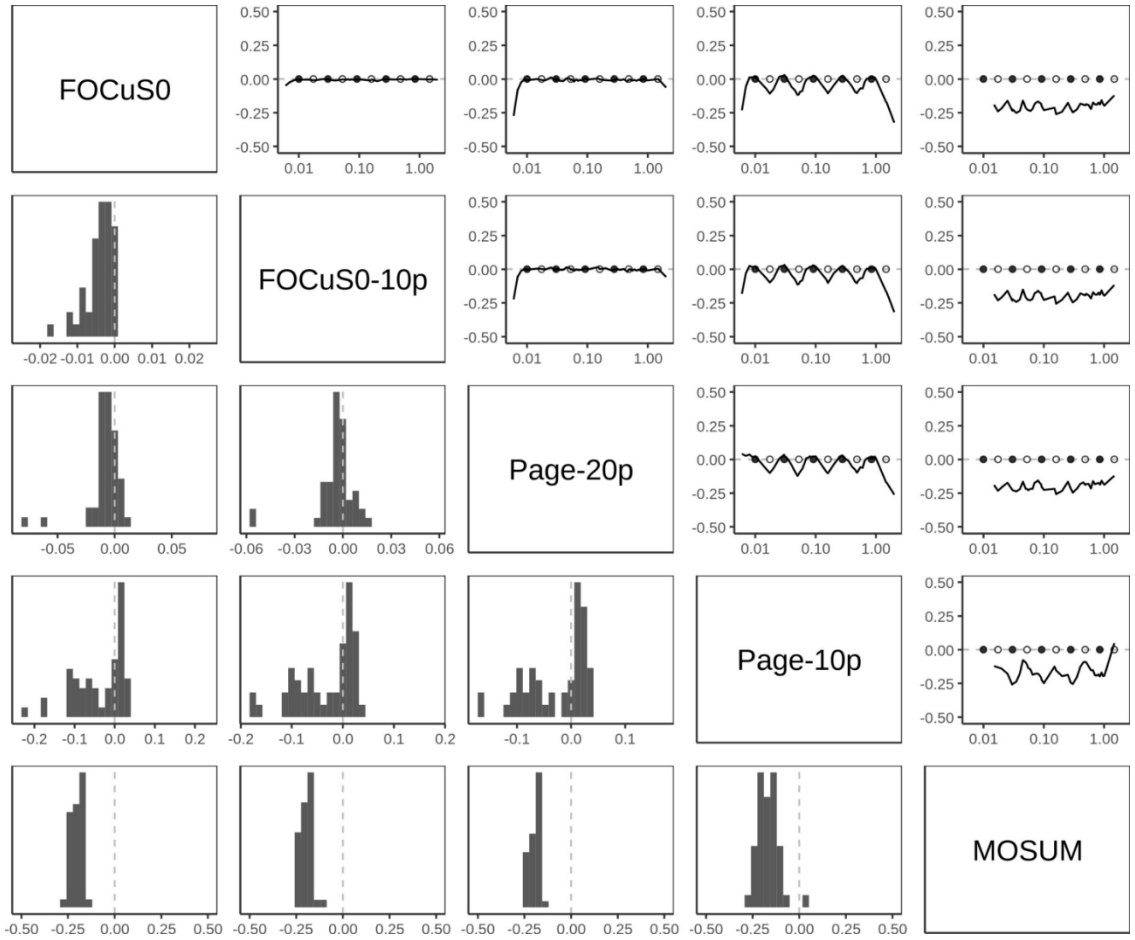


图5：研究中考虑的每种方法的平均检测延迟的对数矩阵。对角线表示第i种测试方法的名称。第(i,j)张图显示了第i种方法和第j种方法的平均检测延迟的对数比，指数高的方法在分子中，指数低的在分母中，所以数值低于零意味着指数低的方法检测延迟低。例如，（1，2）和（2，1）图给出了$FOCuS^0$与$FOCuS^0$ -10p之间的对数比，低于0值意味着$FOCuS^0$的平均检测延迟较小。对角线上的图显示了对数比作为变化幅度的函数；点表示20个点的网格，填充的是与10个点的网格相同。对角线下方的图显示所有对数比的直方图，垂直虚线在0处。

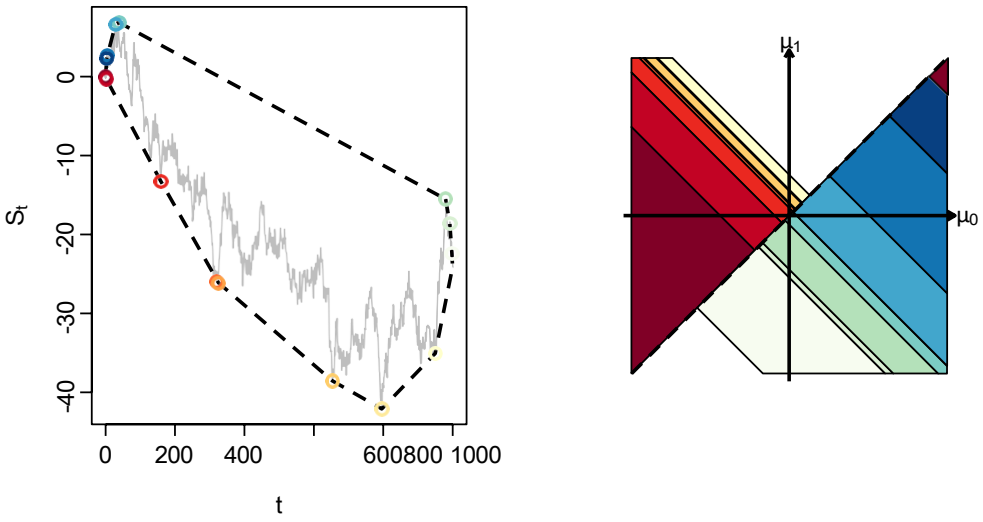


图6：点（t，S_t）的凸面壳，其中$S_t = \sum_{i=1}^{t} x_i$（左图），并绘制了(μ_0, μ_1)空间中不同的二次方对$Q_n(\mu_0, \mu_1)$来说是最优的区域（右图）。
手图）。$Q_n(\mu_0, \mu_1)$定义中的每个二次方都对应于数据随机行走的凸壳上的一个顶点（因为它对应于与该顶点相关的时间的变化），并以与相应顶点相同的颜色着色。

Let $Q_n(\mu_0, \mu_1) = \max_{\tau \in 1,\ldots,n-1} Q_{\tau,n}(\mu_0, \mu_1)$. Then the log likelihood-ratio statistic can be calculated as

$$LR_n = 2 \left[\max_{\mu_0, \mu_1 \in R_1} Q_n(\mu_0, \mu_1) - \max_{\mu \in R_0} Q_n(\mu_0, \mu) \right].$$

因此，如果我们能计算出函数$Q_n(\mu_0, \mu_1)$，就能计算出对数似然比检验统计量。

如果我们固定μ_0，并把$Q_n(\mu_0, \mu_1)$只看作是μ_1的函数，那么，只要有一个常数，这就是上一节中计算的函数。我们可以对以下情况分别计算$\mu_1 > \mu_0$和$\mu_1 < \mu_0$，在每种情况下，该函数都是片状二次函数，其中在时间上引入的二次元是在S的凸壳上$t = \Sigma^t_{i=1} x_i$（见图3）。
For a quadratic introduced at time τ, the quadratic is of the form

$$-\frac{1}{2}\sum_{t=1}^{n} x_t^2 + \tau\mu_0 \left[\frac{S_\tau}{\tau} - \frac{\mu_0}{2} \right] + (n-\tau)\mu_1 \left[\frac{S_n - S_\tau}{n-\tau} - \frac{\mu_1}{2} \right] 。$$

因此，如果我们认为$\mu_1 > \mu_0$，比如说，那么函数$Q_n(\mu_0, \mu_1)$也将是片状二次函数，其中一个二次函数对应于S_t的凸小数上的每个点。这在图6中被形象地显示出来。

因此，计算对数似然比统计量的算法与上一节的算法相似，只是有两个小的区别。

1. 对于区间更新（第1步），在FOCuS^{0}中，对于上行变化，我们可以将注意力限制在$\mu_1 \in [\mu_0, +\infty]$（对于下行变化，则是$(-\infty, \mu_0)$），而在FOCuS中，由于

我们不知道第一段平均数的值，我们需要考虑变化前平均数的所有情况。这意味着，对于FOCuS和向上变化，我们在运行算法2时将第7行改为

$$I_{k+1} \leftarrow 2(s_{k+1} - s_i)/(\tau_{k+1} - \tau_i) 。$$

也就是说，我们不再取这个值和变化前的平均值的最大值。对于向下变化，我们可以应用同样的算法，但是，通过对称，对于符号翻转的数据，即$-x_{1:n}$。

2. 对于FOCuS中的最大化（步骤2）[0]，我们只需要优化最后一段的值，而在FOCuS中，我们还需要对变化前的平均值进行优化。也就是说，在时间n，对于由三重（τ_i，s_i，I_i）定义的二次方，我们计算出

$$\tau_i \left(\frac{S_i}{\tau_i}\right)^2 + (n - \tau)_i \frac{(S_n - S_i)^2}{n - \tau_i} - n \quad 蛇 \quad n^2 .$$

然后，我们找到这些值的最大值，在时间n存储的所有四边形上实现最大化。

通过与定理2相同的论证，我们得出解递归的每次迭代的平均成本是常数。我们在附录B的定理4中推导出与FOCuS的预期候选数相同的约束[0]，表明在时间T上最大化FOCuS的解决方案的预期每次迭代时间和内存复杂度为$O(\log(T))$。

3.1 仿真研究

我们对变化前均值未知的FOCuS和通过训练序列学习的变化前均值已知的$FOCuS^0$进行了比较。这是因为，正如介绍中提到的，当变化前均值未知时，人们可以估计高斯过程的均值，并使用这样的值来运行第2节中介绍的算法。我们特别研究了$FOCuS^0$的性能，因为我们将训练数据的规模从1000个观测值变化到$1\ 10^5$。作为一个基准，我们还比较了具有已知变化前平均值的$FOCuS^0$--它应该显示出最好的性能。我们同时比较了作为阈值函数的平均运行长度和作为变化幅度函数的检测延迟。对于每个实验，我们报告了100次重复的总结，关于检测延迟的结果是针对所选择的阈值，因此每个算法的平均运行长度为1×10^6。在所有情况下，我们模拟了变化前的1×10^5数据点。结果总结在图7中。

FOCuS需要一个较小的阈值来实现不同的$FOCuS^0$实现的相同的平均运行长度，然而，在较大的训练规模上比较这些方法时，差异变得可以忽略不计。关于检测延迟，FOCuS的优势在于它可以利用任何变化之前的数据来提高其对变化前均值的估计，因此当训练数据量较小时，我们看到FOCuS相对于$FOCuS^0$的巨大优势。然而，当训练数据量与变化前的数据量处于同一数量级时，$FOCuS^0$在检测非常小的数据时比FOCuS更有能力

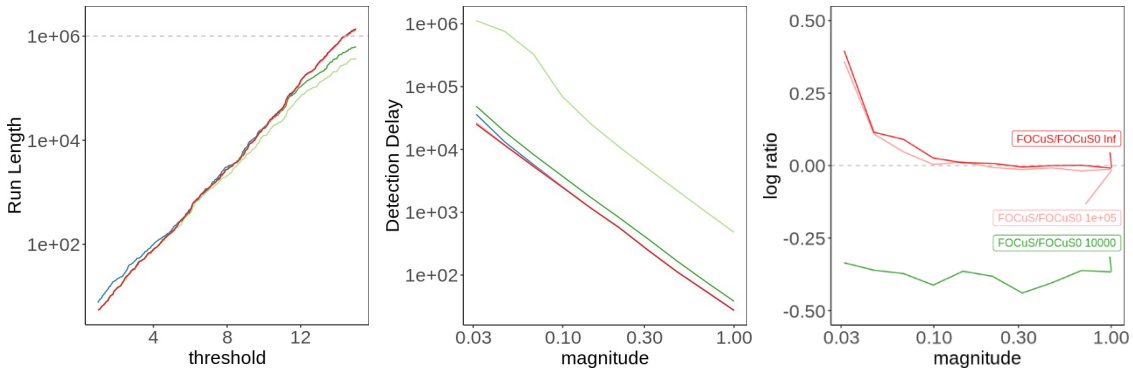


图7：FOCuS的未知变化前和已知变化前的比较：平均运行长度与阈值的对比（左）；平均检测延迟与变化幅度的对比（中）；以及成对方法的平均检测延迟与变化幅度的对数比（右）。对于前两幅图：方法是FOCuS（蓝色）；FOCuS^0，变化前的平均值已知（红色）；FOCuS^0，不同的训练数据大小：1000（浅绿色），1×10^4（深绿色）和1×10^5（粉红色）。

变化，而FOCuS在检测较大变化方面的能力更略强。对于小幅度的变化，FOCuS比FOCuS^0，这一事实是直观的预期。一方面，由FOCuS优化的成本是有界的，使得小的变化无法被检测到，因为成本总是小于预先定义的阈值。另一方面，FOCuS的成本^{0}是无界的，因此假设我们对变化前的平均值有一个合理接近的估计，小的变化可以被识别出来。

4. 存在异常值时的FOCuS

FOCuS的进一步扩展是使用不同的损失函数来处理从高斯对数似然得到的平方误差损失。在第5节应用的激励下，我们将考虑一个稳健的损失函数，即biweight损失，它使我们能够在异常值存在的情况下检测到变化点（见Fearnhead和Rigaill，2019，尽管其他损失函数是片状二次方的，如L_1损失，可以用类似的方法来使用）。偏重损失只是平方误差损失，但以用户选择的最大值K为上限。为了与前面的章节保持一致，我们可以定义一个与数据拟合的相关措施，即减去这个损失、

$$F(x_t, \mu_1) = -\min\left(\frac{(x_t-\mu_1)^2}{2}, K\right)。 \tag{12}$$

然后，如果我们对变化的潜在位置进行最大化，我们的目标是通过监测所产生的对数据的拟合来检测变化。这就导致了以下的函数递归：

$$Q_n(\mu) = \max_{\mu_0}\left(\sum_{t=1}^{n} F(x_t, \mu_0), Q_{n-1}(\mu) + F(x_n, \mu)\right). \tag{13}$$

利用Fearnhead和Rigaill（2019）第3.2节中描述的思路，可以直接对所有的μ有效地实现这个递归。对于这个模型，我们无法恢复一个边界

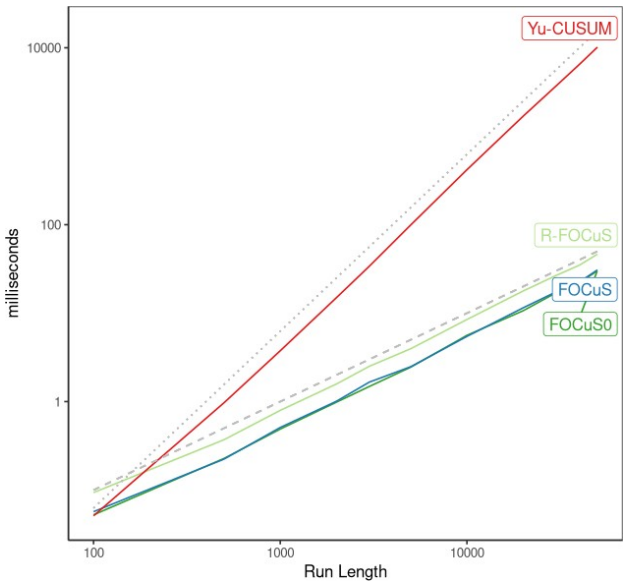


图8：FOCuS^0 、FOCuS、R-FOCuS和Yu-CUSUM的运行时间（以毫秒为单位）与序列长度的关系（两个轴上的对数尺度）。灰线指的是预期的$O(n)$增长（虚线）和$O(n^2)$增长（虚线）。

候选变化点的预期数量。然而，我们根据经验观察到，迭代n的成本为$O(\log(n))$（见图8）。

在图8中，我们给出了FOCuS^0 、FOCuS、(13)中引入的稳健实现(R-FOCuS)和Yu等人(2020)的算法3(表示为Yu-CUSUM)之间的运行时间比较。对长度为100到5×10的多个有限序列的运行时间进行了记录[4] 。为了进行公平的比较，两个实现都是用C++编写的，所有的模拟都是在一台普通的个人计算机上进行的。在比较FOCuS^0 与FOCuS时，我们发现两者的差异不大，都显示了经验性的时间线性增长，而后者稍慢一些。当比较FOCuS和Yu-CUSUM时，我们发现只有在n=100时，FOCuS的运行时间才具有可比性，之后，由于Yu-CUSUM的二次计算复杂性，FOCuS更快。最后，我们注意到R-FOCUS虽然仍然保留了线性计算复杂度，但与更简单的实现相比，它的开销更大。

5. 将FOCuS应用于AWS Cloudwatch的CPU利用率

我们现在通过与Numenta Anomaly Benchmark（Ahmad等人，2017）中的亚马逊CPU利用率数据集上的定制异常检测算法进行比较来评估FOCuS。这些数据集的目的是检测各种亚马逊云观察实例的CPU利用率的异常行为。对于每个数据集的异常行为都是由专家手动标记的，而这些行为则是基础事实。这些数据集如图9所示，显示了一系列的行为。由于点状异常行为很常见，我们将使用R-FOCuS算法。

探测器	精度	召回
R-FOCuS	0.58	0.82
Numenta HTM	0.50	0.76

表1：R-FOCuS和Numenta HTM的精确度和召回率。

在评估算法时，我们将遵循Ahmad等人（2017）的方法。如果检测结果在真实异常值的±0.05-n范围内，则被认为是正确的，其中n是时间序列的长度；并且允许在窗口内进行多次检测。一种方法可以使用每个数据集的前15%，即已知不包括任何异常情况的一部分数据，来设置调整参数。我们使用这个数据来调整偏重损失中的κ和检测阈值，如附录D中所述。

由于一些数据集有多种异常情况需要检测，我们必须对R-FOCuS进行调整，使其在某些异常行为发生变化后不会停止。为了适应R-FOCuS，我们只需在检测被触发后在估计的变化点位置再次启动程序。为了减少假阳性的数量并延长算法的平均运行长度，在每次检测时，我们将阈值膨胀一个对数（τ_s）/对数（$\tau_s - \tau_{s-1}$），其中$\tau_0, \ldots, \tau_k$是估计的变化点位置。这与训练阶段使用的膨胀相同，其形式基于理论，即阈值应与平均运行长度对数成正比（Yu等人，2020）。在分析测试数据时膨胀惩罚是保守的，因为检测到的变化可能是真实的，而不是假阳性，但可以避免数据中的异质性引起的假阳性爆发的问题。

我们将R-FOCuS与numenta HTM进行比较，后者是迄今为止在这些数据上表现最好的算法。Numenta HTM（Ahmad等人，2017）是一种异常检测算法，它采用无监督的神经网络模型来处理时间数据（Cui等人，2016）来进行异常检测。

结果总结在图9和表1中。我们发现，与Numenta HTM相比，R-FOCuS在精确度（检测到的真实异常的比例）和召回率（检测到的真实异常的比例）方面有更好的表现。在个案的基础上，在大多数序列中，两种算法都能正确地标记出异常情况。HTM总体上实现了稍短的检测延迟（除**f**外），但是它产生了更多的假阳性（13个假检测，而R-FOCuS为7个）。在漏检方面，两种算法的表现相似，R-FOCuS在**a**中漏掉了一个由HTM标记的异常，而HTM在**d**中漏掉了一个由R-FOCuS标记的异常。

总的来说，这显示了R-FOCuS在线变化检测的灵活性，特别是考虑到R-FOCuS是一个更简单的方法，它在模型错误描述下运行，而且计算运行时间比Numenta HTM短得多（在3000个观测值上，R-FOCuS大约需要2毫秒，而HTM则需要4分钟）。

6. 讨论

功能性修剪递归的优势之一在于直接操作成本时可以进行大量的扩展。例如，对底层的条件

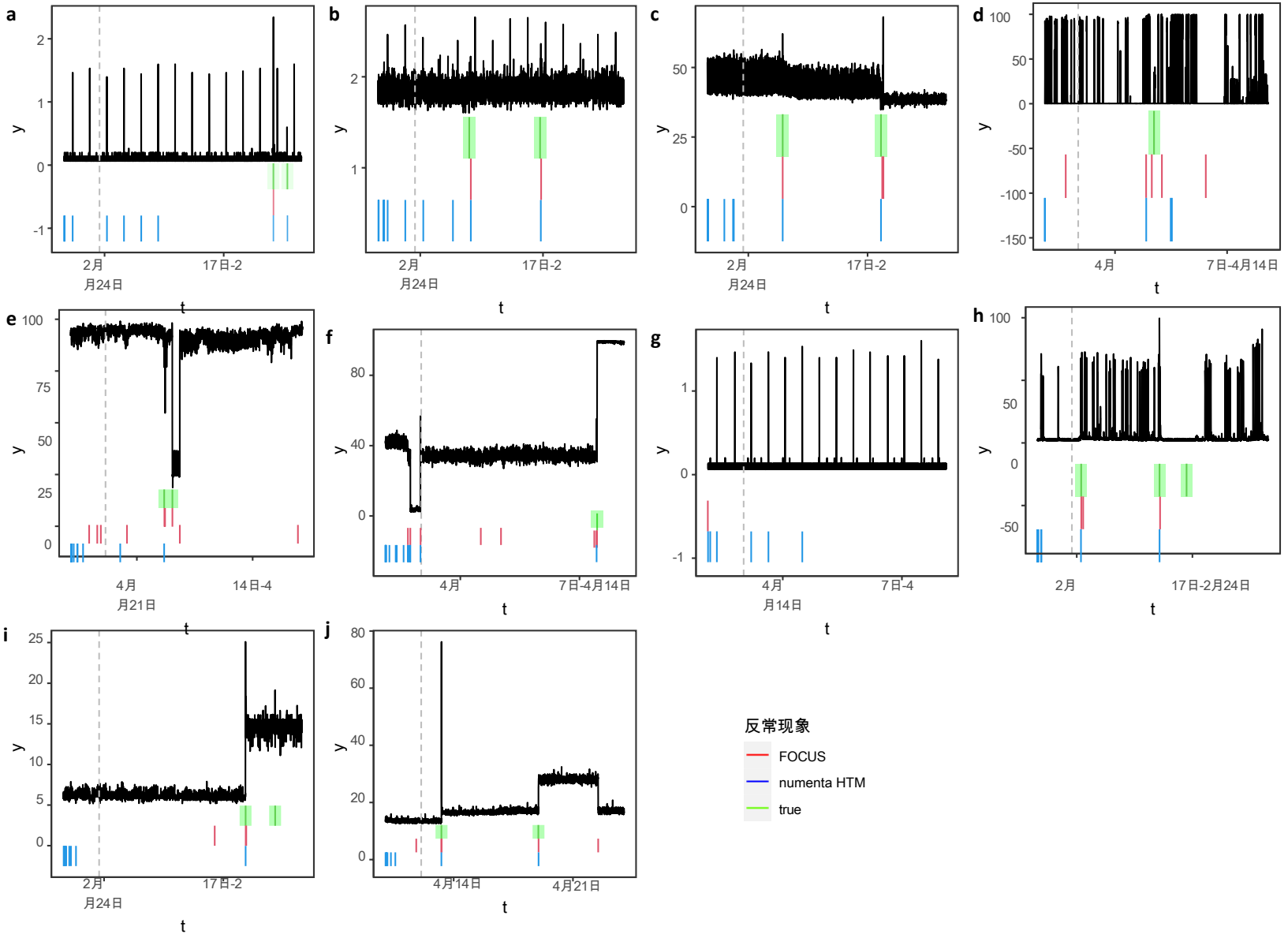


图9：AWS Cloudwatch CPU利用率的8个（**a - j**）不同时间序列。绿色、红色和蓝色段分别对应于R-FOCuS和Numenta HTM的真实和估计异常位置。每个 异常点周围的绿色矩形是异常点窗口（必须检测到异常点才能算作真阳性的区域）。在绿线内的多次检测是允许的，不被认为是假阳性。虚线对应的是试用期，用于训练调整参数：虚线之前的检测不计入最终结果。

		改造前	平均已知	变化前的平均值未知	
稀疏度	量级	恐惧症	荧光粉[0]	恐惧症	荧光粉
0.01	0.25	**318.30**	365.03	**1281.24**	1446.36
0.01	0.5	**95.01**	95.37	336.23	**121.29**
0.01	1	36.37	**25.15**	131.13	**26.63**
0.01	2	12.65	**7.34**	59.83	**7.38**
0.05	0.25	**503.46**	702.48	**2164.01**	2172.04
0.05	0.5	**144.57**	180.37	622.55	**317.72**
0.05	1	43.91	**43.38**	206.59	**48.66**
0.05	2	14.08	**12.40**	86.05	**12.77**
0.1	0.25	**616.11**	803.43	2348.26	**2312.23**
0.1	0.5	**171.11**	233.34	823.61	**525.72**
0.1	1	**50.88**	59.57	258.74	**69.33**
0.1	2	**14.73**	16.99	106.61	**17.64**
1	0.25	**904.74**	1110.48	2903.23	**2506.79**
1	0.5	**245.00**	420.56	1769.32	**1081.09**
1	1	**64.39**	130.30	675.58	**190.29**
1	2	**17.85**	39.74	210.92	**44.09**

表2：多变量序列中不同稀疏程度和量级的变化的平均检测延迟。我们考虑两种情况：变化前均值是已知的和变化前均值是未知的。对于前者，我们使用FOCuS[0]，对于后者，我们使用FOCuS并通过从长度为500的训练数据中估计出的变化前均值实施0CD。稀疏度规定了100个数据流中变化的比例。

对于影响k系列的变化，我们模拟$Z_1, \ldots, Z_k$独立的标准正态变量，并改变第i个系列的平均值，对于$i=1, \ldots, k$，通过$mZ_i/\sqrt{\sum_{j=1}^{k} Z_j^2}$，其中$m$是变化的幅度。所有的变化都发生在时间200。

可以实施推断的平均值对象，以产生制约于特定变化模式的推断（Hocking等人，2020年；Runge等人，2020年），或说明噪声中的波动信号和自相关（Romano等人，2020年），或允许变化之间平均值的已知行为（Jewell等人，2020年）。

递归的主要局限性在于它们依赖于目前只适用于单变量参数函数的函数剪裁思想（不过关于将剪裁扩展到更高维度的想法，请参见Runge, 2020）。然而，在多个参数可能发生变化的情况下，仍有可能应用FOCuS的版本，例如通过分别测试每个参数的变化并合并这些信息。合并FOCuS统计量的一个简单方法是取其最大值或其总和（参见Mei, 2010）。最大值是检测一个或少数序列变化的合适的测试统计量，而总和则适合于检测许多序列的变化（不过，关于跨数据流合并统计量的其他方法，见Enikeeva和Harchaoui，2019；Fisch等人，2021；Tickle等人，2021）。

为了研究这种方法的潜力，我们进行了一项模拟研究，比较了一种同时使用跨数据流的最大和FOCuS统计量的方法，并将其与Chen等人（2022）的0CD方法进行比较。我们使用了与Chen等人（2022）类似的模拟设置，数据来自100个数据流。正如Chen等人（2022）所建议的那样，我们使用来自无变化模型的模拟来选择最大统计量和总统计量以及三种ocd统计量的阈值，因此FOCuS方法和ocd的平均运行长度都是5000个。然后，我们根据不同大小的变化和影响不同数量的数据流的平均检测延迟来比较方法。结果显示在表2中。

ocd方法计算每个流的变化值的网格的sequential-Page统计。然后，它用最大值来组合这些数据。它还使用了两个基于平均变化的对数似然比检验统计量的统计量，这些时间对应于单个序列具有大的顺序-页统计量的变化时间。其中一个就像我们的总和统计，但它假定所有系列的变化时间是相同的，不像我们的实现，它对可能对应于不同时间变化的测试统计进行总和。另一个用于跨数据流组合的统计是为了捕捉稀疏的变化，但在多个数据流发生变化时。

结果显示，如果变化前的平均值是已知的，那么0CD往往比我们简单的合并FOCuS^0 统计的方式表现得更好。虽然我们的方法能够更快检测到稀疏和大的变化，因为0CD方法是用一个相对较小的最大变化规模的网格来分析每个数据流。但是，如果变化前的均值是未知的，并且必须从长度为500的训练数据中估计出来，我们看到，通过使用FOCuS来考虑这个问题会导致性能的大幅提高。性能的提高显然取决于有多少训练数据。

FOCuS不是一种在线算法，因为每次迭代要最大化的四元数的数量可能会波动，而且是无限制的。我们提出了两种方法来实现在线版本，即在每次迭代中最多只对p个四元数进行最大化，其依据是对变化后平均值的几何空间网格进行最大化。这样的方法将一致地优于使用相同网格的sequential-Page。然而，还有其他可能更好的方法来选择要最大化的四元数。首先，我们可以在四元数中循环，因此在时间t+1时开始最大化原来的四元数。

另外，你可以使用观察值来帮助选择要最大化的四元数。例如，如果收到的观察值$x_t > 0$，那么这将增加$Q_t(\mu)$，只有在$\mu \in (0, 2x_t)$时才会增加，而在$\mu = x_t$时最多。因此，考虑只对大于$2x_t$的值最优化的四元数没有意义，可以优先考虑对最接近于x_t的μ区域最优化的四元数。

根据定理4，一个进一步发展的可能领域是研究在备选方案下四分法数量的明确分布，以防统计量没有达到阈值。在这种情况下，我们预计变化的数量会以更快的速度增加，而不是在无效状态下：一个额外的测试可以放在预期的四分法数量上，作为宣布变化的一个故障安全机制。

根据定理4的证明，可以很容易地看出，离线pDPA算法（Rigaill, 2015）为一个变化所存储的候选变化点集包括在FOCuS所存储的变化点集中。这为pDPA在一次变化中的预期复杂度提供了一个约束。未来的工作可以考虑扩展定理4的证明，以获得多于一个变化的pDPA的预期复杂度或其他功能修剪算法，如FPOP（Maidstone等人，2017）或GFPOP（Hocking等人，2020）。

参考文献

Joshua Simon Abramson.*随机游走和Lˊevy过程的一些小数和大数*。博士论文，加州大学伯克利分校，2012年。

Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha.流媒体数据的无监督实时异常检测。*Neurocomputing*, 262:134-147, 2017.

Joham Alvarez-Montoya, Alejandro Carvajal-Gómez, and Juliˊan Sierra-Pérez.使用光纤传感器和应变场模式识别的无人机复合机翼的飞行中和无线损伤检测。*机械系统和信号处理*，136：106526，2020。

Erik Sparre Andersen.On the fluctuations of sums of random variables ii.*Mathematica Scandinavica*, 2:195-223, 1955.

Michele Basseville和Igor V Nikiforov.*突变的检测：理论与应用*，第104卷。Prentice Hall Englewood Cliffs, 1993.

Michele Basseville, Albert Benveniste, Maurice Goursat, and Laurent Meve.航空结构的飞行中维度监测.*IEEE 控制系统杂志*, 27(5)：27-42, 2007.

Yudong Chen, Tengyao Wang, and Richard J Samworth.高维、多尺度的在线变化点检测.*皇家统计学会杂志（B系列）*，84：234-266，2022。

陈占寿和田正.线性模型中变化点监测的修正程序.*数学与计算机仿真*, 81(1):62-75, 2010.

Chia-Shang J Chu, Kurt Hornik, and Chung-Ming Kaun.参数不变性的MOSUM测试.*Biometrika*, 82(3):603-617, 1995.

Gari D Clifford, Ikaro Silva, Benjamin Moody, Qiao Li, Danesh Kella, Abdullah Shahin, Tristan Kooistra, Diane Perry, and Roger G Mark.The PhysioNet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the ICU.In *2015 Computing in Cardiology Conference (CinC)*, pages 273-276.IEEE，2015年。

Yuwei Cui, Subutai Ahmad, and Jeff Hawkins.用无监督的神经网络模型进行连续的在线序列学习。*Neural Computation*, 28(11):2474-2504, 2016.

Peter Eiauer和Peter Hackl.使用MOSUMS进行质量控制。*Technometrics*, 20 (4):431-436, 1978.

Farida Enikeeva and Zaid Harchaoui.稀疏替代物下的高维变化点检测.*The Annals of Statistics*, 47(4):2051-2079, 2019.

Paul Fearnhead和Guillem Rigaill.异常值存在下的变化点检测.*Journal of the American Statistical Association*, 114(525):169-183, 2019.issn 0162- 1459.

Alexander TM Fisch, Idris A Eckley, and Paul Fearnhead.子集多变量集体和点异常检测.*计算和图形统计学杂志》*，第1-12页，2021年。

PA Fridman.一种检测无线电瞬变的方法。*皇家天文学会月刊》*，409（2）：808-820，2010。

Fabio Fuschino, Riccardo Campana, Claudio Labanti, Yuri Evangelista, Marco Feroci, L Burderi, Fabrizio Fiore, Filippo Ambrosino, G Baldazzi, and P Bellutti.HERMES：纳米卫星上的超宽频X和伽马射线瞬态监测器。*Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 936:199-203, 2019.

Josua Gösmann, Tobias Kley, and Holger Dette.a new approach for open-end sequential change point monitoring. *arXiv preprint arXiv:1906.03225*, 2019.

Toby Hocking, Guillem Rigaill, Paul Fearnhead, and Guillaume Bourque.用于基因组数据中峰值检测的受限动态编程和监督惩罚学习算法。*机器学习研究杂志*, 21:1-40, 2020.

Daniel R Jeske, Nathaniel T Stevens, Alexander G Tartakovsky, and James D Wilson.网络监控的统计方法.*Applied Stochastic Models in Business and Industry*, 34(4):425-445, 2018.

Sean W Jewell, Toby Dylan Hocking, Paul Fearnhead, and Daniela M Witten.钙成像数据的快速非凸去卷积。*Biostatistics*, 21(4):709-726, 2020. doi: 10.1093/biostatistics/kxy083.

Claudia Kirch and Joseph Tadjuidje Kamgaing.论估计函数在监测变化点的时间序列中的使用。*Journal of Statistical Planning and Inference*, 161:25-49, 2015.

Claudia Kirch和Silke Weber.基于估计函数的修正的顺序变化点程序.*Electronic Journal of Statistics*, 12(1):1579-1613, 2018.

G. Lorden.对分布变化做出反应的程序.*Ann.Math.Statist.*, 42(6)：1897-1908, 1971.

Robert Maidstone, Toby Hocking, Guillem Rigaill, and Paul Fearnhead.关于大数据的最佳多恳求变化点算法。*Statistics and Computing*, 27(2):519-533, Mar 2017.

Yajun Mei.用于监测大量数据流的高效可扩展方案.
Biometrika, 97(2):419-433, 2010.

Avraham A Melkman.简单折线的凸壳的在线构建.*Infor- mation Processing Letters*, 25(1):11-12, 1987.

Ewan S Page.连续检查计划.*Biometrika*, 41(1/2):100-115, 1954.

Ewan S Page.对发生在未知点的参数变化的检验。*Biometrika*, 42(3/4):523-527, 1955.

Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao.使用源IP地址监控主动检测分布式拒绝服务攻击。在*国际网络研究会议上*，第771-782页。Springer, 2004.

Theodor D Popescu and Dorel Aiordˇachioaie.在R´enyi熵上操作变化检测的新程序，在地震信号处理中的应用。*Circuits, Systems, and Signal Processing*, 36(9):3778-3798, 2017.

AD Pouliezos 和 George S Stavrakakis.*工业过程的实时故障监测*，第12卷。Springer Science & Business Media, 2013.

Marion R Reynolds.累积和控制图中平均运行长度的近似值。*Technometrics*, 17(1):65-71, 1975.

Guillem Rigaill.一种修剪的动态编程算法来恢复具有1至kmax变化点的最佳区段。*Journal de la Societe Francaise de Statistique*, 156 (4):180-205, 2015.

Gaetano Romano, Guillem Rigaill, Vincent Runge, and Paul Fearnhead.在存在局部波动和自相关噪声的情况下检测突然的变化。*美国统计学会杂志》*，2020年。

Vincent Runge.在欧几里得空间中，球的有限交集被球的有限联合所覆盖吗？*优化理论与应用杂志*, 187(2):431-447, 2020.

Vincent Runge, Toby Dylan Hocking, Gaetano Romano, Fatemeh Afghah, Paul Fearnhead, and Guillem Rigaill. gfpop: an R package for univariate graph-constrained change point detection. *arXiv preprint arXiv:2002.03646*, 2020.

Alexander G Tartakovsky, Aleksey S Polunchenko, and Grigory Sokolov.通过变化点检测方法进行高效的计算机网络异常检测.*IEEE Journal of Selected Topics in Signal Processing*, 7(1):4-11, 2012.

Sam O Tickle, IA Eckley, and Paul Fearnhead.一种计算效率高的高维多变化点程序，适用于全球恐怖主义证据。*皇家统计学会杂志：系列A（社会中的统计）*，2021年。

Venugopal V Veeravalli 和 Taposh Banerjee.最快的变化检测。在*学术出版社的信号处理图书馆*，第3卷，第209-255页。Elsevier, 2014.

Liyan Xie, Yao Xie, and George V Moustakides.地震震颤的异步多传感器变化点检测。In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 787-791.IEEE, 2019.

Yi Yu, Oscar Hernan Madrid Padilla, Daren Wang, and Alessandro Rinaldo.a note on online change point detection. *arXiv preprint arXiv:2006.03283*, 2020.

补充材料

附录A.命题1的证明

可以直接证明，例如通过归纳法，解决$Q_n(\mu)$的递归问题(6)可以写成以下形式

$$Q_n(\mu) = \max_{s=0,\ldots,n} \left(\sum_{t=s+1}^{n} \mu x_t - \frac{\mu^2}{2} \right),$$

其中，我们把从$n+1$到n的和视为空，因此等于0。

$$\max_{\mu} Q_n(\mu) = \max_{s=0,\ldots,n} \max_{\mu} \left(\sum_{t=s+1}^{n} \mu x_t - \frac{\mu^2}{2} \right)$$

$$= \max_{s=0,\ldots,n-1} \frac{1}{2} \left(\sum_{t=s+1}^{n} x_t \right)^2 = \max_{w=1,\ldots,n} \frac{1}{2w} \left(\sum_{j=n-w+1}^{n} x_j \right)^2,$$

第二行使用了这样一个事实：当μ是x的样本平均数时，μ上的最大值是$\tau_{:n}$。对于第二步，我们使用了这样一个事实：最大值永远不是空和，因此我们可以从最大化中放弃$s=n$的情况。最后表达式中的条款只是$(1/2)M_w(n)^2$，符合要求。$P(n)$的结果直接来自于$P(n) = \max_w |M_w(n)|$。□

附录B.关于FOCuS存储的预期变化数量

B.1 FOCuS实现的变体

我们研究了FOCuS在每次迭代中存储的候选变化点$\tau \in \{1, \ldots, n\}$的数量。我们报告了本文主体部分所介绍的可能的FOCuS优化方法：

- FOCuS^0，它解决了已知的变化前平均值μ_0（通常为0）和未知的变化后平均值μ_1的问题。

$$Q_n^0 = \max_{\substack{\tau \in \{1,\ldots,n\} \\ \mu_0 = 0, \mu_1 \in \mathbb{R}}} \left(\sum_{t=1}^{\tau} (x_t - \mu_0) - \sum_{t=\tau+1}^{n} (x_t - \mu_1) \right)^2 \quad (14)$$

  这个问题是通过算法2解决的，这个算法类似于Melkman的al- gorithm（Melkman，1987）。

- FOCuS，它解决了变化前和变化后的未知问题。指的是：

$$Q_n = \max_{\tau \in \{1,\ldots,n\}} \left(\sum_{t=1}^{\tau} (x_t - \mu_0) - \sum_{t=\tau+1}^{n} (x_t - \mu_1) \right)^2 \quad (15)$$

$t=1$

$$= \tau + 1$$

B.2 假设和定义

在本节的其余部分，我们让$x_1, \ldots x_n$是我们的有序观察序列。为了表示这样一个序列的子集，我们将写$x_{i:j} = x_i, \ldots, x_j$，$i < j$。我们用$\tau^*$来表示一个真正的变化点。

我们假设：

$$x_i = \mu_i + \varepsilon_i \text{、} \tag{16}$$

其中，ε_i是具有连续分布的i.i.d，μ_i是1或2块的分片恒定信号。

我们把在τ处发生变化的分割$x_{i:j}$的成本定义为：

$$q_{i:j,\tau}(\mu_0, \mu_1) = \sum_{t=i}^{\tau} (x_t - \mu)_0^2 + \sum_{t=\tau+1}^{j} (x_t - \mu)_1^2 \text{。}$$

变化前和变化后的含义是μ_0和μ_1。作为惯例，对于$j = \tau$，我们采取：

$q_{i:j,j}(\mu_0, \mu_1) = \Sigma^{j}_{t=i} (x_t - \mu)_0^2$。

候选变化点的集合　　我们称$^{I^0}$为$_{ij}$储存的候选变化点的集合

在$\mu_1 > 0$的情况下，由FOCuS^0；在$\mu_1 > \mu_0$的情况下，$\mathbf{I}_{i:j}$是由FOCuS存储的集合。根据定义，这些将是：

$$_{I^0_{ij}} = \tau \mid \exists\, \mu_1 > 0, \exists \tau' = \tau, \quad q_{i:j,\tau}(0, \mu_1) < q_{i:j,\tau'}(0, \mu_1)\} \text{、}$$

$$\mathbf{I}_{i:j} = \{\tau \mid \exists\, \mu_1 > \mu_0, \exists \tau' = \tau, \quad q_{i:j,\tau}(\mu_0, \mu_1) < q_{i:j,\tau'}(\mu_0, \mu_1)\} \text{、}$$

根据定义，I^0是在$_{ij}$ 我们的目标将是控制$\mathbf{I}_{i:j}$的大小。

由FOCuS存储的候选变化点。

B.3 主要结果

在从(16)中实现的假设上，我们可以得到以下关于FOCuS^{0}和FOCuS存储的变化点数量的约束。

定理4 *对于所有*$n \geq 1$

$$e(\#^{i0})_{:n} \leq e(\#\mathbf{i}_{1:n}) \text{。}$$

如果μ_i*，相对于*i*来说是常数：*

$$E(\#\mathbf{I}_{1:n}) = 1 + \sum_{1}^{n-1} 1/(t+1) \leq (1 + \log(n))$$

如果μ_i*有一个变化点，我们有*

$$E(\#\mathbf{I}_{1:n}) \leq 2(1 + \log(n/2)) \text{。}$$

根据对称性，同样的结果也适用于FOCuS存储的四元数^0，用于$\mu_1 < 0$，FOCuS为$\mu_1 < \mu_0$。

证明**概述**本定理的证明依赖于三个公理的组合，在此总结一下：

1. 推理5：对于$i \le j < k$，我们有$\mathbf{I}_{i:k} \subseteq \mathbf{I}_{i:j} \cup \mathbf{I}_{j+1:k}$；
2. 定理6：$\mathbf{I}_{i:j}$ 是序列的最大凸小数的极端点 $S_{i:j}$，其中 $S_t = \sum_{k=i}^{t} x_k$；
3. 定理7：控制随机漫步的极端点的数量（源自Andersen, 1955; Abramson, 2012）。

这三个公理在附录B.4中有详细介绍和证明。

证明：根据定义，$\mathbf{I}_{i:j}$ 包括I_0，我们得到第一个不等式。对于这种情况 j 其中，μ_i 相对于i是常数，我们适用Lemma 7。对于μ_i 有a的情况 单一变化点，利用Lemma 5我们可以得到

$$\mathbf{I}_{1:n} \subseteq \mathbf{I}_{1:\tau} \cup \mathbf{I}_{\tau+1:n}。$$

然后我们在$\mathbf{I}_{1:\tau^*}$和$\mathbf{I}_{\tau^*+1:n}$上应用Lemma 7。对于$\tau^* = n/2$，可以得到最坏的情况。□

对这一约束的实证评估可以在附录C.2中找到。

B.4 包容和凸壳法则

一个有用的特性 对于任何$i \le \tau < \tau' \le j$, μ_0 和 μ_1，我们有：

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) = (\mu_0 - \mu_1) \left(\sum_{t=\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu_1)/2 \right), \quad (17)$$

这一特性简化了以下公设的证明。

定理5 *对于$i \le j \le k$*

$$\mathbf{I}_{i:k} \subseteq \mathbf{I}_{i:j} \cup \mathbf{I}_{j+1:k} \quad (18)$$

证明：考虑在$(i+1:j) \cap \mathbf{I}_{i:k}$ 中的任何τ，根据定义

$$\exists\, \mu_0 < \mu_1, \forall\, \tau' \neq \tau,\ \tau' \text{在} (i+1:k),\quad q_{i:k,\tau}(\mu_0, \mu_1) < q_{i:k,\tau'}(\mu_0, \mu_1),$$

然后使用方程（17），我们得到$q_{i:j,\tau}(\mu_0, \mu_1) < q_{i:j,\tau'}(\mu_0, \mu_1)$，因此$\tau$也在$\mathbf{I}_{i:j}$。对于$(j+1:k) \cap \mathbf{I}_{i:k}$ 中的任何τ，我们进行类似的操作。□

下一个定理将集合$\mathbf{I}_{i:j}$ 与$S_{i:j}$ 的下凸壳联系起来。

定理6 $\mathbf{I}_{i:j}$ 中的τ的集合是序列$S_{i:j}$ 的最大凸小数的极端点。

证明：假设τ' 在$\mathbf{I}_{i:j}$ 中。那么，根据$\mathbf{I}_{i:j}$ 的定义，存在一个（μ_0, μ_1），其特征是$\mu_1 > \mu_0$ 这样，对于所有的$\tau < \tau'$、

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau'}(\mu_0, \mu_1) > 0$$

并且对于所有的

$\tau'' > \tau'$ $q_{i:j,\tau}{}'(\mu_0, \mu_1) - q_{i:j,\tau}{}'(\mu_0, \mu_1) < 0$。

使用方程（17），$\tau < \tau'$ 的方程可以得到

$$(\mu_0 - \mu)_1 \; 2\sum_{\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu_1) > 0$$

$$\Rightarrow \quad 2\sum_{\tau+1}^{\tau'} x_t - (\tau' - \tau)(\mu_0 + \mu)_1 < 0$$

$$\Rightarrow \bar{x}_{\tau+1:\tau}{}' < \frac{1}{2}(\mu_0 + \mu_1)、$$

其中$\bar{x}_{\tau+1:\tau}{}' = \frac{\sum_{t=\tau+1}^{\tau'} x_t}{\tau' - \tau}$。第二个不等式由$\mu - \mu < 0$得出。

对$\tau'' > \tau'$ 进行类似的论证，可以得出：

$$\bar{x}_{\tau'+1:\tau'} > \frac{1}{2}(\mu_0 + \mu_1)。$$

对于这样一个(μ_0, μ_1)的存在，使这些不等式在所有$\tau < \tau' < \tau''$ 时都成立，我们需要：

$$\bar{x}_{\tau+1:\tau'} < \bar{x}_{\tau'+1:\tau'}、$$

对于所有的 $\tau < \tau'$

$< \tau''$。

用数据的累积和重写样本平均值，这相当于

对于所有τ和τ''，满足$\tau < \tau' < \tau''$：

$$\frac{S_{\tau'} - S_\tau}{\tau' - \tau} < \frac{S_{\tau''} - S_\tau}{\tau'' - \tau'},$$

因此，$(S_{\tau'}, \tau')$是序列$S_{i:j}$ 的最大凸次方的一部分。

我们现在证明反过来。假设τ' 是$S_{i:j}$ 的凸小数的一部分。那么对于所有满足$\tau < \tau' < \tau''$ 的τ 和 τ''：

$$\frac{S_{\tau'} - S_\tau}{\tau' - \tau} < \frac{S_{\tau''} - S_\tau}{\tau'' - \tau'},$$

因此存在一个m，使得

$$\max_{\tau} \bar{x}_{\tau+1:\tau}{}' < m < \min_{\tau'} \bar{x}_{\tau'+1:\tau'}。$$

现在选取μ_0 和μ_1，使$\frac{1}{2}(\mu_0 + \mu_1) = m$ 和$\mu_0 - \mu_1 = -1$，我们得到使用方程（17）我们得到对于所有$\tau < \tau$、

$$q_{i:j,\tau}(\mu_0, \mu_1) - q_{i:j,\tau}{}'(\mu_0, \mu_1) > 0$$

并且对于所有的

$\tau'' > \tau'$ $q_{i:j,\tau}{}'(\mu_0, \mu_1) - q_{i:j,\tau}{}'(\mu_0, \mu_1) < 0$。

□

下一个定理是基于Andersen（1955）的。

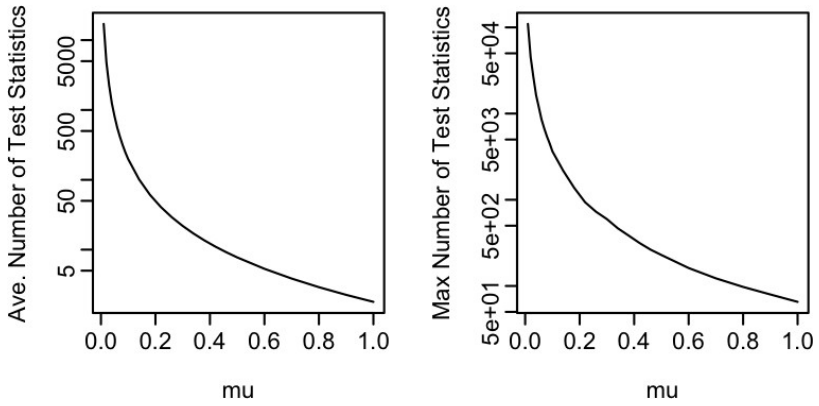


图10：Lorden（1971）对长度为10^6的数据所存储/评估的预期和最大的测试统计量与最小变化大小（mu）的关系图。*y*轴为对数尺度。

定理7 *假设x_t遵循i.i.d连续分布，那么*$E(\#\mathrm{I}_{i:j}) = \sum_{t=1}^{j-i-1} 1/(t+1) + 1 \leq \log(n) + 1$。

证明：我们使用Lemma 6，然后应用Andersen（1955）（定义在第195和196页，然后是第217页的结果）。Andersen（1955）没有包括随机行走的最后一点的顶点，所以我们需要为*j*处的点添加一个1。谐波数列的标准结果表明，$\sum_{t=1}^{j-i-1} 1/(t+1) \leq \log(j-i)$。证明如下：$(j-i) \leq n$。

□

附录C.其他实证结果

C.1 与Lorden (1971)比较

我们实施了Lorden(1971)的算法来检测一个积极的变化。这个算法需要指定一个最小的变化大小，比如说μ^*。然后，它对μ^*，运行sequential-Page程序。该程序将在某些时候重置，即sequential-Page统计量将为0。在这些时候，为任何变化$\mu > \mu^*$计算的sequential-Page统计量将是0。因此，为了检测大于μ^* Lorden（1971）建议计算自上次重置后任何时间开始的变化的似然比测试统计量。也就是说，如果在时间*t*，最后一次重置是在时间τ，那么它将计算在时间τ、$\tau+1$、.....、*t-1*中每一个时间的变化的似然比统计量。，*t* - 1.最后的测试统计量是这些统计量的最大值。这涉及到计算$t-\tau$统计量，在成本上相当于在FOCuS中最大化$t-\tau$二次方。

图10显示了Lorden（1971）算法存储的测试统计量的平均数和最大数与最小跳跃大小的关系。这些都是根据经验对长度为$n=10$的数据进行计算的[6]，其结果是50次复制的平均值。我们还在图11中显示了不同最小变化规模下存储的统计量与*t*的函数关系，以及由FOCuS存储的四分法的预期数量[0]

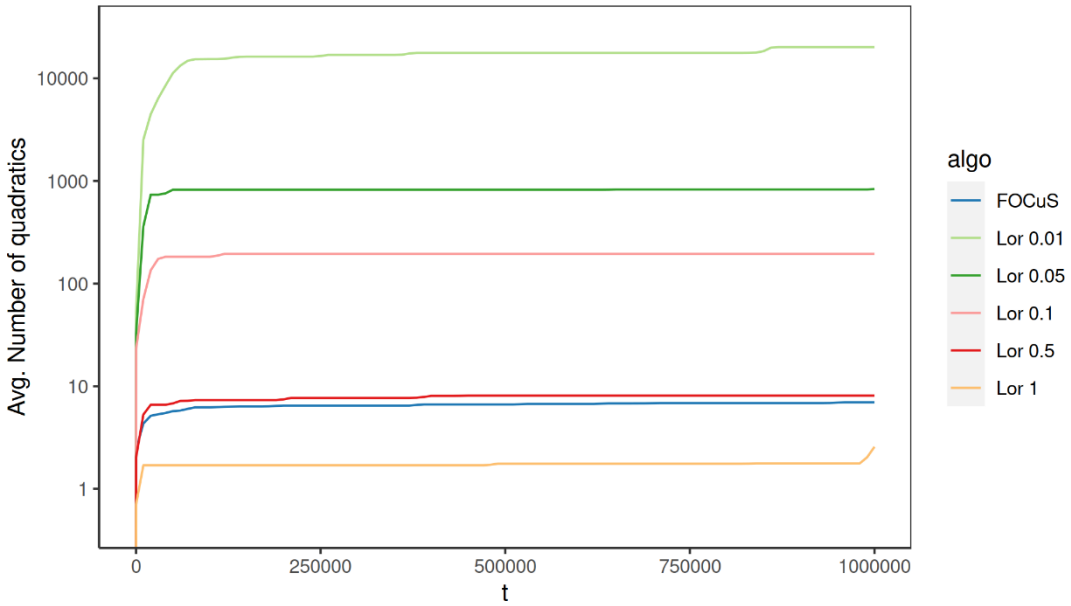


图11：Lorden(1971)存储/评估的测试统计量的预期数量和FOCuS存储的四分法的预期数量[0]，与长度为10的数据的时间关系图[6]。结果是通过对每种算法的50次复制的数据进行单调的平均函数拟合得到的。*y*轴为对数尺度。

对于长度为10的数据[6]，FOCuS[0]，平均将存储约8个四分法。因此，为了提高计算效率，Lorden(1971)需要的最小变化量约为0.5或更高。此外，Lorden(1971)存储的测试统计量的变化要比FOCuS高得多（例如，对于0.8的最小变化量，它可能需要评估超过100个测试统计量），如果每次迭代的计算资源有限制，就会产生问题。

C.2 经验性的约束评价

为了说明定理4的约束，我们模拟了不同长度n的信号（从$n = 2^{10}$到$n = 2^{22}$），没有变化（100次重复）和有一次变化（100次重复）。然后我们记录由FOCuS（左）和FOCuS[0]（右）存储的候选信号的数量。在有变化的情况下，其位置在1和n-1之间被均匀地采样。同样地，变化的幅度也是在[0, 4]中均匀地随机取样。结果总结在图12中。

定理4中的约束适用于FOCuS[0]和FOCuS的向上变化，而对于向下变化，类似的约束通过对称性而成立。这些都给出了一个关于存储的四元数的总约束，如果数据在没有变化的情况下被模拟，那么这个约束是$2(\log(n)+1)$；如果数据被模拟为单一变化，那么这个约束是$4(\log(n/2)+1)$。我们将观察到的四元组存储数量与$4(\log(n) + 1)$作对比，同时绘制$2(\log(n) + 1)$和$\log(n) + 1$的线条。对于FOCuS，我们看到，在数据模拟没有变化的情况下，$2(\log(n)+1)$的约束是很严格的。然而，存储的二次方程的数量非常

对于有变化的模拟数据来说也是如此--这说明这种情况下的4(log(*n*/2) + 1)约束是保守的。对于FOCuS^0，根据经验，在数据模拟无变化的情况下，存储的四边形数量是FOCuS的一半，这与定理4之后的讨论一致。该定理中的约束是基于对由数据的累积和定义的随机行走的凸小数上的顶点数量的约束。然而，对于FOCuS^0，存储的四边形数量等于顶点的一个子集，即被具有正斜率的小数边所跟随的顶点。根据对称性，这将是位于凸次元最小值的顶点，再加上预期的其余顶点的一半。

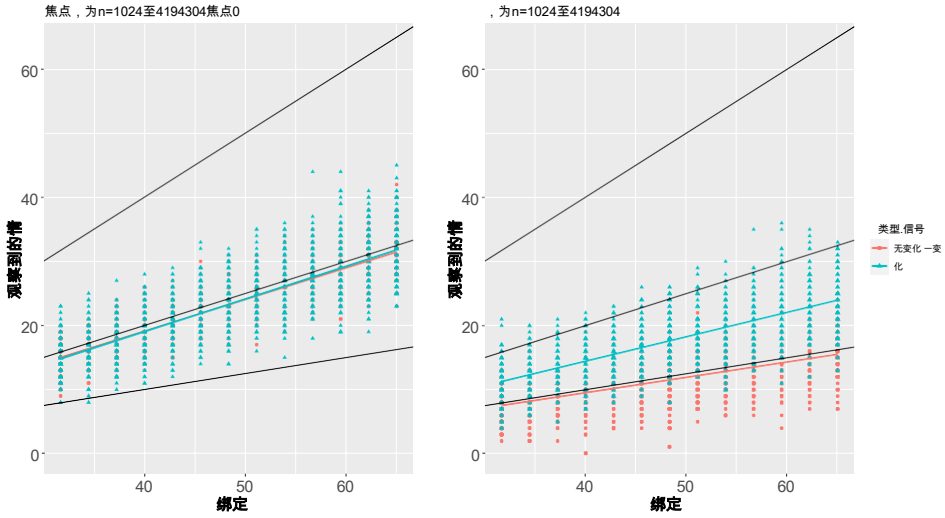


图12：对于没有变化或有一个变化的信号，由FOCuS（左）和FOCuS^0（右）存储的观察到的四边形的数量与4（log（*n*）+1）的界限。三条黑线代表函数$y = x$、$y = 0.5x$和$y = 0.25x$，分别代表4(log(*n*) + 1)、2(log(*n*) + 1)和log(*n*) + 1。红线和蓝线分别是无变化和单一变化情况下的拟合回归线。

附录D.初始参数的估计

我们为以半监督方式运行R-FOCuS（和其他实现）所需的初始参数提出了一个简单的顺序。一般的想法是首先找到双权重损失的κ参数值，并在此基础上调整试用期内的λ阈值。例如，对于每个序列，我们考虑用前*几个*观测值进行训练。为了调整双权重损失的κ参数，我们可以简单地存储初始的w值，如果在这些值的1.5个四分位数范围内有任何观察，*也就是说，*如果我们存在离群值，那么就简单地在这个限制范围内挑选最高出现的κ值。在同一时期，我们估计方差σ^2，并以这个κ值对试用期的归一化数值运行R-FOCuS程序，记录每次迭代的统计数据的轨迹$Q_1, \ldots_w$然后我们设定$\lambda = \kappa \times \max_{i=1,\ldots,w} Q_i$，对于某些$\kappa \in \mathbb{R}^+$。在所提出的应用中，我们在开始时设定$\kappa = 1.5$，然后设定$\kappa = 1.5$。

$\kappa = \log(\tau_s)/\log(\tau_s - \tau_{s-1})$，当检测后重新启动算法时，$\tau_s$ ， τ_{s-1} 是最后一次和之前的最后一次停顿时间。

总的来说，这种估计增加了一个线性的w计算开销，它包括临时存储初始w值和执行κ的估计。