投稿至 *IEEE VIS*。

折线图可视化的数据点选择：方法学评估与循证指南

Jonas Van Der Donckt *† Jeroen Van Der Donckt * Michael Rademaker Sofie Van Hoecke

IDLab，根特大学 - imec，比利时

arXiv:2304.00900v1 [cs.HC] 3 Apr 2023

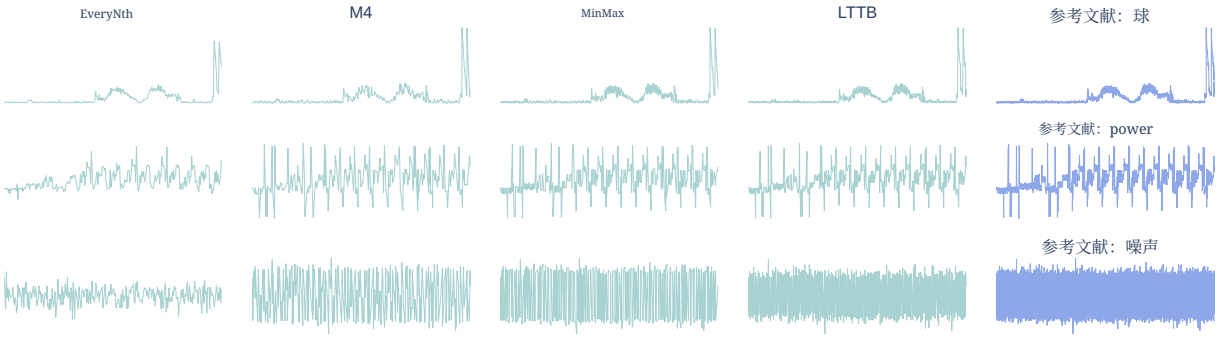


图 1：四种时间序列数据点选择算法的视觉分析。每一行对应一个独特的模板，最右侧列展示由 20 万个数据点构成的参考图像。相邻的四列展示不同的视觉呈现效果，每种效果均由一种数据点选择算法生成，且使用相同数量的输出数据点（$n_{out} = 300$）。所用的可视化属性包括：采用 Plotly 作为工具包，线宽设为 2，并启用抗锯齿。另有 GIF 演示上述可视化效果随n_{out}变化的情况。

摘要

时间序列可视化对于在各个领域识别模式和提取见解至关重要。然而，随着数据集不断增大，有效地将其可视化变得越来越困难。降采样通过数据聚合或数据选择来实现，是应对这一挑战的成熟方法。本文聚焦于数据选择算法，这类算法通过从原始时间序列中选取数值来完成降采样。尽管这些技术已广泛应用于可视化平台和时间序列数据库，关于其评估的文献却十分有限。为此，我们提出了一种全面的、基于指标的评估方法论。该方法论通过评估降采样后的时间序列折线图在视觉上与原始数据的近似程度，分析其视觉代表性。此外，我们还提出了一个名为“视觉稳定性”的新概念，用于衡量更新（流式传输）或与可视化交互（平移和缩放）时发生的视觉变化。我们运用所提出的方法论，在三种开源可视化工具包上评估了四种数据点选择算法，并考量了多种图形绘制属性。根据研究结果，我们为大规模降采样折线图可视化制定了一套循证指南。为促进研究的可复现性，并支持对时间序列数据点选择领域新进展进行定性评估，我们已公开提供本研究的方法论和结果。本文提出的评估方法论及研究所得见解，为该领域的未来研究奠定了基础。

关键词： 时间序列、折线图、降采样、数据聚合、感知、评估、LTTB、M4算法、MinMax。

1 引言

时间序列数据常见于医疗保健、金融和工程等各个领域。这类数据通常包含复杂信息，仅凭汇总统计量无法全面理解，因此可视化是识别模式和提取见解的重要工具。在各种可视化方法中，折线图已被证明在分析不同应用领域的时间序列时尤为有效 [2]。

时间序列数据的规模持续增长，通常包含数百万乃至数十亿个数据点。然而，处理和可视化如此庞大的数据集面临着巨大挑战。因此，在过去十年间，InfluxDB、TimeScaleDB、OpenTSDB 和 Prometheus 等时间序列数据库蓬勃发展，旨在处理并持久化大规模时间序列数据。为了高效地可视化数据，降采样已成为一种成熟的技术，其方法是聚合或选取时间序列的代表性子集，并在降采样点之间进行插值 [1, 2, 21]。降采样在尽可能保留时间序列整体形态的同时减少数据点数量，从而降低网络延迟并缩短渲染时间。因此，它已成为许多广泛采用的时间序列数据库解决方案中的重要组成部分。

本文重点评估数据点选择算法，也称为保值聚合 [16, 17]，这类算法从原始时间序列中选择数据点，以降低数据采样率。考虑到计算效率，我们确定了实践中最成熟的数据点选择算法，包括 EveryNth（每隔 n^{th} 个数据点选取一个）、MinMax、最大三角形三桶算法（LTTB） [31]，和 M4 [16]。这些算法已在工业界广泛采用，例如，Uber 在其 M3 指标平台中将 LTTB 用作降采样函数 [26]，而 TimeScaleDB 则提供 LTTB 作为服务器端超函数 [8]。尽管应用广泛，由于缺乏适用于所有场景的统一指标、可视化组件（例如线宽、线形、工具包）之间相互作用复杂，以及对小幅缩放或平移等有限交互过程中的视觉稳定性关注不足，对这些算法的深入评估仍不充分。

*贡献相同
†电子邮箱：jonvdrdo(dot)(lastname)(at)ugent(dot)be

为应对这些挑战，本文提出一种全面的基于指标的方法，用于评估数据点选择算法的视觉质量，重点关注两个方面：视觉代表性和视觉稳定性。视觉代表性通过采用基于图像的指标，评估降采样后的时间序列折线图在视觉上近似原始数据的程度，从而捕捉工具包和线条绘制特征。视觉稳定性是本文提出的新概念，用于量化两种略有差异的聚合之间的感知连贯性；在逐步更新可视化时，例如数据流、平移或小幅缩放期间，这一概念尤为重要。研究已表明，这些交互方式最有助于以视觉方式探索时间序列数据 [30, 32, 34], ，因此，视觉稳定性是面向可视化的时间序列降采样中不可或缺的方面。此外，我们的方法纳入了多种时间序列模板，并主张将选定数据点的数量作为因变量，从而能够对数据效率进行深入比较。

应用所提出的方法论，我们在三种可视化工具包中评估了前述四种数据点选择算法，并考察了线宽和抗锯齿等多种图形绘制属性。基于评估结果，我们为采用降采样的大规模折线图可视化制定了一组指南。

总而言之，本文的贡献有三点。首先，我们提出了一种稳健且全面、基于指标的数据点选择算法评估方法论，用于量化视觉代表性以及一个名为“视觉稳定性”的新概念。其次，我们运用所提出的方法论评估了四种重要的降采样算法，同时考虑了多种可视化工具包和各类图形绘制属性。我们将分析所得见解归纳为一组指南，供采用降采样的大规模时间序列可视化使用。最后，我们提供了所提出框架的高质量开源实现及相应结果，以确保可复现性，并为分析和改进新型降采样算法铺平道路。该实现可在 github.com/predict-idlab/ts-datapoint-selection-vis 获取。

2 相关工作

本节全面概述该领域的相关工作。我们首先考察时间序列可视化技术，然后讨论数据聚合的高层方法。具体而言，我们重点关注保值数据聚合算法，同时考量其实用适用性及该领域现有的研究。通过这种方式，我们旨在清晰阐明当前的最新进展，并找出本研究致力于解决的不足。

2.1 时间序列可视化

可视化对于理解时间序列数据至关重要，尤其是考虑到这种数据模态的复杂性。毕竟，人们常常认为人眼是终极的数据挖掘工具 [23]。研究发现，对于大多数时间序列分析任务而言，简单的可视化形式（如基于折线的图表）已足够，而不必采用更复杂或更具体的方法 [2]。因此，本研究将聚焦于折线图可视化的代表性。

交互式可视化技术对于探索大量时间序列数据至关重要。Shneiderman 等人提出了可视信息寻求准则，强调应先探索数据概览，再通过缩放和筛选获取所需细节 [30]。此外，Walker 等人指出，交互式折线图可视化能够提高执行可视化导向任务的效率 [34]。

在先前的工作中，我们开发了 plotly-resampler。它是 Plotly 可视化库 Python 绑定的封装，通过在后端处理图形交互回调来实现缩放绘图功能 [32]。与其他现有工具相比，plotly-resampler 的两大优势是无需数据库管理系统，并且支持在轨迹级配置数据聚合配置。本研究所得结果可直接应用于 plotly-resampler。

2.2 折线图可视化的数据聚合

数据聚合技术已成为应对大数据和可视分析挑战的关键手段 [5, 12, 21]。这些方法能够解决可视化大型数据集时遇到的若干挑战。具体而言，传输和呈现的数据点越多，前端渲染速度和响应能力就会显著下降，因此数据聚合至关重要。此外，可视化大量数据的一个常见副作用是过度绘制，即多个数据点因可视化画布尺寸有限而相互重叠 [4, 34]。数据聚合利用了这一现象，有可能省略部分过度绘制的数据点。

可视化的数据聚合有两种不同方法：降采样和按密度聚合。降采样会输出一个低维的代表性数据集，而按密度聚合则采用共享的密度颜色编码来渲染数据图像。降采样还可进一步分为保值聚合和特征数据聚合。下文将讨论按密度聚合和特征数据聚合；第 2.3 节将深入探讨保值聚合，这是本文的重点。

2.2.1 按密度聚合

按密度进行数据聚合，是指使用统一的密度颜色编码来渲染数据图像 [6]。这种技术适用于地图和点云等单一的大规模数据模态的可视化，但当数据中包含多个不同的模态时，就会面临挑战。此时，每种模态都需要使用可区分的密度颜色编码，这往往会导致过度绘制。然而，时间序列数据通常由不同的模态构成，因此这种技术并不那么适用。此外，将数据转换为图像会限制子图内的区分度和交互性，例如切换时间序列轨迹的功能，从而进一步降低其在时间序列可视化中的适用性。

Datashader 是目前已知唯一适用于时间序列数据（即折线图）的按密度数据聚合实现 [14]。

2.2.2 特征聚合

特征数据聚合通过突出属性或趋势来减少数据量，使用均值、中位数和平滑等方法。由于这些技术的目标无法客观量化，因此需要通过用户研究来评估其代表性。

Control 项目于20世纪90年代推出，是最早将抽样用于可扩展聚合查询的项目之一，例如 SUM、COUNT 和 AVG [13]。这种方法可以快速可视化近似查询结果，同时允许用户指定置信区间。通过传达结果不确定性，用户便能信任可视化结果 [21]。

此外，Rong 等人提出了 ASAP（注意力优先级自动平滑）[27], ，该方法通过自适应优化降噪和趋势保留来平滑时间序列，引导用户关注显著偏差。

2.3 保值算法

保值数据聚合会从原始时间序列中选取数据点，同时力求保留其整体形状。为此，这类技术会利用选定数据点之间的插值。

与特征聚合和按密度聚合技术不同，保值聚合的目标是近似

表 1：数据点选择算法概览，其中 N表示时间序列长度，n_{out} 表示聚合输出大小。

	(i) Time	(ii) 内存	(iii) 可并行化	数据点 每个数据桶
EveryNth	$O(n_{out})$	$O(1)$	✓	1
MinMax	$O(N)$	$O(1)$	✓	2
M4 [16]	$O(N)$	$O(1)$	✓	±4
LTOB [31]	$O(N)$	$O(1)$	✓	1
LTTB [31]	$O(N)$	$O(1)$	×	1
ModeMedianB [31]	$O(N)$	$O(N/n_{out})$	±	1
Visval [33]	$O(N \log(N)$</style	$\Omega(N)$	±	1
动态规划 [9]	$O(N \log(N)$</style	$\Omega(N)$	±	1
MinStdErrB [31]	$O(N^2)$	$O(N^2)$	×	1
LLB [31]	$O(N^2)$	$O(N^2)$	×	1

真实图像。真实图像也称为模板图像或参考图像，是包含所有数据点 [16, 31] 的可视化。因此，可以使用图像质量评估指标 [36] 来评估保值方法的视觉代表性。请注意，真实图像可能存在过度绘图。

2.3.1 实际适用性

对于现实世界中的大规模时间序列数据，降采样算法必须满足特定的计算要求，包括：(i) 运行时间复杂度至多为$O(N)$，即对数据进行一次遍历；(ii) 避免使用随数据规模增长的额外数据结构，因为维护和更新这类结构在计算上不可行；以及 (iii) 支持并行化，通过将数据分区分配给多个线程或处理器核心，缓解线性扩展带来的负担。

表 1 总结了满足值保留要求的时间序列可视化算法在上述三项要求方面的情况。请注意，空间复杂度指输入和输出数组之外所需的额外内存。只有水平线以上的五种算法满足所提出的要求。其中，最大三角形单数据桶算法（LTOB）和最大三角形三桶算法（LTTB）都基于三角形面积最大化。然而，由于 LTOB 是 LTTB 的一种短视变体（仅考虑相邻的两个数据点） [31]且 Gil 等人 [11] 证明了 LTTB 优于 LTOB，因此本研究未纳入 LTOB。

水平线以下的算法无法满足所提出的要求。表中包含两种折线简化算法：道格拉斯-普克算法（动态规划） [9] 和 Visvalingam-Whyatt算法（Visval） [33]。由于其时间复杂度上界为 $O(N\log(N)$，且使用 $\Omega(N)$ 数据结构来限制重复计算，这两种算法都不适用于大规模时间序列可视化。此外，表中还列出了 Steinarsson [31] 提出的另外三种不适用的算法，其中众数-中位数桶算法具有不理想的 $O(N/n_{out})$ 空间复杂度，而最小标准误差桶算法的运行时间和空间复杂度均为 $O(N^2)$。

在以下小节中，我们将讨论四种适用于大规模时间序列可视化的保值算法（不包括 LTOB），它们构成本研究的重点。

2.3.2 EveryNth

EveryNth 算法也称为抽样或抽取，是一种高效的保值降采样方法。该方法从输入数据数组中每隔n^{th} 个数据点选取一个，其中n 是输入数组长度（N）与目标输出数组长度（n_{out}）的比值。该算法的时间复杂度为$O(n_{out})$，空间复杂度为$O(1)$。值得注意的是，在表 1 所列算法中，只有该算法的运行规模随n_{out} 线性增长，而非随N线性增长。尽管该算法易于并行化，但由于抽样任务本身很简单，多线程可能无法带来显著收益。

2.3.3 MinMax

MinMax 通过为每个数据桶选取垂直极值，旨在保留时间序列数据的形状。该算法会选取每个分箱中的最小值和最大值，但不强制形成交替模式。因此，可能会连续出现相邻的局部最小值（或最大值），从而无法正确捕捉底层的交替变化。MinMax 的时间复杂度为$O(N)$，空间复杂度为$O(1)$。该算法易于并行化，因此可在多核处理器上高效执行。

2.3.4 M4算法

Jugel 等人 [16],提出的 M4 是一种保值聚合方法，可实现像素级精确数据聚合。对于每个分箱，它会选取两个水平极值（首个、末个）和两个垂直极值（最小值、最大值）。选取这些数据点的时间复杂度为线性（$O(N)$），空间复杂度为 $O(1)$。该算法也易于并行化。值得注意的是，M4 在每个分箱中选取的数据点多于其他算法。因此，Bae 等人提出了基于像素间梯度的 M4（IGM4） [3], , 通过后处理步骤移除冗余数据点。这使选定数据点的数量平均减少了 20%，但每个分箱中的数据点仍明显多于其他降采样算法。

2.3.5 最大三角形三桶算法（LTTB）

Steinarsson 提出了几种保值降采样算法 [31]。本文聚焦于 LTTB 算法，因为与 LTTB 相比，其他提出的算法要么时间复杂度不理想，要么视觉代表性较差，见表 1。LTTB 基于有效三角形面积这一概念，该概念常用于折线简化算法。它会在每个数据桶（即分箱）中选取数据点，使其与前一个已选数据点及下一个数据桶的平均值构成的三角形面积最大。LTTB 的时间复杂度为 $O(N)$，空间复杂度为 $O(1)$。尽管其运行时间随数据规模线性增长，但由于面积计算成本较高，其增长斜率大于 MinMax 和 M4。LTTB 无法并行化，因为它需要按顺序遍历数据：前一个已选数据点是局部面积计算的一部分

2.4 评估时间序列数据聚合

已有多项研究着手评估数据聚合在时间序列可视化中的视觉代表性。

Shi 等人 [29] 提出了两种基于数据的指标，用于评估制图折线简化：位移度量（简化折线与原始折线之间的最大垂直距离）和形状失真度量（简化折线与原始折线之间倾角的平均差异）。

Jugel 等人 [16] 使用结构相似性指数（SSIM）作为基于图像的指标，来评估并比较他们的 M4算法数据聚合算法。然而，Nilsson 等人 [25] 和 Sara 等人 [28], 均对 SSIM 的可靠性提出质疑，并指出基于 SSIM 评估图像质量可能导致错误结论，且无法始终准确反映重建误差。

Gil 等人 [11] 提出了一种方法，通过使用多种保值算法和基于数据的误差评分，自适应地识别适用于时间序列压缩的最优数据点选择算法。然而，该方法存在局限性：未纳入 MinMax，仅评估了平均绝对误差（MAE），且将降采样信号插值至原始信号会导致$O(N)$空间复杂度。此外，他们的结果表明，使用 LTTB 或 M4算法仅比最优策略略逊一筹，这令人质疑该方法的实际适用性。

据我们所知，此前尚无研究评估降采样方法在更新（流式传输）或与可视化交互（平移和缩放）时的稳定性。不过，

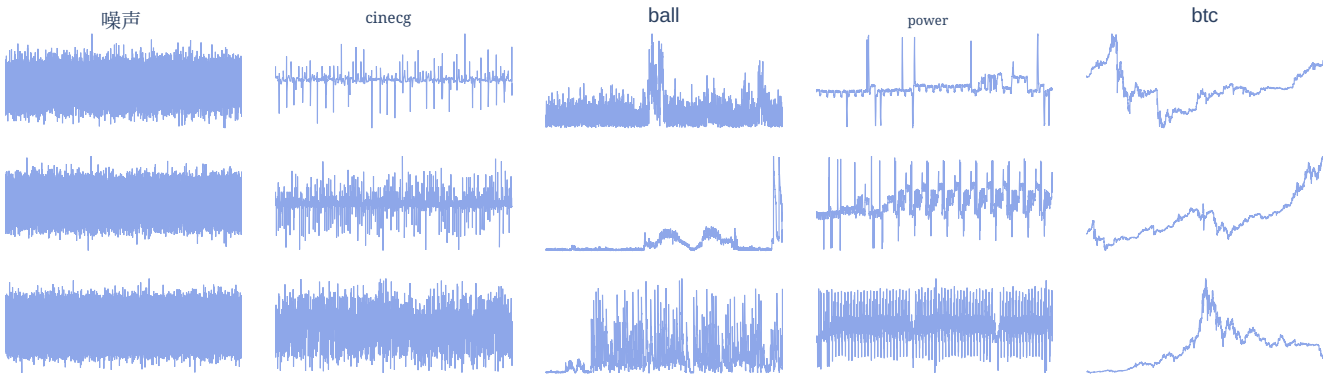


图 2：所用时间序列模板网格。每列代表一个不同的时间序列，行表示模板尺寸。

如第 2.1 节所述，交互性是探索时间序列可视化的最高效方式。因此，视觉稳定性将是我们所提方法论的两个核心方面之一。

3 方法论

本研究旨在评估并比较用于可视化的时间序列数据点选择算法。具体而言，我们关注两个方面：视觉代表性和视觉稳定性。

视觉代表性 指降采样后的时间序列图在视觉上反映原始渲染数据的程度。为评估这一点，我们使用三种基于图像的指标，并通过 OR 卷积掩码对每种指标进行聚合（即缩放）。

视觉稳定性 旨在捕捉缩放、平移或流式传输时发生的视觉变化，也就是在视图大部分重叠的情况下重新渲染时发生的变化。为评估视觉稳定性，我们使用两种侧重于残差误差的数据空间指标。

为确保对降采样算法进行稳健且公平的比较，我们的方法论采用多个能够代表广泛时间序列数据集的时间序列数据集。此外，我们主张以降采样点的数量衡量数据效率，以便于跨算法比较。

本节首先介绍研究设计，包括时间序列模板以及应如何评估数据效率。随后，我们说明用于评估视觉代表性的图像空间指标，并介绍用于评估视觉稳定性的数据空间指标。

3.1 研究设计

我们的方法论采用多个时间序列模板，涵盖一系列时间序列属性，包括周期性和噪声程度。这些时间序列模板的长度各不相同，从而全面呈现同一数据源在不同缩放级别下的情况。此外，我们建议使用 n_{out} 作为比较降采样算法数据效率的公平指标。

我们研究设计的一个关键方面是对所有可视化采用固定的画布尺寸。本研究依照 Jugel 等人的做法，选择了一个 (800, 250 [16]) 画布。通过在改变n_{out}的同时保持画布尺寸不变，我们确保比较具有一致性，从而能够得出有关n_{out} 与画布宽度之间关系的结论。此外，由于本研究仅关注单变量时间序列可视化，即仅涉及单条线的图表，因此我们选择以灰度渲染折线图可视化。作出这一决定是因为无需在同一绘图中区分多种模态。因此，生成的可视化图像是二维矩阵，其中每个像素仅包含一个亮度值（$[0, 255]$）。

3.1.1 时间序列模板

为确保全面评估，我们的方法论采用多个时间序列模板，涵盖广泛的时间序列属性，例如周期性、平稳性、噪声程度、时间序列在可视化画布上的覆盖面积，以及高频成分的存在。这些模板也称为图像模板，表示特定尺寸N的信号。这些模板中N变化的范围全面呈现了各种缩放级别，而缩放级别是与时间序列数据交互的一种主要方式 [2]。我们通过实证分析选定了$N \in \{50K, 200K, 1M\}$ 这些尺寸。这些尺寸被认为适用于可视化（即能够渲染参考图像而不会导致程序崩溃），同时也能提供有参考价值的数据视图。

我们的研究采用了五种不同的时间序列信号，如图 2 所示。其中包括：(1) 生成的高斯噪声，代表一种高频平稳噪声；(2) 来自 UCR 时间序列档案库的心电信号，具有显著的峰值和周期性 [7]，由 Franco 等人 [10]使用；(3) 来自 DEBS 2013 数据集的正向峰值球速信号 [24]，由 Jugel 等人 [16, 17]使用；(4) 来自 DEBS 2012 数据集的周期性功耗信号 [15]，呈现显著峰值，由 Jugel 等人 [16, 17]使用；以及 (5) 根据 Steinarsson 使用的股票数据构建的非平稳信号，该信号由 1 分钟间隔的历史比特币数据组成 [20]，[31]。图 2 显示，随着心电信号和功耗信号的模板大小 N 增加，可见的峰值和周期也随之增多。值得注意的是，球速时间序列信号在 N 变化时会呈现截然不同的模式。

3.1.2 数据效率

评估降采样算法的有效性时，关键是根据输出数据点的数量 n_{out} 对其进行比较。这个数量直接关系到传输至前端并在其中显示的数据量，进而影响可视化性能。以往研究关注的是降采样所用的分箱数量 [16, 17]，但忽略了不同算法在每个分箱中的数据点数量可能不同这一事实（如表 1 所示）。因此，我们建议将 n_{out} 作为比较降采样算法的公平指标。

3.2 视觉代表性

视觉代表性衡量降采样时间序列图在视觉上与原始渲染数据（也称为参考可视化）的接近程度。为评估这一点，我们将降采样时间序列的可视化与参考可视化进行比较，这相当于比较图像。因此，我们采用三种能反映比较不同方面的图像空间指标：均方误差、像素误差（带容差范围）和 DSSIM。这三种指标均属于全参考图像质量评估指标，并被归类为空间域方法 [36]。

我们的目标是通过均方误差和像素误差捕捉绝对误差，同时通过（D）结构相似性指数评估感知质量。尽管均方误差常用于衡量信号保真度，但有人批评它未能考虑某些图像信号特征和人类感知 [35][1]。因此，鉴于人类感知对图像中的结构信息高度敏感，我们采用结构相似性指数的一种差异度变体。借助图像空间指标，我们可以探究影响可视化的各种参数。这些参数包括但不限于光栅化后端、绘图工具包、线条绘制样式、线宽和抗锯齿。尤其值得关注的是，这种方法还可以评估可视化工具包所采用的默认配置。

下面，我们介绍用于评估视觉代表性的三种图像空间指标，随后描述我们使用的 OR-卷积掩码，该掩码将三种逐像素指标的输出分别汇总为单一分数。

3.2.1 均方误差

用于评估视觉代表性的第一种指标是均方误差（MSE）。我们针对每个像素计算平方误差，再将 OR-conv 掩码内所有像素的误差汇总为单一数值。该指标衡量像素的绝对误差，但与平均绝对误差不同，均方误差对抗锯齿造成的细微着色误差惩罚较轻。

3.2.2 像素误差

用于评估视觉代表性的第二种指标通过二值化来量化像素误差。当下采样后可视化中的像素值与参考可视化不同时，该像素即被判定为误差（赋值为 1）。为将像素误差汇总为单一指标，我们先计算像素误差总数，再除以 OR-conv 掩码的面积进行归一化。与均方误差不同，该指标对像素值的细微差异也给予同等惩罚。然而，对于混叠图形，这种一视同仁的惩罚可能降低指标的可解释性，因为它会惩罚人类感知无法察觉的细微着色差异。为此，我们在像素差异上引入 20 的边界，仅纳入可见误差。该指标称为像素误差边界 20（PEM 20）。

3.2.3 （D）结构相似性指数

作为第三个指标，我们使用结构差异度（*DSSIM*），即结构相似性指数（*SSIM*）的缩放版本；Jugel 等人也采用了该指标 [16, 17]。*SSIM* 是评估感知质量时常用的指标 [25], ，其产生的相似度得分范围为 -1 到 1。得分为 1 表示完全相似，0 表示不相似，-1 表示完全反相关。因此，图像V_1 与 V_2 之间的差异度定义为：

$$DSSIM(V_1, V_2) = \frac{1 - SSIM(V_1, V_2)}{2}$$

与前面的指标一样，(*D*)*SSIM* 的输出是一个矩阵，其中每个像素都对应一个值。Jugel 等人通过对逐像素值应用全局均值[2]，将其汇总为单一指标。然而，在本研究中，我们应用 OR-conv 掩码，仅对折线图所在区域的*DSSIM* 值取均值，参见第 3.2.4 节。

[1]请注意，PSNR 与这一特定任务无关，因为图像的最大像素值始终为 255，这会导致显示出与均方误差相同的趋势。[2]Jugel 等人向作者确认，他们对 [16,17], 中的*DSSIM* 进行了数据反转，实质上将 *DSSIM* 从 $[0:1]$ 转换为 $[1:0]$。

3.2.4 OR-conv 掩码

我们提出一种名为“OR-conv 掩码”的新技术，以更有针对性地汇总逐像素指标。通常，全参考图像质量评价指标通过全局均值在整幅图像范围内进行汇总。然而，折线图通常只占画布的一小部分，若使用全局均值进行汇总，得到的指标会被平滑化，并受到画布填充程度的影响。为解决这一问题并便于进行跨模板比较，我们提出了 OR-conv 掩码。该技术仅汇总可视化中渲染的数据点邻域内的逐像素值。更多详情请参阅 GitHub 仓库中的说明，其中包含示例和实现细节。

3.3 视觉稳定性

视觉稳定性用于评估在数据增量有限的情况下更新显示视图时发生的视觉变化。鉴于交互性对于有效探索时间序列数据至关重要，降采样算法的稳定性是一个需要考虑的重要方面 [32], 。这一点也体现在 Shneiderman 的可视信息寻求准则中：“先概览，再缩放和筛选，然后查看细节——按需获取” [30]。

视觉稳定性评估主要包括两种情况：平移和缩放。平移是指将显示的数据范围偏移一个量 $\Delta \neq 0$，从而得到范围 $[x_{start} + \Delta, x_{end} + \Delta]$。正偏移量表示向右平移，负偏移量表示向左平移。缩放会将数据范围更改为 $[x_{start} + \Delta_l, x_{end} + \Delta_r]$，且至少有一个 $\Delta \neq 0$ 和 $\Delta_l \neq \Delta_r$（因为 $\Delta_l = \Delta_r$ 是平移）。缩放还可进一步分为两种子情况：放大（$\Delta_l \geq 0, \Delta_r \leq 0$）和缩小（$\Delta_l \leq 0, \Delta_r \geq 0$）。左侧偏移量为正表示向左放大，右侧偏移量为负表示向右放大，而二者组合则表示左右同时放大。相反，右侧偏移量为正、左侧偏移量为负，分别表示向右缩小和向左缩小；二者组合则表示左右同时缩小。请注意，流式传输对应于 $\Delta_l = 0$ 和 $\Delta_r > 0$的缩小。此外，值得注意的是，视觉稳定性与视觉代表性互为补充：后者关注较大幅度的缩放（因模板大小N不同而异），前者则侧重于较小幅度的更新。

视觉稳定性主要受所选降采样算法的影响，而不是可视化工具包的具体配置。尽管可以采用图像空间方法评估稳定性，但该方法面临诸多挑战，例如需要像素完全重叠，以及在更新过程中加入新的全局极值后无法更改 y轴范围。为避免这些局限并有效量化稳定性，我们采用数据空间指标。具体而言，我们首先在更新后的降采样数据与原始降采样数据的交集处，对更新后的数据进行插值，从而能够在相同的 x 值处进行数据空间比较。随后，我们计算两个稳定性指标：平均绝对误差（MAE）和峰值处平均绝对误差（MAEP）。需要注意的是，这些度量指标用于评估相对于原始降采样数据的稳定性，并不考虑降采样时间序列的视觉代表性；后者通过视觉代表性指标进行度量。

3.3.1 平均绝对误差（MAE）

平均绝对误差（MAE）用于量化原始降采样数据与更新后的降采样数据之间的平均差异。为了使不同模板之间的绝对误差更具可比性，我们将其除以原始降采样数据的最小-最大范围进行归一化。该指标考虑了所有降采样数据点，整体反映绝对差异。不过，由此也会导致该指标容易将少数较大的误差平滑掉。

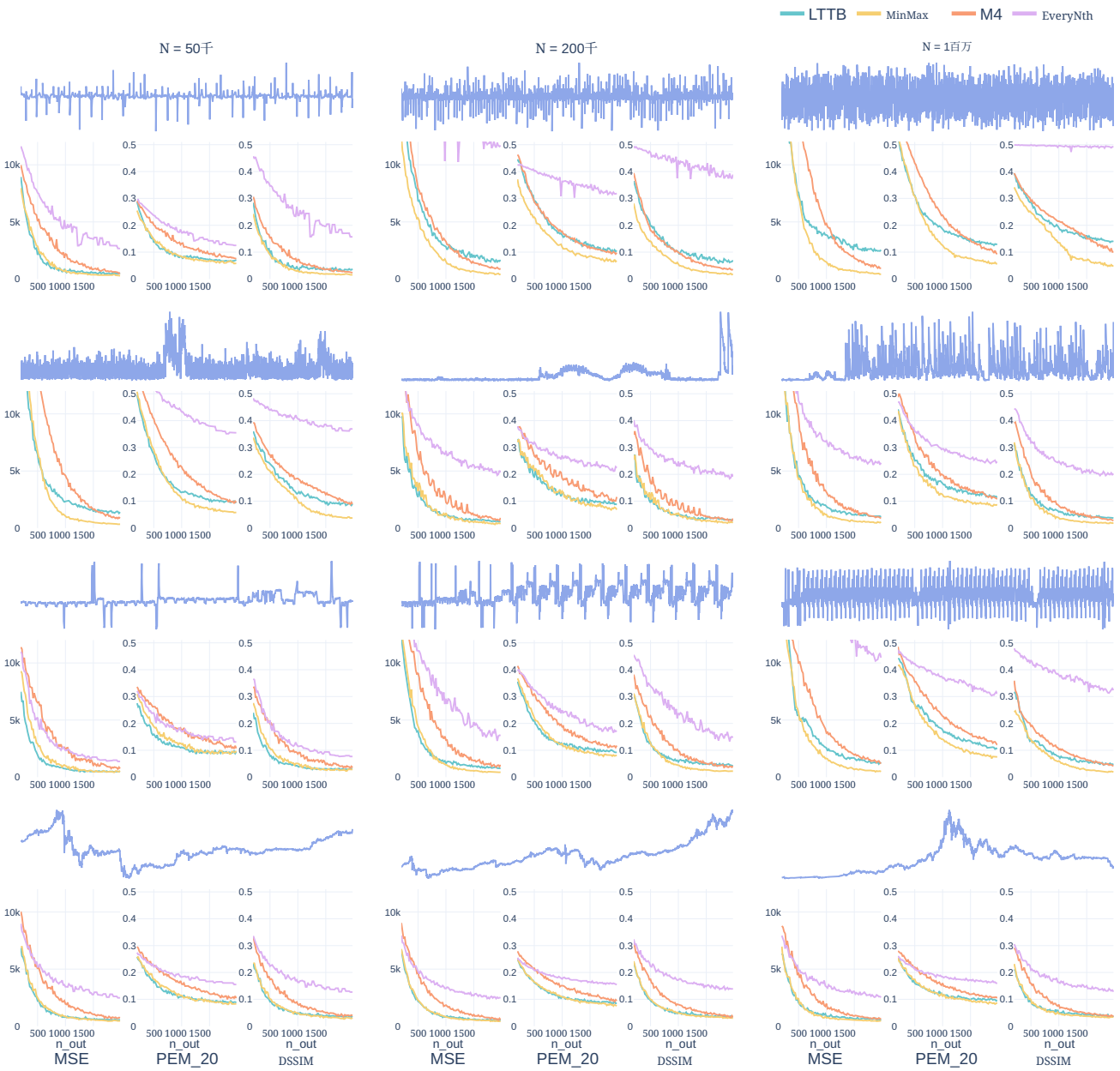


图 3：评估不同图像模板下数据点选择算法的视觉代表性。每行显示一个不同的时间序列数据集，各列表示模板大小。所有图像模板（包括显示在指标子图上方的蓝色参考模板）均使用 Plotly 的默认设置生成（线性插值，线宽为 2）。指标子图显示了随着n_{out}（x 轴）增大（范围为 [200, 2000]），聚合方法性能的变化趋势。*PEM* 20 指边界值为 20 的像素误差：将差异超过 20 的像素二值化，并除以卷积掩码大小以计算比率。*DSSIM* 表示 OR 卷积掩码缩放结构相异度（见第 3.2.3 节），而 MSE 表示卷积掩码缩放均方误差。

3.3.2 峰值平均绝对误差（MAEP）

峰值平均绝对误差（MAEP）的计算方式与上文所述的平均绝对误差（MAE）类似，但只考虑数据中的峰值。峰值定义为原始聚合数据中的局部最大值或局部最小值，在时间序列中比其他部分更为显眼 [19]。该指标与 Shi 等人 [29], 提出的最大垂直距离部分一致，旨在捕捉最大的偏差。因此，相较于 MAE，MAEP 对视觉稳定性的衡量更为敏感，因为它只关注数据中视觉上最显眼的部分。

4 结果

本节将介绍并讨论应用所提出的方法论，并考察 EveryNth、MinMax、LTTB 和 M4 算法时得出的研究结果。下面两个小节将进一步探讨我们方法论的两个方面。

4.1 视觉代表性

我们使用三种广泛应用的 Python 可视化工具包评估视觉代表性：Matplotlib、Plotly 和 Bokeh。选择这些工具包，是因为它们是 Python 中最具影响力的可视化工具，而 Python 是用于与大规模

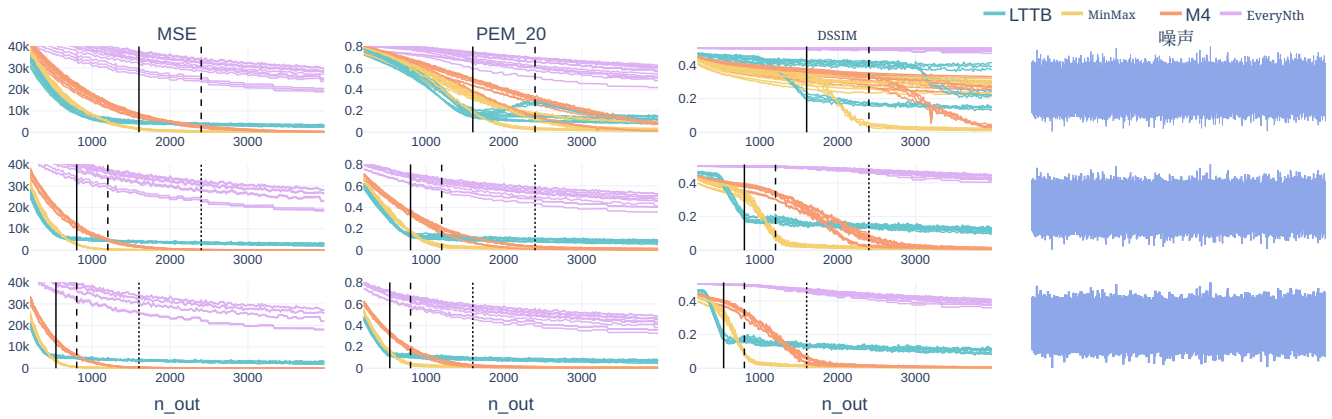


图4：不同线宽（1、2、3）、数据规模（50k、200k、1M）和工具包（Plotly、Bokeh、Matplotlib）下噪声的视觉代表性，n-4000。聚合图像与线宽相同的参考图像进行比较。

数据 [18]互操作的主流语言。具体而言，我们通过两种方法考察数据点选择算法的视觉代表性。

首先，我们保持折线图配置不变，同时改变图像模板，重点考察 Plotly 默认配置。这种实验设计贴近用户通常不会调整这些设置的实际情况。通过这一分析，我们可以识别不同模板下各类降采样算法在数据效率方面的趋势，避免不同配置造成干扰。

其次，我们采用固定的时间序列模板，具体为平稳噪声信号，以研究不同可视化库和线宽对数据点选择算法视觉代表性的影响。

最后一个小节探讨了在实践中实现像素级完美的 M4 聚合算法时的一些具体问题。

4.1.1 数据效率与总体趋势

图 3 展示了 Plotly 默认配置下四种数据点选择算法的视觉代表性指标。通过绘制这些指标随n_{out}（各指标子图的 x 轴）变化的情况，我们可以评估数据效率，即这些算法在特定（较低）n_{out} 取值下的表现。代码仓库 [3] 中提供了采用 Plotly 默认配置为另外两种工具包绘制的类似图示。

总体观察表明，对于每个模板子图，各聚合器在不同指标下的趋势和排序都相对一致。作为第二个符合预期的趋势，我们注意到，增大n_{out} 会提升视觉代表性。就聚合器的排序而言，MinMax 和 LTTB（视模板而定）通常表现最佳，M4 排名第三，而 EveryNth 表现最差。此外，在每项指标中，LTTB 和 MinMax 曲线的形状往往相似，拐点出现在同一位置，但斜率略有不同。这些观察结果对另外两种工具包同样成立，详见脚注 3。

在数据效率方面，MinMax 和 LTTB 表现突出，是数据效率最高的聚合器，在 $n_{out} < 1000$ 时取得了最佳指标值。根据模板不同，其中一种算法可能略胜另一种。这些发现与 Gil 等人 [11], 的报告一致，他们的结果表明 LTTB 是一种数据效率较高的算法。MinMax 在 cinecg-200k、cinecg-1M、ball-1M 和 power-1M 等粗糙模板上表现出色。对于所有其他模板，我们观察到它们在 $n_{out} < 800$ 时表现相同。随着 n_{out} 不断增加，我们注意到 MinMax 的表现逐渐接近 LTTB，甚至低于 LTTB。尤其是在高粗糙度图像模板（例如 ball-50k、cinecg-1M）上，MinMax 的指标值会降至低于 LTTB（以及 M4算法）。对于低粗糙度数据，例如所有比特币模板，即使在较大的 n_{out} 下，LTTB 和 MinMax 的表现也完全相同。M4算法和 EveryNth 的数据效率较低，且收敛速度明显慢于 MinMax 和 LTTB，这符合预期：EveryNth 采用简单直接的选取方式，而 M4 算法每个分箱输出 4 个选定数据点，降低了其数据效率。

对于大多数图像模板，PEM 20 指标会收敛至 0.1，这可归因于混叠造成的着色误差 [4]。这一观察也得到以下事实的支持：当n_{out}较大时，均方误差会降至接近零，表明较小的（阴影）误差是 PEM 20 值为 0.1 的主要原因。请注意，由于 PEM 20 按 OR 卷积掩码的大小进行缩放，对于占据更多画布空间的图像模板，其值下降得更多；这是因为在填充区域的外缘，阴影所占的相对空间更少，例如 ball-50k、cinecg-1M 和 power-1M。

4.1.2 数据效率与线宽

图 3 展示了平稳噪声时间序列信号的视觉代表性指标，其中每一行对应一种不同的线宽。这些可视化涵盖了全部三种工具包，子图中的模板大小 N 有所变化，因为对于平稳噪声数据，模板大小对图像模板的影响极小。

一个关键发现是，对于每个指标子图，聚合器性能曲线在不同数据规模和工具包下均呈现一致的趋势。值得注意的是，线宽为 1 时，DSSIM 和 PEM 20 子图中出现了一些不一致之处；我们已在 GitHub 仓库中提供详细说明，见脚注 3。

对于这一时间序列信号，LTTB 是数据效率最高的方法，这可归因于它对三角形面积的最大化。该算法倾向于在每个分箱的最小值和最大值之间交替取值，因此特别擅长覆盖较大区域。在预告图 1 中，LTTB 对噪声数据进行聚合时，最小值和最大值之间的这种交替十分明显。尽管如此，LTTB 并不会收敛到最小值；这同样可以用算法偏好最小值与最大值交替来解释，因为这种偏好可能导致它排除由 MinMax 和 M4 选取的全局极值。因此，M4 最终会收敛到与 MinMax 相同的值，这在 $lw = 3$ 行中清晰可见。此外，值得注意的是，M4 和 EveryNth 的收敛速度往往要慢得多。

对于每种聚合器，我们在 PEM 20 和 DSSIM 指标的性能曲线上观察到一个拐点，即曲线趋于收敛的位置。我们注意到，该拐点的n_{out} 位置会随线宽增加而减

[3] github:ts-datapoint-selection-vis/details/visual-representativi ty

[4] 此 GIF 展示了混叠对 PEM 20 的影响。

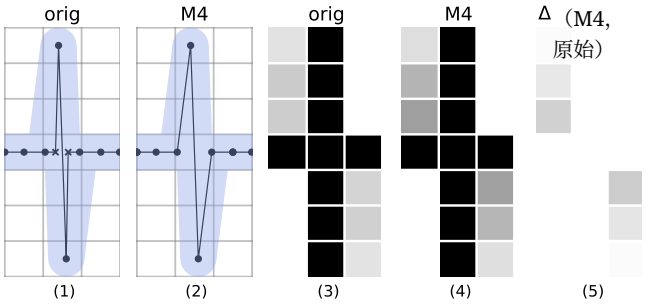


图 5：应用抗锯齿时与 M4 相关的像素误差可视化。子图（1）展示了带有 1 像素宽阴影的原始时间序列。在子图（2）中应用 M4 后，（1）中以 ×-符号标记的两个数据点被排除。子图（3）和（4）分别展示了原始时间序列和 M4 聚合时间序列的抗锯齿渲染结果，而子图（5）则突出显示了二者之间的差异。

少。这表明，增加线宽可以提高数据效率。进一步研究后，我们推导出一个通用公式，用于根据画布宽度（cw）和线宽（lw）确定所观察到的拐点的具体n_{out} 位置：

$$elbow = \frac{cw * \bar{n}_{ga}}{lw}$$

此处，$\bar{n}^{ga}$ 表示达到保证交替所需的选定数据点数量。由于 LTTB 倾向于最小值与最大值交替，$\bar{n}^{ga}$ 为 2。相比之下，MinMax 最多可能需要 3 个数据点才能实现交替，因此其 $\bar{n}^{ga}$ 为 3。M4 可视为 MinMax 的一种变体，它还会选取数据桶中的第一个和最后一个数据点，因此所需的 $\bar{n}^{ga}$ 是 MinMax 的两倍，即 6。图 4 使用实线、虚线和点划线分别表示 $\bar{n}^{ga}$ 值为 2（LTTB）、3（MinMax）和 6（M4）时的拐点位置。随着线宽增加，数据效率提高的趋势在图像模板网格 [5] 中也很明显。

4.1.3 像素级完美的 M4

前几个小节重点讨论了如何提高视觉代表性方面的数据效率，这并非 M4 的突出优势，但其像素级完美性才是。正如 Jugel 等人所指出的，实现像素级完美的 M4 需要一个宽度为画布 [16] 像素宽度四倍的 n_{out}。本小节将探讨使用 M4 算法创建像素级完美聚合所需考虑的实际问题。

我们首先关注抗锯齿，也称为像素二值化，这是提高可视化感知质量的常见技术 [22]。本研究考察的三种工具包均默认启用抗锯齿。此外，Bokeh 和 Plotly 均不提供渲染混叠图形（即禁用抗锯齿）的选项。对于折线图等简单图形，一种直接有效的抗锯齿方法是，确定矢量图形中给定区域的像素占用率，并以该百分比作为颜色阴影。图 5 运用这一方法说明，启用抗锯齿时，M4 无法生成像素级完美的可视化。子图 (2) 表明，使用 M4 移除数据点会导致阴影线的斜率和起点出现细微偏差；由于像素区域占用率的百分比不同，抗锯齿像素的着色也随之产生差异 (5)。值得注意的是，子图 (3) 和 (4) 是使用 Matplotlib 默认的栅格化后端构建的，展示了 M4 聚合对该工具包的直接影响。

影响 M4 像素级精确度的另一个可视化配置因素是线宽。增大线宽会导致更大的阴影区域，从而可能造成更严重的像素着色误差。Bokeh 的默认线宽为 1 像素，Plotly 的默认线宽为 2 像素，而 Matplotlib 以点为单位定义线宽，相当于图形每英寸点数的 $1/72$。因此，若不手动调整这些工具包中的线宽，就很难实现像素级精确的 M4。

除了配置混叠和线宽之外，我们还发现，要实现像素级完美的 M4算法，需要修改其他可视化相关参数，例如光栅化后端和 x 轴范围分箱。

最后，要建立分箱与像素列之间的映射（$n_{out} = 4 * cw$），必须提前控制画布宽度或获知其数值，这并非总是易事。图表布局元素，包括 y 轴刻度、图例和字号，都会影响最终的画布宽度。此外，这些元素可能会因更新而改变，例如用户与图表交互后 y 轴刻度发生变化，这使得画布宽度的评估更加复杂。此外，基于 Web 的可视化框架（如 Plotly 和 Bokeh）可能会延伸至整个网页，并受到浏览器页面缩放和缩放操作的影响。

总之，鉴于实现像素级完美的可视化需要对可视化环境进行大量控制，在大多数现有可视化工具包中，使用 M4算法实现像素级完美的可视化十分困难，甚至不可能。Jugel 等人提到，“在实际实现中，工程师必须确保 $n_{out} = 4 * cw$，以获得最佳效果” [16]，但这只是他们论文中的一个次要论点。Bae 等人展示了在 D3 中实现 M4算法的一种实用方法，但既未提及抗锯齿，也未验证其实现是否达到像素级完美 [3]。如上所示，影响 M4算法性能的因素比最初预想的更多。因此，我们认为，M4算法不应仅被视为另一种数据点选择算法，而应被视为一种数据高效可视化技术，因为它与数据点的渲染密切相关。

4.2 视觉稳定性

如方法论部分所述，视觉稳定性分为两种主要情况：平移和缩放。缩放还可进一步分为放大和缩小（后者在某种程度上对应于流式传入新数据）。因此，我们分析三种不同情况：平移、放大和缩小。偏移量以比例形式定义，并基于N 计算，具体取值如下：$[1/53, 1/27, 1/17, 1/11, 3/23, 2/11]$。请注意，这些选定的偏移比例的分母均为素数，这比对齐偏移量更符合现实世界中的用户交互。例如，若选择 0.1 作为偏移比例，则交集上的分箱将完全对齐，从而使 MAE(P) 指标为 0。

图 6 展示了五个数据集的视觉稳定性结果，每一行代表一个长度为 200k 的时间序列模板。我们重点分析了 200K 模板，以避免 50K 和 1M 模板中出现的过低或过高粗糙度，见图 2。此外，加入噪声本身就涵盖了高粗糙度的情况。对于最后一行的比特币时间序列，与其他算法相比，MinMax 和 LTTB 展现出更优的稳定性，而 M4算法最不稳定。在其上方的后续三行中，分别对应 ball、power 和 cinecg 数据集，MinMax 的稳定性始终超过其他算法，且通常具有显著优势。在考察 平均绝对误差 时，M4算法的表现经常最差，其次是 EveryNth 或 LTTB。在考察 峰值平均绝对误差 时，EveryNth 表现得最不稳定，其次是 M4算法，然后是 LTTB。值得注意的是，当仅关注峰值（峰值平均绝对误差）时，MinMax 与其他算法之间的差距更加明显。对于最后一列中的生成噪声，MinMax 再次成为最稳定的算法，而与之前的模板相比，LTTB 则最不稳定。另一个值得注意的

[5] 这个 GI F 展示了线宽对图像模板 gr 的影响 id.



图 6：对应用于不同时间序列模板的各类数据点选择算法的视觉稳定性进行评估。每列代表一个长度为 200K 的时间序列模板。两行分别展示了所考察降采样算法的平均绝对误差 和峰值平均绝对误差 分布，并以颜色区分。每个箱线图都涵盖多个偏移量（向左和向右），并包含从 200 到 2000、增量为 200 的n_{out} 个值。

观察是三种不同情况下的结果变化都很小。此外，该 GIF 通过展示增大 n_{out} 有助于提高稳定性这一现象，补充了我们的研究发现；这是一个合乎逻辑但颇为有趣的结果[6]。

MinMax 稳定性更高的一个合理解释是其具有较强的局部性，即数据点选择在多大程度上取决于各分箱自身的数值，而不受相邻分箱影响。局部性越强，稳定性越高，对齐问题的易感性也越低。在所有考察的算法中，MinMax 的局部性最高，因为它会为每个分箱选择最小值和最大值，且不受相邻分箱影响。相比之下，LTTB 表现出较弱的局部性：它以第一个数据点初始化顺序数据点选择过程，随后将前一个分箱中选定的数据点（以及下一个分箱的平均点）作为局部算法的输入。此外，局部算法会选择使三角形面积最大的点；如第 4.1.2 节所述，这使 LTTB 倾向于逐箱最小值与最大值交替，从而进一步加剧对齐问题在数据点选择过程中的传播。这种相互作用使 LTTB 更容易受到对齐问题影响，如图 6 中含有生成噪声的情况所示。尤其对于高噪声数据，LTTB 的不稳定性最为显著，因为可供选择的局部峰值很多。此外，EveryNth 算法更容易受到对齐影响，因为偏移量会直接影响所有选定数据点。M4 是 MinMax 与 EveryNth 的组合，其所选数据点中有一半容易受到对齐影响。在 M4 中，最小值和最大值始终位于第一个点和最后一个点之间（或恰好位于这两个点处），因此每个分箱中三条视觉插值线有两条会受到偏移量影响（只有最小值与最大值之间的插值基本不受影响）。因此，我们观察到，M4 大多数时候与 EveryNth 表现相当或略好，但无法达到 MinMax 所提供的稳定性。对于生成噪声的情况，这一点尤为明显，见图 6 以及预告图 1 中的示意图。

[6]有关视觉稳定性的更多详情（显示对于 MEAP 指标，至少纳入了 30% 的数据点）可在 GitHub 仓库中找到。

5 指南

根据评估结果，我们提出以下基于证据的指南，指导如何通过数据点选择实现大规模时间序列可视化：

1. **使用 MinMax 或 LTTB 进行数据高效聚合**：对于较小的n_{out}，MinMax 和 LTTB 已被证明是数据效率最高的聚合算法。然而，对于较大的n_{out}，MinMax 在视觉代表性方面往往优于 LTTB。

2. **使用 MinMax 以获得更优的视觉稳定性**：MinMax 始终表现出更优的视觉稳定性，显著优于其他算法。尽管 LTTB 在大多数情况下排名第二，但当峰值较多时，其稳定性较差，这是因为 LTTB 倾向于采用分箱极值交替。

3. **将像素级完美的 M4 视为一种可视化技术，而非数据点选择算法**：要实现像素级完美的 M4，需要使用混叠图形，并控制多种与可视化相关的参数，例如线宽、栅格化后端、x 范围分箱和画布宽度。

4. **若追求高数据效率，应考虑线宽**：增大线宽已被证明是提高数据效率的有效方法，但会牺牲粒度。我们发现，线宽、画布宽度与n_{out}之间存在明确关系，后者决定了性能曲线的拐点。

尽管 MinMax 似乎鲜少受到关注，例如，Gil 等人 [11]，Bae 等人 [3],和 Steinarsson [31]开展的研究均未考虑这一算法，但我们的结果仍表明，MinMax 是一种极具竞争力的降采样算法，且往往优于其他算法。只有 Jugel 等人 [16], 与我们的结果一致，将 MinMax 较高的数据效率作为其评估中的一项（次要）观察结果加以指出。因此，我们认为，未来研究应给予 MinMax 更多关注。

6 结论

本研究提出了一种基于大量指标的方法论，用于评估可视化中的数据点选择算法，并量化

视觉代表性以及一个称为“视觉稳定性”的新概念。我们的方法论采用多种时间序列数据集，每种数据集均有三种不同规模，涵盖广泛的时间序列属性和缩放级别。为衡量视觉代表性，我们使用了三种基于图像的指标，以评估可视化工具包参数的影响。我们引入 OR-conv 掩码，以便更有针对性地进行逐像素指标聚合，从而促进不同时间序列模板之间的一致比较。此外，我们主张使用选定数据点的数量作为更公平的数据效率衡量标准。视觉稳定性则通过两种数据空间指标进行衡量，从而克服在图像空间中评估时可能出现的栅格化对齐限制。

我们采用所提出的方法论评估了 EveryNth、MinMax、M4算法和 LTTB 算法，并考虑了各种图形绘制属性。在大多数情况下，就视觉代表性而言，LTTB 和/或 MinMax 表现更优。我们还探讨了线宽对数据效率的影响，并基于实证结果，利用线宽、画布宽度和保证交替次数 $\bar{}n_8{}^a$，推导出一个通用公式，用于确定性能曲线的拐点。此外，我们指出，在实践中实现像素级完美的 M4 并非易事，甚至可能无法实现。值得注意的是，我们首次观察到，在存在抗锯齿的情况下，M4 永远无法达到像素级完美。就视觉稳定性而言，MinMax 被证明是最稳定的算法，且往往显著领先。我们通过联系降采样算法的局部性，对这些结果作出了解释。基于这些发现，我们为采用数据点选择的大规模时间序列可视化制定了一套指南。总体而言，MinMax 算法凭借其出色的性能，理应得到更多关注，而大多数相关研究却未将其纳入考量。

所提出的方法论及获得的结果均已公开，可促进可复现性并推动该领域的进一步研究。本研究为获取新见解和评估降采样算法奠定了基础。

6.1 局限性

尽管本研究对用于可视化的时间序列数据点选择提供了诸多见解，但仍需承认其存在一些局限。首先，与其他相关研究 [11, 16, 27, 31], 一致，本研究将范围限定在没有间断的规则采样数据，因此可能无法充分代表真实世界时间序列数据。此外，本研究未开展用户研究；尽管各项指标在一定程度上能够衡量视觉代表性和视觉稳定性，但它们只是用户感知的替代指标。因此，未来研究可考虑开展用户研究，以确定所用指标的阈值：低于这些阈值时，用户感知不会有所改善。我们公开提供的代码库为此类研究奠定了坚实基础，因为所研究的工具包中有两个可集成到网页中，从而能够开展网络调查。最后，本研究仅考察了会影响视觉代表性的一组有限的可视化配置，例如线宽和工具包。由于线形及其他配置较少被更改，本研究未对其进行考察；不过，所提出的方法论和实验设计可用于在未来研究中考察这些因素。

作者贡献

JonasVanDerDonckt：概念化、方法论、软 件、验证、形式分析、数据整理、撰写——初稿、可视化。**Jeroen Van Der Donckt**：概念化、方法论、软件、验证、形式分析、数据整 理、撰写——初稿、可视化。**Michael Rademaker**：审阅与编辑。**Sofie Van Hoecke**：审阅与编辑。

补充材料

补充材料可在 GitHub 上获取：
https://github.com/predict-idlab/ts-datapoint-selection-vis。该代码仓库包含：(1) 四种已研究降采样算法的实现；(2) 用于复现实验的代码；(3) 包含所收集数据的 Parquet 文件；以及 (4) 所用数据的预处理结果，并附有该数据的下载地址。

致谢

作者谨向 Louise Van Calenbergh 致谢，感谢她为稿件进行校对。Jonas Van Der Donckt（1S56322N）获得佛兰德斯研究基金会（FWO）博士研究奖学金资助。本研究的部分工作是在 imec.AAA Context-aware health monitoring 项目范围内完成的。

参考文献

[1] R. Agrawal、A. Kadadi、X. Dai 和 F. Andres。《大数据可视化的挑战与机遇》。载于第七届计算智能与数字生态系统管理国际会议论文集，第169–173页，巴西卡拉瓜塔图巴，2015年10月。ACM。[2] W. Aigner、S. Miksch、W. M¨uller、H. Schumann 和 C. Tominski。《面向时间的数据可视化——系统化视角》。《计算机与图形学》，31(3):401–409，2007年6月。ZSCC: 0000450。[3] P. Bae、K.-W. Lim、W.-S. Jung 和 Y.-B. Ko。《面向网络可视化服务的 M4 实际实现》。《通信与网络期刊》，19(4):384–391，2017年。[4] J. A. Bednar 和 J. Signell。《大数据让常见绘图陷阱变得更糟》。github.com/holoviz/datashader/examples/user_guide/1_Plotting_Pitfalls.ipynb。[5] N. Bikakis。《大数据可视化工具》。《arXiv:1801.》*08336 [cs]*, 2018年2月。ZSCC: 0000075 arXiv: 1801.08336。[6] M. A. Breddels。《使用 vaex 对十亿个对象进行交互式（统计）可视化与探索》。《国际天文学联合会论文集》，12(S325):299–304，2016年10月。[7] H. A. Dau、A. Bagnall、K. Kamgar、C.-C. M. Yeh、Y. Zhu、S. Gharghabi、C. A. Ratanamahatana 和 E. Keogh。《UCR 时间序列档案库》。《IEEE/CAA 自动化学报》，6(6):1293–1305，2019年。[8] J. Davi Paganini。《数据库中的降采样：数据局部性如何改善数据分析》。www.timescale.com/blog，2023年2月。[9] D. H. Douglas 和 T. K. Peucker。《用于减少表示数字化线条或其夸张变形所需点数的算法》。《制图学：地理信息与地理可视化国际期刊》，10(2):112–122，1973年。[10] J. Franco、A. Garcia 和 A. Gil。《面向工业 4.0 数据可视化的多变量自适应降采样算法》。载于第*16*届工业与环境应用软计算模型国际会议（*SOCO 2021*），第588–597页。Springer，2022年。[11] A. Gil、M. Quartulli、I. G. Olaizola 和 B. Sierra。《利用统计方法从时间序列中智能选择数据》。《IEEE Access》，9:44390–44401，2021年。[12] E. Y. Gorodov 和 V. V. Gubarev。《大数据应用中的数据可视化方法分析综述》。《电气与计算机工程期刊》，2013:1–7，2013年。[13] J. M. Hellerstein、R. Avnur、A. Chou、C. Hidber、C. Olston、V. Raman、T. Roth 和 P. J. Haas。《交互式数据分析：Control 项目》。《计算机》，32(8):51–59，1999年。[14] Holoviz-community。《Datashader：快速、准确地渲染最大规模的数据》。https://github.com/holoviz/datashader。[15] Z. Jerzak、T. Heinze、M. Fehr、D. Gr¨ober、R. Hartung 和 N. Sto-janovic。《DEBS 2012 大挑战》。载于第六届 *ACM* 分布式事件系统国际会议论文集，第393–398页，2012年。[16] U. Jugel、Z. Jerzak、G. Hackenbroich 和 V. Markl。《M4：面向可视化的时间序列数据聚合》。《VLDB基金会论文集》，7(10):797–808，2014年6月。[17] U. Jugel、Z. Jerzak、G. Hackenbroich 和 V. Markl。《VDDA：关系数据库中可视化驱动的数据自动聚合》。《VLDB 期刊》，25:53–77，2016年2月。

[18] Kaggle 公司。《Kaggle 机器学习与数据科学现状 2022》。技术报告，2022年10月。[19] D. H. Kim、V. Setlur 和 M. Agrawala。《探索读者如何整合图表与图注：折线图案例研究》。载于*2021*年*CHI*计算机系统中的人为因素会议论文集，第1–11页，2021年。[20] P. Kottarathil。《比特币历史数据集》。kaggle.com/datasets/ btcinusd，2022年3月。[21] B. C. Kwon、J. Verma、P. J. Haas 和 C. Demiralp。《面向可扩展视觉分析的采样》。《IEEE计算机图形学与应用》，37(1):100–108，2017年1月。[22] W. J. Leler。《人类视觉、抗锯齿与廉价的4000行显示器》。《ACM SIGGRAPH计算机图形学》，14(3):308–313，1980年。[23] J. Lin、E. Keogh 和 S. Lonardi。《大型时间序列数据库中非平凡模式的可视化与发现》。《信息可视化》，4(2):61–82，2005年6月。ZSCC: 0000178。[24] C. Mutschler、H. Ziekow 和 Z. Jerzak。《DEBS 2013大奖赛》。载于第七届 *ACM* 分布式事件系统国际会议论文集，第289–294页，2013年。[25] J. Nilsson 和 T. Akenine-M¨oller。《理解 SSIM》。《arXiv 预印本 arXiv:2006.13846》，2020年。[26] B. Raskin 和 N. Aggarwal。《十亿数据点挑战：为高基数时间序列数据构建查询引擎》。uber.com/ billion-data-point-challenge，2018年12月。[27] K. Rong 和 P. Bailis。《ASAP：通过时间序列平滑确定注意力优先级》。《VLDB基金会论文集》，10(11):1358–1369，2017年8月。[28] U. Sara、M. Akter 和 M. S. Uddin。《通过特征相似性指数、SSIM、均方误差和峰值信噪比进行图像质量评估——一项比较研究》。《计算机与通信杂志》，7(3):8–18，2019年。[29] W. Shi 和 C. Cheung。《矢量概化折线简化算法的性能评估》。《制图学杂志》，43(1):27–44，2006年。[30] B. Shneiderman。《《眼睛说了算》：按数据类型划分的信息可视化任务分类法》。载于*1996*年 *IEEE*视觉语言研讨会论文集，第336–343页。IEEE，1996年。[31] S. Steinarsson。《用于视觉呈现的时间序列降采样》。硕士论文，冰岛大学，2013年。[32] J. Van Der Donckt、J. Van Der Donckt、E. Deprost 和 S. Van Hoecke。《Plotly-resampler：面向大型时间序列的有效视觉分析》。载于*2022*年*IEEE*可视化与可视分析会议（*VIS*），第21–25页。IEEE，2022年。[33] M. Visvalingam 和 J. D. Whyatt。《通过重复删除点实现折线概化》。《制图学杂志》，30(1):46–51，1993年。[34] J. Walker、R. Borgo 和 M. W. Jones。《TimeNotes：时间序列数据的有效图表可视化与交互技术研究》。《IEEE可视化与计算机图形学汇刊》，22(1):549–558，2016年1月。[35] Z. Wang 和 A. C. Bovik。《均方误差：取之还是弃之？重新审视信号保真度度量》。《IEEE信号处理杂志》，26(1):98–117，2009年。[36] G. Zhai 和 X. Min。《感知图像质量评估：综述》。《中国科学：信息科学》，63:1–52，2020年。