Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

Size Matters: Choosing the Most Informative Set of Window Lengths for Mining Patterns in Event Sequences

Jefrey Lijffijt · Panagiotis Papapetrou · Kai Puolamäki

Received: 3 April 2014 / Accepted: — / Published online: — © The Author(s) 2014

Abstract In order to find patterns in data, it is often necessary to aggregate or summarise data at a higher level of granularity. Selecting the appropriate granularity is a challenging task and often no principled solutions exist. This problem is particularly relevant in analysis of data with sequential structure. We consider this problem for a specific type of data, namely event sequences. We introduce the problem of finding the best set of window lengths for analysis of event sequences for algorithms with real-valued output. We present suitable criteria for choosing one or multiple window lengths and show that these naturally translate into a computational optimisation problem. We show that the problem is NP-hard in general, but that it can be approximated efficiently and even analytically in certain cases. We give examples of tasks that demonstrate the applicability of the problem and present extensive experiments on both synthetic data and real data from several domains. We find that the method works well in practice, and that the optimal sets of window lengths themselves can provide new insight into the data.

Keywords Event sequence · Pattern mining · Window length · Output-space clustering · Exploratory data analysis

J. Lijffijt (✉)
Department of Engineering Mathematics, University of Bristol
MVB Woodland Road, BS8 1UB, Bristol, United Kingdom &
Department of Information and Computer Science, Aalto University
E-mail: jefrey.lijffijt@bristol.ac.uk

P. Papapetrou
Department of Computer and Systems Sciences, Stockholm University
Forum 100, SE-164 40 Kista, Sweden

K. Puolamäki
Finnish Institute of Occupational Health
Topeliuksenkatu 41 a A, FI-00025 Helsinki, Finland

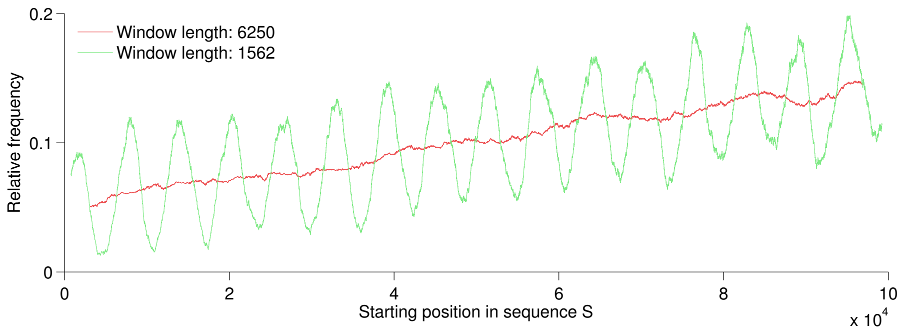

Fig. 1 Analysis of data at different levels of granularity may reveal different patterns in the data. In this example, visualisation of the relative frequency of an event—using sliding windows of different lengths—reveals different trends in the data. The generative process for this sequence is described in Section 6.2.

1 Introduction

Event sequences often contain continuous variability at different levels. In other words, their properties and characteristics change at different rates, concurrently. For example, the sales of a product may slowly become more frequent over a period of several weeks, but there may be interesting variation throughout a week at the same time. To provide an accurate and robust view of such multi-level structural behaviour, one needs to determine the appropriate levels of granularity for analysing the underlying sequence. This is especially relevant when using a sliding window.

Sliding windows are frequently used in several sequence analysis tasks, such as mining frequent episodes (Mannila et al 1997), finding biological or time series motifs (Chiu et al 2003; Das and Dai 2007), analysing electroencephalograms (EEGs) (Sörnmo and Laguna 2005), or in linguistic analysis of documents (Biber 1988). However, such methods are often parametrised by a user-defined window length and it can be unclear how to choose the most appropriate window length(s).

This problem can be solved by defining an appropriate objective function and using an optimisation algorithm to select the best window length. In some cases an appropriate cost function may be easy to specify, but that is not always so. Besides, using a single window length may leave out important information. In this paper, we introduce a framework and a generally applicable objective function that aims at finding a small set of window lengths that together provide as much information as possible about the underlying data, with respect to a quantity of interest.

Example. Consider as a statistic the frequency of an event in an event sequence. Such statistics may involve variation at different levels simultaneously. Figure 1 shows an example of the relative frequency of an event over time computed using sliding windows of lengths 1562 and 6250 (the generative process for this sequence is described in Section 6.2). We observe that each window describes a different view of the data: the longer window suggests a smoothly increasing frequency throughout the sequence, while the shorter window captures a periodic behaviour of the statistic.

Summary of Contributions. We introduce a novel framework and a generally applicable objective function to find the most informative set of window lengths for analysing event sequences with any algorithm that outputs real numbers. We prove that optimising the objective function is NP-hard in general, and show that we can approximate the solution efficiently using an existing algorithm with some modifications. We also prove that for certain simple statistics and data distributions, the optimisation problem can be solved exactly analytically. We show that these exact solutions are useful to compare empirical results with.

Furthermore, we give examples of tasks that demonstrate the applicability of the problem to different domains, and study optimal window lengths for both synthetic and real data. We present experiments on data from several application domains: text books, DNA sequences, and sensor measurements. We find that the optimisation algorithm works sufficiently well, that the optimal window lengths for finding patterns in various types of data vary significantly, and that the optimal window lengths themselves can also provide useful information about the data.

This manuscript is an extended version of Lijffijt et al (2012). the main differences are the following. The problem setting is more general and we provide more justification for why the proposed criteria for selecting window lengths are useful, i.e., that it enables a user to infer the values of the relevant statistic at all window lengths as well as possible. We propose a different optimisation algorithm (see above). We prove that the problem can be solved analytically in certain cases. All experiments have been redesigned, are more extensive and now include significance testing. We study an additional type of data: real-valued sensor measurements.

Outline. The remainder of this paper is outlined as follows. The related work is discussed in Section 2, and the formal problem setting is presented in Section 3. The framework and the generally applicable objective function are introduced in Section 4, the experiments are reviewed in Section 7, and the paper is concluded in Section 8.

2 Related Work

Sliding windows have been used in many application domains that involve sequences (discrete or continuous). However, window lengths are chosen either empirically or they are optimised for the task at hand. To the best of our knowledge, no earlier work has proposed a principled method for choosing a set of window lengths that optimally summarise the data for a given statistic and data mining task. To provide a context for this work, we briefly review the main approaches from several domains.

String and Text Mining. Sliding windows are used frequently in string mining. Indexing methods for string matching based on *n-grams* (Li et al 2007a), i.e., subsequences of length *n*, employ sliding windows of fixed or variable length to create dictionaries and speed-up approximate search for strings in large collections of texts. Determining the appropriate window length is a challenge, as small window lengths result in higher recall but large index structures.

In text mining, looking at different linguistic dimensions of text results in extracting different views of the underlying text structure (Biber 1988). One way to quantify these views is by using sliding windows. Recently, an interactive text analysis tool[1] has been developed for exploring the effect of window length on three commonly studied linguistic measures: type-token ratio, proportion of hapax legomena, and average word length. However, the window length is user-defined.

Bioinformatics. Several sliding window approaches have been proposed for analysing large genomes and genetic associations. Existing methods can be categorised into two groups: fixed-length vs. variable-length sliding windows (Bourgain et al 2000; Toivonen et al 2000; Mathias et al 2006; Li et al 2007b; Papapetrou et al 2012). For the case of fixed-length windows it is hard to determine the optimal window length per task while variable-length windows provide higher flexibility.

A variable window length framework for genetic association analysis employs principal component analysis to find the optimum window length (Tang et al 2009). Sliding windows have also been used for searching large biological sequences for tandem repeats (Benson 1999), motifs (Das and Dai 2007), and poly-regions (Papapetrou et al 2006). In all cases it is assumed that there exists only one optimum length and the solution is limited to the task of genetic association analysis.

Stream Mining. A typical task in stream mining is to detect and monitor frequent itemsets in an evolving stream, counted over sliding windows. We present a brief survey of the use of sliding windows in stream mining, though the overall setting is very different from the problem studied in this paper and a setup requiring online learning is not considered here.

In the case of the fixed-length window model the length of the window is set at the beginning, and the data mining task is to discover recent trends in the data contained in the window (Demaine et al 2002; Golab et al 2003; Karp et al 2003; Jin et al 2008). In the time-fading model (Lin et al 2005) the full stream is taken into account in order to compute itemset frequencies but the frequencies are weighted by recency, i.e., recent transactions have a higher weight as compared to older transactions.

The tilted-time window (Giannella et al 2003) can be seen as a combination of different scales reflecting the alteration of the time scales of the windows over time. In the landmark model, particular time periods are fixed while the landmark designates the start of the system until the current time (Jin and Agrawal 2005; Karp et al 2003). Calders et al (2008) introduced a frequency measure based on a variable window length by defining the frequency as the maximal frequency over all windows until the latest event. Variants of these methods have been proposed for specific objectives.

Time Series. Enumerating frequently occurring patterns is a common problem in time series. Such patterns are called *motifs* due to the analogy to their discrete counterparts in computational biology. Efficient motif discovery algorithms have been proposed, based on sliding windows, for summarising and visualising massive time series databases (Chiu et al 2003; Mueen et al 2009).

Papadimitriou and Yu (2006) proposed a method for discovering locally optimal patterns in time series at multiple scales along with a criterion for choosing the best window lengths. However, this is a local heuristic and applies only to continuous data. Also related is the problem of scale-space decomposition of time-series (Vespier

[1] `http://www.uta.fi/sis/tauchi/virg/projects/dammoc/tve.html`

et al 2012), which aims at defining several frequency bands that correspond to the components of a signal.

Fourier and Wavelet transforms (Sörnmo and Laguna 2005) are used pervasively to analyse periodicity in time-series. Although it may be possible to apply such methods for certain statistics, these methods are not generally applicable to the problem setting considered here. Fourier and Wavelet transforms require numeric input, while we focus on selecting window lengths for finding local patterns in event sequences where the event labels are not restricted to numbers.

Finally, several algorithms have been proposed recently for efficient discovery of motifs of variable length (Li et al 2012; Yingchareonthawornchai et al 2013; Mueen 2013). The key challenge in motif discovery is how to enumerate motifs of variable length efficiently without performing exhaustive search of all possible lengths. Although granularity selection is a problem in motif discovery, the general problem studied in this paper deviates from the objective of motif discovery. We do not aim at finding repeated patterns of variable length in time series, but rather for the most informative set of window lengths for analysing event sequences.

3 Problem Setting

3.1 Preliminaries

Given a set of event labels L, an *event sequence* S of length n is defined as $S = (s_1, \ldots, s_n)$, where $s_i \in L$, for all $i \in \{1, \ldots, n\}$. We denote the *subsequence* of S starting at position i with length ω as $S_{i,\omega} = (s_i, \ldots, s_{i+\omega-1})$. We use the term *window length* to refer to the length of the subsequences, ω. Thus, analysis of the data using a sliding window of length ω and step size 1 means that we look at all subsequences of S of length ω. We assume that an analyst using our method is interested in analysing data using an algorithm takes as input a subsequence $S_{i,\omega}$ and outputs a real number $f(S_{i,\omega})$. We refer to this output $f(S_{i,\omega}) : L^{\omega} \to \mathbb{R}$ as the *statistic* (of interest). Examples of possible statistics f are given in Section 3.3, but in principle f can be any function or algorithm.

3.2 Problem Definition

Our aim is to find a set θ of k granularity levels (window lengths) that are most informative with respect to the statistic f and the event sequence S. An intuitive way to express the informativeness is to measure how well we can predict values of the statistic f at other granularity levels. Hence, the problem can be translated into finding a set of k window lengths that allows an analyst to predict f as well as possible for all window lengths that we are interested in.

To this end, we assume that the end-user is able to specify a set Ω that contains all potentially interesting window lengths. We argue that a set of window lengths θ is *most informative* if it enables an end-user to infer the behaviour of the data (i.e., the statistic f) at all window lengths in Ω as well as possible. This expression allows us to formulate the problem of finding the most informative set of window lengths as a regression problem.

Depending on the task at hand, one may consider different regression/objective functions. In general the objective function depends both on a set θ of k window lengths and the parameters of a regression function, such that, at each position i in S, if we are given the value of f at those k window lengths, we can accurately estimate f for all other window lengths at position i. In this paper, we consider an objective function that corresponds to solving k-medoids clustering in the output space.

More formally, let $\Omega = \{\omega_1, \ldots, \omega_m\}$ be the set of m window lengths that we consider potentially interesting for analysing the structure of S. We denote the subset of k window lengths that we select, i.e., the interesting parameters of the optimisation problem, as $\theta = \{\theta_1, \ldots, \theta_k\} \subseteq \Omega$. Although the predictive model may contain other parameters, we are not interested in their values; our aim is that those parameters should be easy to guess approximately for a user. Thus, we are interested only in the corresponding window lengths.

Let $\bar{f}(S, \theta, i) = \{f(S_{i,\theta_1}), \ldots, f(S_{i,\theta_k})\}$ be the set of real numbers that corresponds to the values of f at position $i \in \{1, \ldots, n^*\}$ for the k window lengths in θ. For simplicity we define $n^* = n + 1 - \max_{\omega \in \Omega}(\omega)$, which ensures that we do not consider subsequences that are partly unknown. The following definition allows us to state the optimisation problem more succinctly.

Definition 1 (Reconstruction Function) A *reconstruction function* $g(\bar{f}(S, \theta, i), \omega) : \mathbb{R}^k \times 1 \to \mathbb{R}$ is a function that, given the set of values $\bar{f}(S, \theta, i)$ and a window length ω, estimates the value of f for window length ω; in other words, g is an estimator for $f(S_{i,\omega})$.

Figure 2 provides an illustration of the mechanism of a reconstruction function. The optimisation problem that corresponds to the granularity selection task is then the following:

Problem 1 (k-Windows Problem) Given an event sequence S, a statistic f, and a set of window lengths Ω, find a set of k window lengths $\theta = \{\theta_1, \ldots, \theta_k\} \subseteq \Omega$ and a reconstruction function g that minimise

$$\frac{1}{n^*} \sum_{i=1}^{n^*} \sum_{\omega \in \Omega} \left(f(S_\omega(i)) - g(\bar{f}(S, \theta, i), \omega)\right)^2.$$

The reconstruction function g can in principle be any regression function. However, this would lead to a practically impossible optimisation task, as it is infeasible to explore the space of all possible regression models. Hence, we propose that, depending on the task at hand, we can restrict the set of possible models to obtain a tractable optimisation problem. For example, we can restrict g to the class of nearest neighbour regressors (see Section 4.1), in which case the optimisation problem is equivalent to the k-medoids clustering problem.

The idea here is that any additional parameters used by g, i.e., those that are not in $\bar{f}(S, \theta, i)$, are kept implicit and not shown to the user. Neither is g itself considered to be interesting. Hence, we should restrict g to regression functions that are easy to comprehend by end-users. In the remainder of this paper we study only the nearest neighbour regressor, which is introduced in Section 4.1.

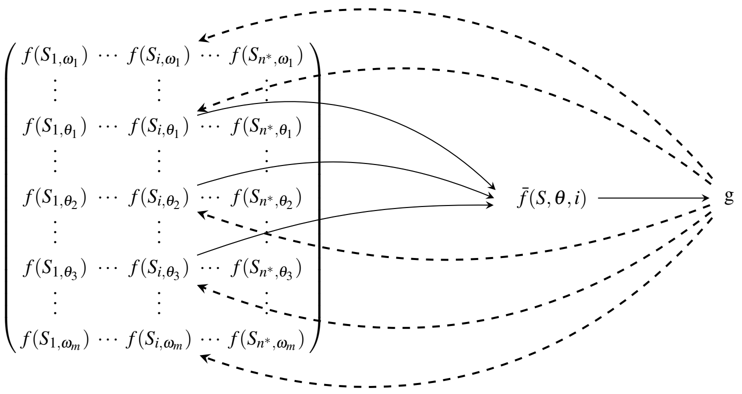
$f(S_{1,\omega_1})$ ⋯ $f(S_{i,\omega_1})$ ⋯ $f(S_{n^*,\omega_1})$
$f(S_{1,\theta_1})$ ⋯ $f(S_{i,\theta_1})$ ⋯ $f(S_{n^*,\theta_1})$
$f(S_{1,\theta_2})$ ⋯ $f(S_{i,\theta_2})$ ⋯ $f(S_{n^*,\theta_2})$
$f(S_{1,\theta_3})$ ⋯ $f(S_{i,\theta_3})$ ⋯ $f(S_{n^*,\theta_3})$
$f(S_{1,\omega_m})$ ⋯ $f(S_{i,\omega_m})$ ⋯ $f(S_{n^*,\omega_m})$
$\bar{f}(S,\theta,i)$
g

Fig. 2 Illustration of the reconstruction function g. Each row corresponds to a window length in Ω and each column to a position in the event sequence S. Function g estimates the value of the statistic f for all window lengths at position i in S, based on the values of f for a small set window lengths $\{\theta_1,\ldots,\theta_k\}$, in this case $k = 3$.

3.3 Examples of statistic f

We give three examples of statistic f, each of which is also used in the experimental evaluation. The first definition of f is the relative frequency of an event $q \in A$:

$$f(S_{i,\omega}) = \frac{\#\, of\ occurrences\ of\ q\ in\ S_{i,\omega}}{\omega}. \tag{1}$$

Note that by definition $\omega = |S_{i,\omega}|$. Alternatively, f may be defined as the *hapax legomenon* ratio of a sequence, i.e.,

$$f(S_{i,\omega}) = \frac{\#\, of\ events\ occurring\ exactly\ once\ in\ S_{i,\omega}}{\omega}. \tag{2}$$

For real valued data the mean can be used as a statistic, in which case f is defined by

$$f(S_{i,\omega}) = \frac{1}{\omega} \sum_{j=i}^{i+\omega-1} s_j. \tag{3}$$

The utility of these three definitions in practice is shown in Sections 6 and 7.

4 Solving k-Windows Using k-Medoids

In this section, we introduce our approach to selecting the k most informative window lengths. To do so, we choose as reconstruction function g a partition-based nearest neighbour regressor. This restricted problem setting is defined in Section 4.1. In Section 4.2 an auxiliary data structure, called the *Window-Trace matrix*, is introduced, and the optimisation algorithm is described in Section 4.3.

4.1 Partition-based Regression

To make the problem setting tractable, we restrict the reconstruction function g to the following class of regression functions.

Definition 2 (*k*-Partition NN Regressor) A *k-partition nearest neighbour regressor* is a *reconstruction function* $g(\bar{f}(S,\theta,i),\omega) : \mathbb{R}^k \times 1 \to \mathbb{R}$ that implicitly contains a partitioning of the set Ω into k non-overlapping clusters. Each cluster is represented by a single window length, i.e., $g(\bar{f}(S,\theta,i),\omega) = f(S_{i,\theta_j})$, where $\theta_j \in \theta$ is the representative window length for the cluster where ω belongs.

In the remainder of this paper, we study we restrict to studying the nearest neighbour regressor g. Problem 1 then translates to the problem of partitioning the set of all window sizes Ω into k clusters and selecting for each cluster one representative window size, such that the expectation of the squared error is minimised. This problem is equivalent to the k-medoids clustering problem.

Since the statistic f is unconstrained and the k-medoids clustering problem is NP-Hard (Aloise et al 2009), this optimisation problem is also NP-Hard, although this may not be true for all statistics f. Several optimisation algorithms to obtain good approximations; the algorithm that we propose to be appropriate in this setting is described in Section 4.3.

4.2 The Window-Trace Matrix

To solve Problem 1 we use an auxiliary matrix, called the *Window-Trace (W-T) matrix*. This matrix stores the values of statistic f for a set of indices in I and for all window lengths in Ω. More specifically, let S be the input sequence and f the statistic at hand. Then the W-T matrix $\mathscr{T}$ contains all values of $f(S_{i,\omega})$ for all window lengths $\omega \in \Omega$ and all indices $i \in I$. $\mathscr{T}$ is given by

$$\mathscr{T}_{ji} = f(S_{i,\omega_j}). \tag{4}$$

The most accurate representation is obtained by choosing $I = \{1,\ldots,n^*\}$. However, for the reasons of computational efficiency we select the set of indices I by sampling N indices from $\{1,\ldots,n^*\}$ uniformly in random and without replacement, where N is a given parameter. Furthermore, we use $\mathscr{T}_{j*}$ to denote the row of $\mathscr{T}$ corresponding to window length ω_j.

4.3 Optimization Algorithm

To solve the optimisation problem given in Section 4.1, we use the `Clustering LARge Applications` (`Clara`) algorithm (Kaufman and Rousseeuw 1990), which is a well-known algorithm to efficiently solve the k-medoids problem. However, we use a small improvement to increase the quality of the solution, which is described in the remainder of this section.

Algorithm 1 Clara++($\mathscr{T}$, k, r, s)

```
$\Theta_1 = \text{uniform}([1,\dots,n])$ {Pick a number between 1 and $n$ uniformly at random, $n$ is the number of rows in $\mathscr{T}$}
for i = 2 to $k$ do
  $\Theta_i = \text{rand}([1,\dots,n])$ {Pick a number between 1 and $n$ at random with probability proportional to the distance to the closest medoid in $\Theta$}
end for
$cost^* = \infty$
for i = 1 to $r$ do
  $S = \Theta \cup \text{uniform}(\{1,\dots,n\} \setminus \Theta, s-k)$ {Assign $S$ a set that contains $\Theta$, the best set of medoids currently known, and pick $s-k$ other row indices uniformly at random}
  $\mathscr{T}^S = \begin{bmatrix} \mathscr{T}_{S_1,*} \\ \vdots \\ \mathscr{T}_{S_s,*} \end{bmatrix}$ {Select the $s$ rows of $\mathscr{T}$ whose index is in $S$}
  $\Theta = \text{PAM++}(\mathscr{T}^S, k)$ {Compute PAM++ solution on sample}
  $cost = \text{computeClusteringCost}(\mathscr{T}, \Theta)$ {Compute cost for full matrix}
  if $cost < cost^*$ then
    $\Theta^* = \Theta$
    $cost^* = cost$
  end if
end for
return $\Theta^*$
```

The general strategy of `Clara` is to repeatedly take a small data sample and solve the clustering problem on the sample using the `Partitioning Around Medoids (PAM)` subroutine. The data sample always contains the current best medoids, and these are updated whenever a solution is found that has lower error on the full data. Arthur and Vassilvitskii (2007) study the effects of seeding—the process of choosing the initial representatives for each cluster—for the k-`means` algorithm, and introduce a simple method of 'careful' seeding that leads to an approximation ratio on the solution. Their improved algorithm is known as k-`means++`.

Although there have been studies on the effects of seeding for `Clara` algorithm, e.g., by Pakhira (2008), these are different from the change proposed here. Specifically, the change that we make to the original `Clara` algorithm is both in the initial seeding and in the seeding in the `PAM` subroutine that produces a clustering on a subset of the data points. The approximation ratio of (Arthur and Vassilvitskii 2007) holds also for the k-medoids method studied here, since the k-`means++` seeding method gives a proper k-medoids solution, the bound is without further optimisation, and the cost can only decrease further during the remaining steps of `Clara`. We expect this 'careful' seeding to substantially increase the result quality.

We name the improved variants `Clara++` and `PAM++`, respectively. Pseudocode for the methods is given in Algorithms 1 and 2. The parameters r and s are related to a trade-off between quality and time complexity, they are the number of repetitions and number of samples included in the `PAM` subroutine, respectively. In the original `Clara` algorithm these are not considered to be parameters and have default values of $r = 5$ and $s = 40 + 2k$ (Kaufman and Rousseeuw 1990). However, we will study the effects of increasing the default values in Section 6.

Algorithm 2 PAM++($\mathscr{T}$, k)

```
$\Theta_1 = \text{uniform}([1,\ldots,n])$ {Pick a number between 1 and $n$ uniformly at random, $n$ is the number of rows in $\mathscr{T}$}
for i = 2 to $k$ do
    $\Theta_i = \text{rand}([1,\ldots,n])$ {Pick a number between 1 and $n$ at random with probability proportional to the distance to the closest medoid in $\Theta$}
end for
$\Theta_{old} = \{\}$
while $\Theta_{old} \neq \Theta$ do
    $\Theta_{old} = \Theta$
    for $i$ = 1 to $n$ do
        $L_i = \arg\min_{j\in[1,\ldots,k]}(\|\mathscr{T}_{i*} - \mathscr{T}_{\Theta_j *}\|^2)$ {Label each point with nearest medoid}
    end for
    for i = 1 to $k$ do
        $C = \{x \mid L_x = i\}$ {Find the set of points in cluster $i$}
        $\Theta_i = \arg\min_{x\in C}\sum_{y\in C}\|T_{x*} - T_{y*}\|^2$ {Pick best medoid for cluster $i$}
    end for
end while
return $\Theta$
```

A Matlab implementation of the method and scripts for reproducing all experiments can be found on the website of the first author[2].

Computational Complexity. Let N be the number of columns of $\mathscr{T}$, i.e., the number of samples, and let m be the number of rows of $\mathscr{T}$: $m = |\Omega|$. The memory required to store the Window-Trace matrix $\mathscr{T}$ is $\mathscr{O}(m \cdot N)$ and if we assume that the complexity of computing the statistic $f(S_{i,\omega})$ is constant, then the computational complexity to create the W-T matrix is also $\mathscr{O}(m \cdot N)$.

`Clara++` consists of the initial selection of k medoids and then executing the `PAM++` subroutine r times, each on a data sample of size s, plus computing the cost of the clustering on each iteration. The initialisation of the k medoids has a computational complexity of $\mathscr{O}(k \cdot m \cdot N)$, because we have to compute the distance to all other points for each medoid. Let t denote the number of iterations required for convergence of the `PAM++` subroutine. Since computing the full distance matrix takes $\mathscr{O}(s^2 \cdot N)$ steps, the computational complexity of `PAM++` is $\mathscr{O}(s^2 \cdot N + t \cdot k \cdot s + t \cdot s^2)$ and since by definition $k < s$ this simplifies to $\mathscr{O}(s^2 \cdot (N + t))$. Also, we know that computing the cost of a clustering has complexity $\mathscr{O}(k \cdot m \cdot N)$, thus we find that the total computational cost of `Clara++` is $\mathscr{O}(r \cdot s^2 \cdot (N + t) + r \cdot k \cdot m \cdot N)$. That is, the cost is linear in the number of window lengths m and the number of data samples N and the number of repetitions r, but quadratic in s, the number of samples considered in an iteration of `PAM++`.

Notice that, given the set of window lengths to consider, Ω, the parameters of the optimisation algorithm, N, r, and s, determine the computational cost. Hence, one can choose freely choose an appropriate trade-off between solution quality and speed. In practice, solutions can be computed quickly, for example, computing one of the solutions for the largest data set studied in this paper (Section 7.3) takes 80 seconds using a straightforward Matlab implementation running on a single processor core (Intel Core i5 2.4 GHz notebook processor).

[2] Currently `http://users.ics.aalto.fi/lijffijt`

5 Analytical Solutions

For certain statistics and data distributions, it is possible to derive the solution, or at least the function for the distance between two window lengths, exactly. In this section, we present an analytical solution for the case where the statistic is the frequency of an event and the event sequence comes from a Bernoulli process. From the result it follows that for a Bernoulli process, the set of window lengths (i.e., the clustering) is independent of the frequency of an event.

Preliminaries. Let $(X_1, \ldots, X_n)$ be a sequence of Bernoulli random variables with common parameter p, i.e., $X_i \in \{0,1\}, \Pr(\{X_i = 1\}) = p$, for all $i \in \{1, \ldots, n\}$. The random variables could, for example, denote the occurrences of an event. Similar to the notation for event sequences, we use $X_{i,\omega}$ to denote the subsequence of length ω starting at position i, $(X_i, \ldots, X_{i+\omega-1})$. Let the statistic f be the relative frequency of ones:

$$f(X_{i,\omega}) = \frac{1}{\omega} \sum_{j=i}^{i+\omega-1} X_j. \tag{5}$$

The selection of an optimal set of window lengths is based on the squared error between predictions made using those window lengths (Problem 1). Under the constraint of using a k-partition nearest neighbour regressor, the predictions correspond to the value of the nearest window length (Section 4.1). Thus, to select the optimal window lengths, we have to compute the distance (squared error) between all pairs of window lengths. We find that the distance between window lengths is as follows.

Theorem 1 *For the statistic and generative process described above, the expected distance between two window lengths γ and ω, with $\gamma < \omega$, is*

$$\mathrm{E}\left[d(\omega,\gamma)\right] = \frac{\omega - \gamma}{\omega\gamma} p(1-p).$$

Proof See Appendix A.

We observe that there is no interaction between the window lengths γ, ω and the event probability p, which implies that all distances relative to each other are independent of p. Thus, for this specific statistic and data distribution, the optimal window lengths are unaffected by the event frequency, and depend only on the set of window lengths Ω.

Optimal Window Lengths for a Bernoulli Process. To investigate what optimal solutions would look like for a Bernoulli process, we have conducted the following experiment. Using the `PAM++` algorithm, we have computed optimal sets of window lengths for $k = |\theta| = 3, 4, 5, 6$, and window lengths from 1 to $m = 1024, 4096, 16384, 65536$, using the distance function given above.

The result is visualised in Figure 3. Due to the discreteness of the optimisation problem, and the fact that we do not solve the optimisation problem exactly, there are some minor variations, but the overall trend is very clear: the window lengths in each set follow an exponential pattern.

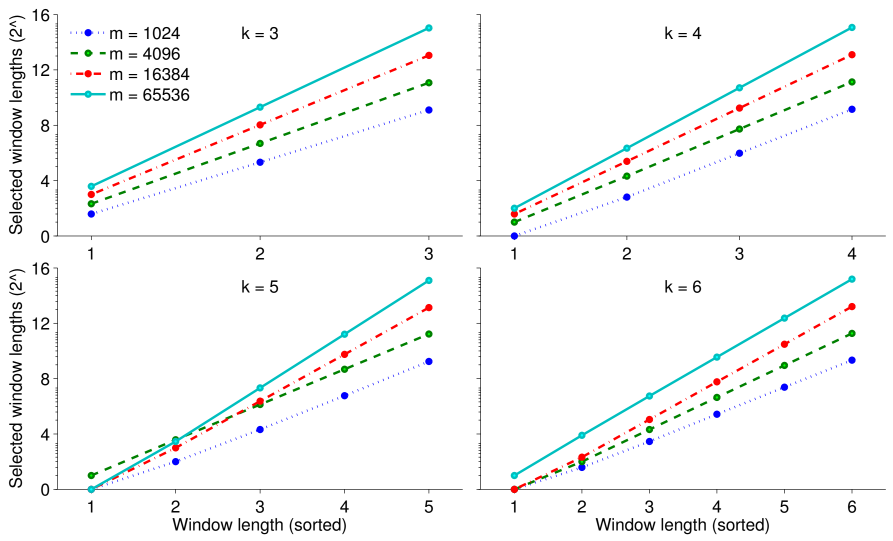

Fig. 3 Optimal sets of window lengths for analysis of data arising from a Bernoulli process, for various number of optimal window lengths *k* (different figures) and various maximum window lengths (indicated by color). Each dot represents an optimal window length and the dotted lines connect the window lengths from each set. We observe that, although there are small deviations, the optimal window lengths in each set grow exponentially.

6 Evaluation on Synthetic Data

Although we have analytically derived what to expect regarding optimal sets of window lengths for two basic types of random processes (see Section 5), we do not know to what extent there is variation in the solution given by the `Clara++` algorithm, which is important in determining the significance of a result. To provide a baseline for the results in Section 7, we have designed four experiments based on randomly generated data, where we know precisely what the properties of the data are.

6.1 Bernoulli Process with Fixed Rate

We are interested in the variation of the set of window lengths given by `Clara++`. We use Algorithm 3 to generate random data from a Bernoulli process with fixed rate, given parameters n and p, which are the length of the sequence and the probability of the event occurring at any position, respectively.

Algorithm 3 Simulate a fixed-rate Bernoulli process SIM1(n, p)

```
for $i = 1$ to $n$ do
    $S(i) = Bernoulli(p)$
end for
```

Experiment 1. Since `Clara++` is non-deterministic, the output may vary, even with the same input sequence. In the first experiment, we tested the stability of the solution in terms of the optimal window lengths, using a single sequence generated by Algorithm 3 with parameters $n = 1{,}999$ and $p = 0.1$.

We investigated how stable the result of the algorithm is, while varying the number of repetitions from 10 to 80 (doubling the value each time) and the number of samples from 40 to 320 (also by doubling the value each time). We varied the number of clusters from 1 to 4 and we used window lengths from 1 to 1,000. As the statistic we used the relative frequency of the event (Equation 1). We repeated the experiment for each setting 100 times. For comparison, we also tested the variability of the `PAM++` algorithm.

The results are presented in Figure 4. We observe that the variation with the default parameter settings is quite large. For example, the results for $k = 4$ in the top left figure show that the smallest window length is sometimes larger than the second largest window length in another run. We see also that the variation is greatly reduced when increasing the number of repetitions and/or the number of samples.

In the bottom right figure, we find that the set of window lengths is quite stable when we set the number of repetitions (r) and the number of samples (s) both to 80. As shown in Section 4.3, the computational complexity is linear in the number of repetitions (and independent of the size of the data because we use sampling), so typically it will be possible to use more repetitions and samples to improve the certainty of obtaining a close to optimal result.

Experiment 2. Several data sets, even if they are from the same generative process, may give quite different results. Thus, secondly, we tested the stability of the solutions given by `Clara++` for different data sets that have the same properties. We generated 1 data set with parameters $n = 1{,}999$, $p = 0.1$, and then produced 100 versions by randomly permuting the indices of the sequence. We tested the optimal window lengths for $k = 1-4$ on each data set. We used 80 repetitions and 80 samples as parameters for `Clara++`, because we found in the previous experiment that these are good choices, and the other parameters were kept the same as in the previous experiment.

The results are presented in Figure 5. We observe that there is much more variation than in the previous experiment, which can be explained by the fact that the input sequences are slightly different in each repetition. The observed variance can be used in future experiments to draw conclusions with respect to the significance of differences in sets of window lengths obtained for various events or data sets.

6.2 Bernoulli Process with Variable Rate

In the previous experiments, the frequency of the event is fixed over time, which leads to the sequence having structure only on a single scale. To test the ability of our method for finding the true underlying scale at which the data is structured, we designed an algorithm to simulate a Bernoulli process with variable rate.

The full process is described in Algorithm 4. The first component of the variable rate is based on a slow increase of the event frequency over time, which ranges from

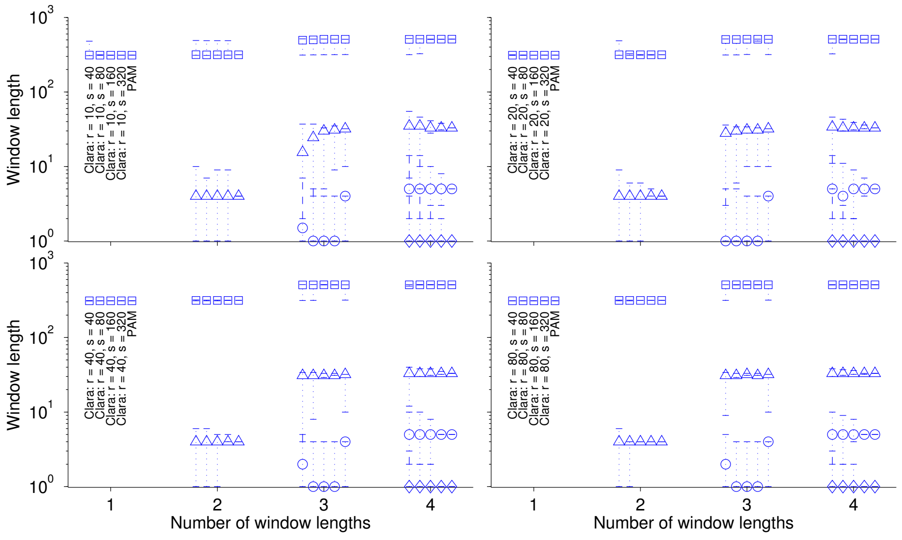

Fig. 4 Stability of the set of window lengths from `Clara++` for varying number of repetitions (one value per figure) and number of samples (adjacent bars in each figure). Squares, triangles, circles and diamonds represent the medians for various window lengths, the dotted lines represent 90 % confidence intervals and dashed lines denote that the confidence intervals for the window lengths are overlapping. For comparison, the variability for the `PAM++` algorithm is also shown for each number of window lengths. We observe that increasing the number of repetitions and the number of samples both have a considerable positive effect in reducing the variability of the result.

Algorithm 4 Simulate a variable-rate Bernoulli process SIM2(n, p, c)

```
for $i = 1$ to $n$ do
    $t_1 = 0.5 + (i-1)/(n-1)$; // Multiplier for scale 1: [0.5–1.5]
    $t_2 = 0.5 \cdot \sin(c \cdot 2 \cdot \pi \cdot (i-1)/(n-1))$; // Multiplier for scale 2: [−0.5–0.5]
    $S(i) = Bernoulli(p \cdot (t_1 + t_2))$
end for
```

$0.5 \cdot p$ at the start to $1.5 \cdot p$ at the end of the sequence S. The second component consists of the event frequency going up and down rhythmically, based on a sine wave with peak amplitude 0.5 and mean 0. Finally, both components are added together to give the variable event frequency, multiplied by the parameter p. The extra parameter, c, decides the periodicity of the sine wave, hence the second scale. We have generated a sequence with parameters $n = 100{,}000$, $p = 0.1$ and $c = 16$. The sequence has 10,009 events and has also been used to generate Figure 1.

Experiment 3. As discussed in Section 4.2, we try to estimate the optimal set of window lengths for a sequence using a W-T matrix based on samples from the data. We investigated empirically how many samples $\mathscr{T}$ should be based on to obtain a solution close to the solution that was obtained on the full matrix, i.e., the matrix $\mathscr{T}$ that covers the whole input sequence. We have varied the number of samples from 1 to 16,384 using powers of 2 and computed the solution 100 times for each sample size to assess the variance. We have used window lengths from 1 up to $\lfloor n/c \rfloor = 6{,}250$ (which is the scale of the second component in the data) and $k = 3$.

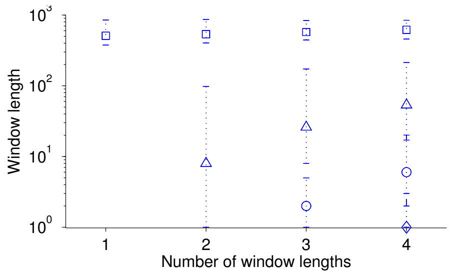

Fig. 5 Stability of the set of window lengths from `Clara++` over 100 data sets with the same properties. Squares, triangles, circles and diamonds represent the medians for various window lengths, the dotted lines represent 90 % confidence intervals and dashed lines denote that the confidence intervals for the window lengths are overlapping. The variation is much greater than in Figure 4, indicating that the variability in the results due the non-deterministic nature of the optimisation algorithm is much smaller than the variability over data sets with the same properties.

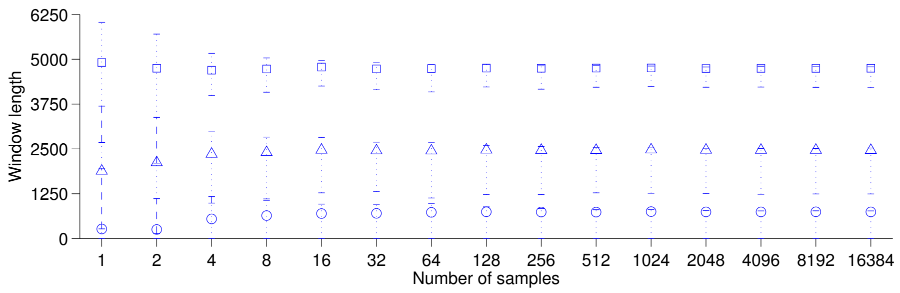

Fig. 6 Stability of window lengths from `Clara++` on a sequence obtained from simulating a Bernoulli process with rate that varies over time, using various numbers of samples to construct the Window-Trace matrix $\mathscr{T}$. Squares, triangles and circles represent the medians for various window lengths, the dotted lines represent 90 % confidence intervals and dashed lines denote that the confidence intervals for the window lengths are overlapping. Surprisingly, the variability in the solutions does not decrease significantly when increasing the number of samples beyond 32.

Figure 6 illustrates the results. We observe that the solutions are remarkably robust: the solutions using only 8 samples are already quite accurate approximations and from 32 samples and up, the solutions are practically equivalent. Thus, we can conclude that for simple data sets like this, a Window-Trace matrix based on 32 positions in S is sufficient.

Experiment 4. Finally, we tested if we can retrieve the two scales that are present in the synthetic sequence. To prevent making it too easy for the algorithm, we use window lengths from 1 to 20,000 and the Window-Trace matrix $\mathscr{T}$ is based on 1,000 indices in S. In a typical setting, we do not know how many scales a data set has. It is useful to note that a higher k always provides more information, thus choosing k too high is better than too low. For exploratory purposes, we use $k = 3$. Figure 7 illustrates the results. We find that the variable trend in the data can be identified well.

As a sanity check, we have computed the optimal solution for a data set generated with Algorithm 3 with an equal number of events. The prediction error on both data sets with both solutions is given in Table 1. We observe that the solutions are clearly

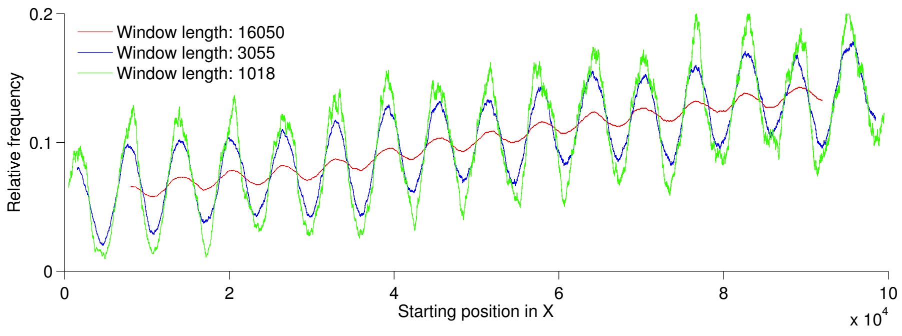

Fig. 7 The representation of the data based on the solution for $k = 3$ on a sequence obtained from simulating a Bernoulli process with rate that varies over time. The variable trend in the data is clearly shown by the shorter window lengths, while the longest window length reveals the slow trend.

Table 1 A cross-comparison of the total prediction error for two comparable synthetic data sets with the optimal solution for either sequence. The solutions are clearly different from each other and the prediction error differs by a factor of two or more.

Data sequence	Fixed-rate optimal solution (16148, 9058, 3044)	Variable-rate optimal solution (16050, 3055, 1018)
Fixed rate	$\mathbf{0.71 \cdot 10^6}$	$1.91 \cdot 10^6$
Variable rate	$0.22 \cdot 10^6$	$\mathbf{0.11 \cdot 10^6}$

specific to the data, and that in both cases optimising the set of window lengths for the data specifically leads to only half as much error. We also observe that the data from the time-varying process is much more predictable, the error is almost ten-fold lower. This is expected, as the data from the fixed rate process is maximally random.

Notice that our objective is to find a set of window lengths that is most informative with respect to the chosen statistic and the event sequence. The method is not designed to find subtle differences in structure that have only a small effect of the informativeness of a set of window lengths. Hence, we do not expect, e.g., that the window lengths found here would be sensitive to small variations in periodicity of the signal or that would be efficient in distinguishing components of the signal with approximately similar frequencies. Other methods are better suited to find such structures, such as FFT in the case of periodicity.

6.3 Choosing Proper Parameter Values

Based on the previous experiments we draw the following conclusions regarding the parameter choices:

- We find that the accuracy of the solution can be increased by using more repetitions in the Clara++ algorithm. We recommend at least 80 repetitions, instead of the default value of 5. More complex data and a larger set of window lengths possibly require more repetitions.
- Increasing the number of samples also has a strong effect on the accuracy of the solution. However, increasing the value should be done with care, as it the

computational complexity is quadratic in the number of samples. We recommend to use 80 samples or more, instead of the default value of 40+2k.
- The number of samples in the Window-Trace matrix can be small; 32 samples is sufficient for a Bernoulli sequence.
- For these synthetic sequences, the uncertainty present in the data is larger than the variability of the solutions for the optimisation problem.

7 Evaluation on Real Data

To evaluate the usefulness of our problem setting in practice, we have designed four experiments on real data. In Section 7.1 we consider tracking the frequency of several words of varying type and frequency throughout the novel *Pride and Prejudice*. In Section 7.2 we study what window lengths would be appropriate for tracking the evolution of hapax legomenon ratio throughout texts from various genres. In Section 7.3 we examine tracking the frequency of nucleotides and dinucleotides in two reference genomes from the NCBI repository, and in Section 7.4 we identify the appropriate window lengths for analysing multi-scale structured time series.

7.1 Optimal Window Lengths for Several Words

Burstiness (Katz 1996) and *dispersion* (Gries 2008) of words in natural language corpora have become important concepts in research in linguistics (Gries 2008), natural language processing (Madsen et al 2005) and text mining (Lijffijt et al 2011). Burstiness and dispersion are used interchangeably to refer to measures for the variability of the frequency of a word, i.e., a poorly dispersed or very bursty word tends to be highly frequent in some (parts of) texts and infrequent in all other (parts of) texts. In Section 5, we have shown that the optimal set of window lengths does not have a relation to the frequency of the event studied, thus it would be interesting to know if the optimal set of window lengths does depend on the burstiness of an event in a sequence.

To test this, we conducted the following experiment. We downloaded the popular novel *Pride and Prejudice* by Jane Austen, which is freely available through Project Gutenberg (http://www.gutenberg.org/). The novel has approximately 120,000 words. We selected the 30 most and least bursty words with a frequency of at least 100. In this case, we measured the burstiness of a word by fitting a Weibull distribution to the inter-arrival time distribution of the word, then the shape parameter of the distribution is a measure for burstiness (Altmann et al 2009; Lijffijt et al 2011). The Weibull (or stretched exponential) distribution is a two-parameter exponential family distribution which can be used to model the distribution of the interarrival-times of the words. To study the effect of burstiness on the optimal sets of window lengths, we used $k = 3$ and window lengths from 1 to 2000.

The result is shown in Figure 8. The value for the Weibull β parameters are given at the top of the figure. We identify a clear trend: for the two smaller window lengths, we observe that these are longer for bursty words than for non-bursty words, although there is quite some variation within the two groups. The effect is strongest

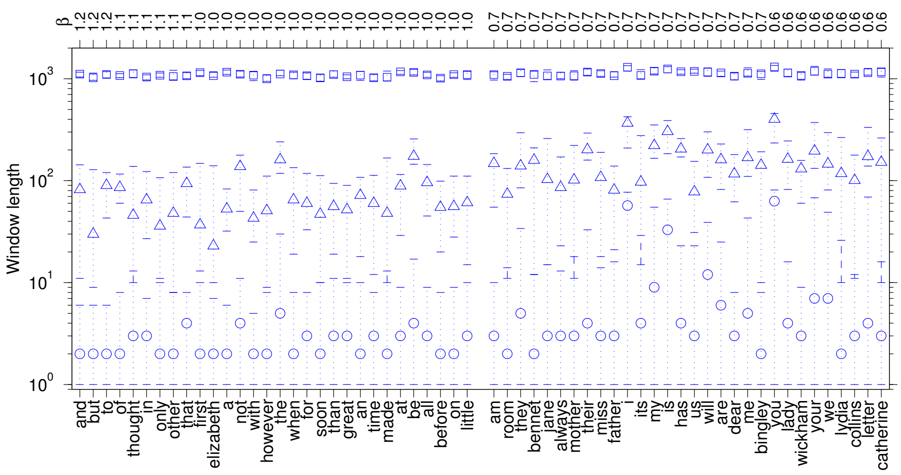

Fig. 8 Optimal sets of window lengths for analysing the evolving frequency of 60 words in the novel *Pride and Prejudice*, for $k = 3$. Squares, triangles and circles represent the medians for various window lengths, the dotted lines represent 90 % confidence intervals and dashed lines denote that the confidence intervals for the window lengths are overlapping. The words are sorted by burstiness, i.e., the Weibull β parameter. We observe that the variability of the window lengths over different runs is quite large, but that the algorithm chooses significantly longer window sizes in the case of bursty words. This trend is most visible for the middle window lengths (triangles).

for the words "I" and "you", which are the most frequent bursty words. Although the effect is weak, the average and median window lengths for the longest windows are also higher for bursty words than for non bursty words (mean/median non-bursty vs. bursty: 1083/1083 vs. 1136/1134, std. non-bursty vs. bursty: 63 vs. 81).

The fact that bursty words give longer window lengths may be due to the fact that they exhibit a larger scale structure (bursts and intervals between bursts) than the more uniformly distributed non-bursty words. The variation over individual words inside the groups is likely due to an interaction with the frequency of the word and because the Weibull β conveys not exactly the same 'burstiness' as is captured by our method.

7.2 Hapax Legomenon Ratio in Several Genres

The genre of a text largely determines its structure, which can be measured in terms of several linguistic features, for example the hapax legomenon ratio of texts (Biber 1988). We investigated if the optimal set of window lengths shows significant variation over texts from different genres, using the British National Corpus (The British National Corpus 2007) and the genre annotation from (Lee 2001). We have randomly sampled 100 texts from the BNC for each of the main genres in the corpus: conversation, imaginative fiction, academic prose and newspaper texts. The statistic used is now hapax legomenon ratio, as given in Equation (2), and we have used window lengths from 1 to 1,000 and $k = 3$.

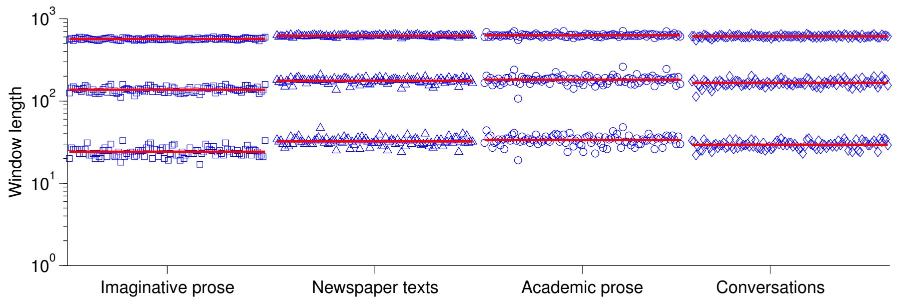

Fig. 9 Optimal sets of window lengths for analysing the evolving Hapax legomenon ratio for 400 texts from the British National Corpus, for various genres. Each point corresponds to a window length selected for that book, given the value of k, and red lines present the averages of the four genres. Both imaginative prose and conversations are significantly different from the other three genres ($p < 10^{-3}$, Wilcoxon rank-sum test, all three averages). The selected window lengths are shorter, possibly indicating a more uniform scale structure.

The result is shown in Figure 9. Although the set of window lengths varies over texts within each genre, we find that imaginative prose and conversations each seem to have a different structure than the texts from other three genres. This suggests that the scale structure of imaginative prose is more uniform than for other genres. A likely explanation is that the texts in the imaginative prose class are long coherent stories, while the texts in the other classes are collections of articles, topics and conversations.

7.3 Frequency of Nucleotides throughout DNA

Studies in biology and bioinformatics have shown that DNA chains consist of a number of important, known functional regions, at both large and small scales, which contain a high occurrence of one or more nucleotides (Papapetrou et al 2006). Examples of such regions include: *isochores*, which correspond to multi-megabase regions of genomic sequences that are specifically GC-rich or GC-poor and exhibit greater gene density; *CpG islands*, that correspond to regions of several hundred nucleotides that are rich in the dinucleotide CpG which is generally under-represented (relative to overall GC content) in eukaryotic genomes and their presence in the genome has been associated with gene expression in nearby genes.

We have studied Chromosome 1 of two organisms: `Homo Sapiens` (human) and `Canis Familiaris` (dog), of lengths 225 and 122 million nucleotides, respectively. The data has been downloaded from the NCBI data repository[3]. We focused on six event types: the four nucleotides A, C, G, and T, as well as dinucleotides TA and CG. We tested our algorithm using $k = 3$ and window lengths up to 10,000. The statistic used in our experiments was the relative event frequency and we sampled 1,000 columns for the W-T matrix.

In Figure 10 we see a comparison of the best window lengths found by our algorithm for the two organisms. We observe that the four single nucleotides as well as

[3] `http://www.ncbi.nlm.nih.gov`

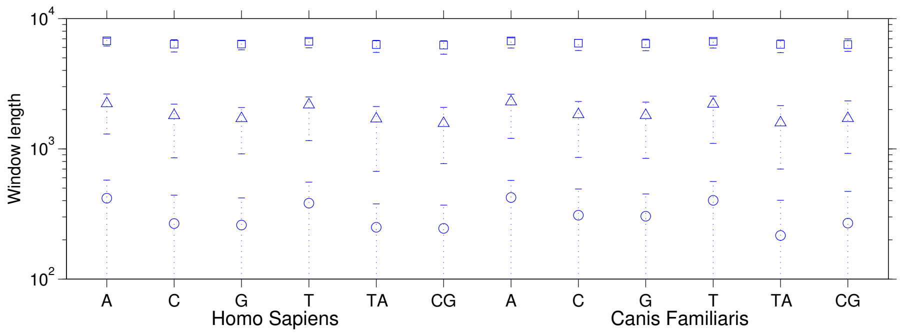

Fig. 10 Optimal sets of window lengths for the evolving frequency of nucleotides in Homo Sapiens chromosome 1 and dog chromosome 1, for $k = 5$.

the two dinucleotides exhibit highly similar behaviour for both organisms. This is explained by the high genomic structural similarity between humans and dogs (Kirkness et al 2003). Nonetheless, we see that the nucleotides C and G and both dinucleotides behave substantially different from the nucleotides A and T. One may think that this is merely a frequency effect, as the nucleotides A and T are much more frequent. However, from Section 5 we know that the frequency of an event does affect the distances between window lengths, but not the clustering.

Figure 11 illustrates the running frequency over the first 200,000 bases for chromosome 1 of Homo Sapiens, for all four nucleotides and the two dinucleotides, using the optimal window lengths. We observe that the different window lengths give somewhat different views of the data. As expected, the exact locations of bursts of the dinucleotide are identified most accurately by the shortest window length. However, the significance of each burst is seen directly from the line corresponding to the longest window length, since that line takes a fairly constant value throughout most of the sequence. Hence, there is clear value in using multiple window lengths, although in case two window lengths may be sufficient.

7.4 Smoothing of Time Series

An example of a time series with multi-scale structure comes from the Infrawatch project (Knobbe et al 2010; Vespier et al 2012). The data consists of 24 hours of measurements from a strain sensor on a bridge (the *Hollandse brug* in the Netherlands). The data contains structure at three time scales: a high frequency component generated by individual cars and trucks passing on the bridge, a medium frequency component generated by traffic jams, and a low frequency component generated by weather effects (e.g., temperature). The sensor was sampled at 10 Hz and the total length of the time series is 860,953 measurements.

We tested our method on this data with the following parameter setting. Since the scale space is potentially very large, and the frequencies below 1 Hz (window length 10) are not interesting, we constructed the set of potentially interesting window

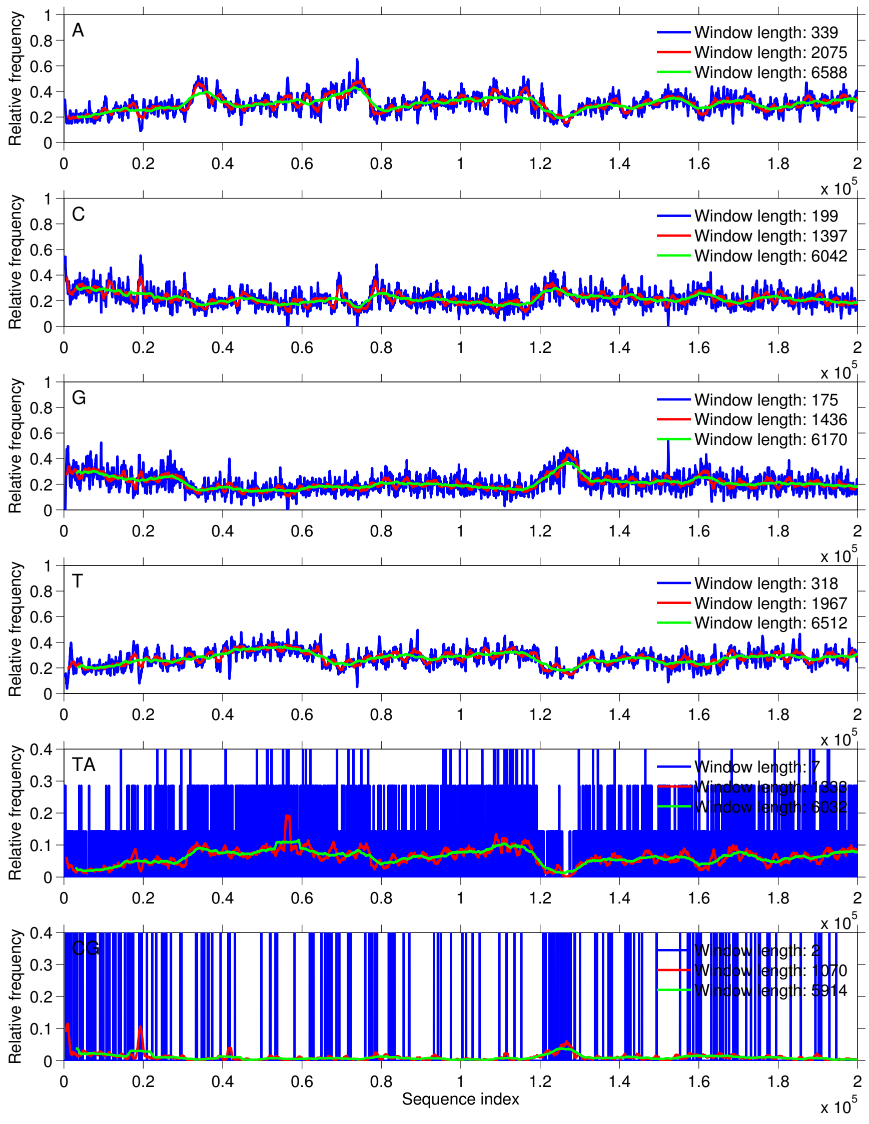

Fig. 11 Frequency of the studied (di-)nucleotides over the first 200,000 bases for chromosome 1 of Homo Sapiens, using the best window lengths for that (di-)nucleotide.

lengths $\Omega = \{10, 14, 20, \ldots, 163840, 231705\}$, i.e., rounded powers of $\sqrt{2}$ starting at window length 10. Also we use $N = 1,000$ and $k = 3$.

The result is presented in Figure 12. At a first glance, it is difficult to say whether the three window lengths correspond directly to the three time scales that are present in the data, because the window lengths correspond to different views and not to frequency bands, as studied, for example, by Vespier et al (2012). Still, we observe that the three window lengths give very different views of the data, each of which repre-

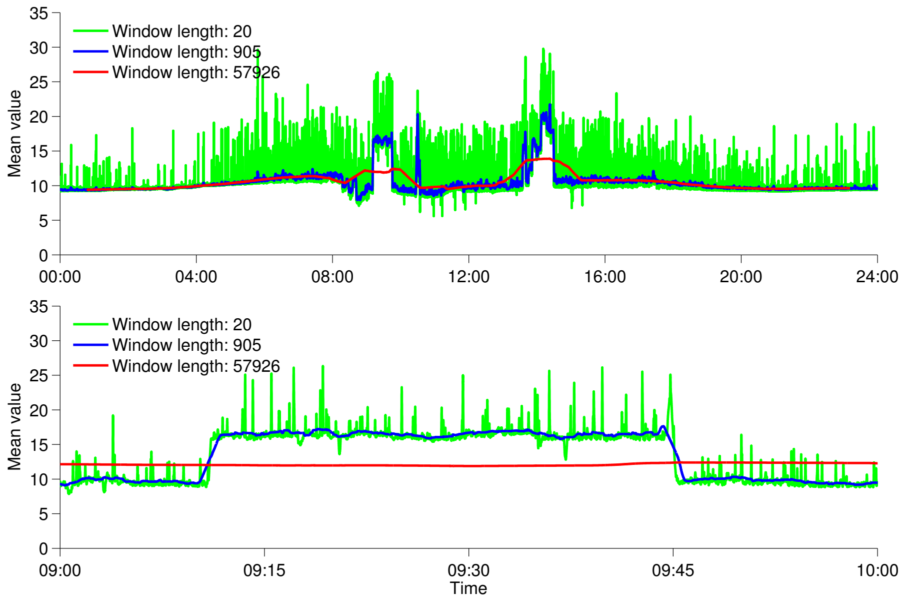

Fig. 12 Smoothing of a time-series containing measurements from a strain sensor on a bridge in the Netherlands, using an optimised set of three window lengths. The top figure shows the full sequence (24 hours), while the bottom figure shows a zoom-in on the traffic jam that occurred between 9am and 10am.

sents a different time scale. The substantial difference between the window lengths becomes most clear in the zoomed-in figure. Evidently, this provides strong support for our claim that it can be useful to study a set of window lengths.

8 Conclusions

We have studied the novel problem of identifying a set of window lengths that contain the maximal amount of information in the data. We have presented a generally applicable objective function that users could employ, which can also be efficiently optimised algorithmically, or analytically for certain simple statistics and data distributions. We have extensively studied the performance of the proposed optimisation algorithm, as well as the identified solutions for three examples of sliding window statistics on both synthetic data and real data. We have illustrated that the analytical results and the computational results on synthetic data are useful as a baseline for practical use. We have illustrated how sampling can be used to obtain the optimal set of window lengths more efficiently, making the method practical for (collections of) sequences of any size. Moreover, we have shown that the window lengths themselves can show interesting properties of the data; among other findings, we have identified relations between the optimal window lengths and (1) the structure of sequences composed of multiple interleaved sources and (2) the burstiness of events.

A question left for further research is how many window lengths a user should employ in practical settings. One may be able to find a good trade off by exploring the value of the loss function for various number of window lengths, while an alternative approach is to just select the number of window lengths as high as practically feasible, since more windows is always more informative. Although it was our initial goal, we have not explored the use of the method in an interactive setting, where a user could for example fix one or more window lengths in advance, give the optimisation algorithm hints about good window lengths, or construct the set of window lengths interactively. There are clearly many interesting opportunities for future research in this direction.

Acknowledgements. We thank Heikki Mannila for useful discussions and feedback. This work was supported by the the Finnish Doctoral Programme in Computational Sciences (FICS), the Finnish Centre of Excellence for Algorithmic Data Analysis Research (ALGODAN) and the Finnish Centre of Excellence in Computational Inference Research (COIN). We acknowledge the computational resources provided by Aalto Science-IT project.

References

Aloise D, Deshpande A, Hansen P, Popat P (2009) NP-hardness of Euclidean sum-of-squares clustering. Machine Learning 75:245–248

Altmann EG, Pierrehumbert JB, Motter AE (2009) Beyond word frequency: Bursts, lulls, and scaling in the temporal distributions of words. PLoS ONE 4(11):e7678

Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proc. of SODA

Benson G (1999) Tandem repeats finder: a program to analyze DNA sequences. Nucleic Acids Research 27(2):573–580

Biber D (1988) Variation across speech and writing. Cambridge University Press

Bourgain C, Genin E, Quesneville H, Clerget-Darpoux F (2000) Search for multifactorial disease susceptibility genes in founder populations. Annals of Human Genetics 64(3):255–265

Calders T, Dexters N, Goethals B (2008) Mining frequent items in a stream using flexible windows. Intelligent Data Analysis 12(3):293–304

Chiu B, Keogh E, Lonardi S (2003) Probabilistic discovery of time series motifs. In: Proc. of ACM SIGKDD, pp 493–498

Das MK, Dai HK (2007) A survey of DNA motif finding algorithms. BMC Bioinformatics 8(Suppl 7):S21

Demaine ED, López-Ortiz A, Munro JI (2002) Frequency estimation of internet packet streams with limited space. In: Proc. of ESA, pp 348–360

Giannella C, Han J, Robertson E, Liu C (2003) Mining frequent itemsets over arbitrary time intervals in data streams. Tech. Rep. Technical Report TR587, Indiana University

Golab L, DeHaan D, Demaine ED, López-Ortiz A, Munro JI (2003) Identifying frequent items in sliding windows over on-line packet streams. In: Proc. of IMC, pp 173–178

Gries ST (2008) Dispersions and adjusted frequencies in corpora. International Journal of Corpus Linguistics 13(4):403–437

Jin C, Yi K, Chen L, Yu JX, Lin X (2008) Sliding-window top-k queries on uncertain streams. The VLDB Journal

Jin R, Agrawal G (2005) An algorithm for in-core frequent itemset mining on streaming data. In: Proc. of IEEE ICDM, pp 210–217

Karp RM, Shenker S, Papadimitriou CH (2003) A simple algorithm for finding frequent elements in streams and bags. ACM Transactions on Database Systems 28(1):51–55

Katz SM (1996) Distribution of content words and phrases in text and language modelling. Natural Language Engineering 2(1):15–59

Kaufman L, Rousseeuw PJ (1990) Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons

Kirkness EF, Bafna V, Halpern AL, Levy S, Remington K, Rusch DB, Delcher AL, Pop M, Wang W, Fraser CM, Venter JC (2003) The dog genome: survey sequencing and comparative analysis. Science 301(5641):1898–1903

Knobbe A, Blockeel H, Koopman A, Calders T, Obladen B, Bosma C, Galenkamp H, Koenders E, Kok J (2010) Infrawatch: Data management of large systems for monitoring infrastructural performance. In: Proc. of IDA, pp 91–102

Lee DYW (2001) Genres, registers, text types, domains and styles: clarifying the concepts and navigating a path through the BNC jungle. Language Learning & Technology 5(3):37–72

Li C, Wang B, Yang X (2007a) VGRAM: improving performance of approximate queries on string collections using variable-length grams. In: Proc. of VLDB, pp 303–314

Li Y, Sung WK, Liu JJ (2007b) Association mapping via regularized regression analysis of single-nucleotide-polymorphism haplotypes in variable-sized sliding windows. American Journal of Human Genetics 80(4):705–715

Li Y, Lin J, Oates T (2012) Visualizing variable-length time series motifs. In: Proc. of SDM, pp 895–906

Lijffijt J, Papapetrou P, Puolamäki K, Mannila H (2011) Analyzing word frequencies in large text corpora using inter-arrival times and bootstrapping. In: Proc. of ECML-PKDD, pp 341–357

Lijffijt J, Papapetrou P, Puolamäki K (2012) Size matters: Finding the most informative set of window lengths. In: Proc. of ECML-PKDD, pp 451–466

Lin CH, Chiu DY, Wu YH, Chen ALP (2005) Mining frequent itemsets from data streams with a time-sensitive sliding window. In: Proc. of SDM

Madsen RE, Kauchak D, Elkan C (2005) Modeling word burstiness using the dirichlet distribution. In: Proc. of ICML, pp 545–552

Mannila H, Toivonen H, Verkamo AI (1997) Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3):259–289

Mathias RA, Gao P, Goldstein JL, Wilson AF, Pugh EW, Furbert-Harris P, Dunston GM, Malveaux FJ, Togias A, Barnes KC, Beaty TH, Huang SK (2006) A graphical assessment of p-values from sliding window haplotype tests of association to identify asthma susceptibility loci on chromosome 11q. BMC Genetics 7:38

Mueen A (2013) Enumeration of time series motifs of all lengths. In: Proc. of ICDM, pp 547–556

Mueen A, Keogh EJ, Zhu Q, Cash S, Westover B (2009) Exact discovery of time series motifs. In: Proc. of SDM, pp 473–484

Pakhira MK (2008) Fast image segmentation using modified CLARA algorithm. In: Proc. of ICIT, pp 14–18

Papadimitriou S, Yu P (2006) Optimal multi-scale patterns in time series streams. In: Proc. of ACM SIGMOD, pp 647–658

Papapetrou P, Benson G, Kollios G (2006) Discovering frequent poly-regions in dna sequences. In: Proc. of IEEE ICDM Workshops, pp 94–98

Papapetrou P, Benson G, Kollios G (2012) Mining poly-regions in dna sequences. International Journal of Data Mining and Bioinformatics (IJDMB) 6(4):406–428

Sörnmo L, Laguna P (2005) Bioelectrical Signal Processing in Cardiac and Neurological Applications. Elsevier Academic Press

Tang R, Feng T, Sha Q, Zhang S (2009) A variable-sized sliding-window approach for genetic association studies via principal component analysis. Annals of Human Genetics 73(Pt 6):631–637

The British National Corpus (2007) Version 3 (BNC XML Edition). Distributed by Oxford University Computing Services on behalf of the BNC Consortium, URL `http://www.natcorp.ox.ac.uk/`

Toivonen H, Onkamo P, Vasko K, Ollikainen V, Sevon P, Mannila H, Herr M, Kere J (2000) Data mining applied to linkage disequilibrium mapping. American Journal of Human Genetics 67(1):133–145

Vespier U, Knobbe A, Nijssen S, Vanschoren J (2012) Mdl-based analysis of time series at multiple time-scales. In: Proc. of ECML-PKDD, pp 371–386

Yingchareonthawornchai S, Sivaraks H, Rakthanmanon T, Ratanamahatana CA (2013) Efficient proper length time series motif discovery. In: Proc. of ICDM, pp 1265–1270

A Proof of Theorem 1.

Preliminaries. Let $(X_1,\ldots,X_n)$ be a sequence of Bernoulli random variables with common parameter p, i.e., $X_i \in \{0,1\}, \Pr(\{X_i = 1\}) = p$, for all $i \in \{1,\ldots,n\}$. The random variables could, for example, denote the occurrences of an event. Similar to the notation for event sequences, we use $X_{i,\omega}$ to denote the subsequence of length ω starting at position i, $(X_i,\dots,X_{i+\omega-1})$. Let the statistic f be the relative frequency of ones:

$$f(X_{i,\omega}) = \frac{1}{\omega}\sum_{j=i}^{i+\omega-1} X_j. \tag{5}$$

The selection of an optimal set of window lengths is based on the squared error between predictions made using those window lengths (Problem 1). Under the constraint of using a k-partition nearest neighbour regressor, the predictions correspond to the value of the nearest window length (Section 4.1). Thus, to select the optimal window lengths, we have to compute the distance (squared error) between all pairs of window lengths. We find that the distance between window lengths is as follows.

Theorem 1 *For the statistic and generative process described above, the expected distance between two window lengths γ and ω, with $\gamma < \omega$, is*

$$\mathrm{E}[d(\omega,\gamma)] = \frac{\omega-\gamma}{\omega\gamma}p(1-p).$$

Proof The expected distance between two window lengths γ and ω is

$$\mathrm{E}[d(\omega,\gamma)] = \mathrm{E}\left[\frac{1}{n^*}\sum_{i=1}^{n^*}\left(f(X_{i,\gamma})-f(X_{i,\omega})\right)^2\right].$$

Since $X_1,\dots,X_n$ are i.i.d. random variables, this simplifies to

$$\mathrm{E}[d(\omega,\gamma)] = \mathrm{E}\left[\left(f(X_{1,\gamma})-f(X_{1,\omega})\right)^2\right].$$

Assuming without loss of generality that $\gamma < \omega$, we find that

$$\begin{aligned}
f(X_{1,\omega}) &= \frac{1}{\omega}\sum_{j=1}^{\omega} X_j \\
&= \frac{1}{\omega}\sum_{j=1}^{\gamma} X_j + \frac{1}{\omega}\sum_{j=1+\gamma}^{\omega} X_j \\
&= \frac{\gamma}{\omega} f(X_{1,\gamma}) + \frac{\omega-\gamma}{\omega} f(X_{1+\gamma,\omega-\gamma}).
\end{aligned}$$

Thus we can rewrite the expected distance as

$$\begin{aligned}
&\mathrm{E}[d(\omega,\gamma)] \\
&= \mathrm{E}\left[\left(f(X_{1,\gamma}) - \frac{\gamma}{\omega}f(X_{1,\gamma}) - \frac{\omega-\gamma}{\omega}f(X_{1+\gamma,\omega-\gamma})\right)^2\right] \\
&= \mathrm{E}\left[\left(\frac{\omega-\gamma}{\omega}\right)^2\left(f(X_{1,\gamma}) - f(X_{1+\gamma,\omega-\gamma})\right)^2\right] \\
&= \left(\frac{\omega-\gamma}{\omega}\right)^2 \mathrm{E}\left[\left(f(X_{1,\gamma}) - f(X_{1+\gamma,\omega-\gamma})\right)^2\right] \\
&= \left(\frac{\omega-\gamma}{\omega}\right)^2 \mathrm{E}\left[f(X_{1,\gamma})^2\right] + \mathrm{E}\left[f(X_{1+\gamma,\omega-\gamma})^2\right] - 2\,\mathrm{E}\left[f(X_{1,\gamma})f(X_{1+\gamma,\omega-\gamma})\right].
\end{aligned}$$

These three expectations are

$$\begin{aligned}
\mathrm{E}\left[f(X_{1,\gamma})^2\right] &= \frac{p(1-p)}{\gamma} + p^2, \\
\mathrm{E}\left[f(X_{1+\gamma,\omega-\gamma})^2\right] &= \frac{p(1-p)}{\omega-\gamma} + p^2, \text{ and} \\
\mathrm{E}\left[f(X_{1,\gamma})f(X_{1+\gamma,\omega-\gamma})\right] &= p^2.
\end{aligned}$$

For brevity, we skip the derivation for these three expectations. They can be derived, for example, using the fact that the variance of a binomial distribution is $\mathrm{Var}[Bin(n,p)] = \mathrm{E}\left[Bin(n,p)^2\right] - \mathrm{E}[Bin(n,p)]^2 = np(1-p)$, and its expectation is $\mathrm{E}[Bin(n,p)] = np$.

By writing out the expected distance we find that

$$\begin{aligned}
\mathrm{E}[d(\omega,\gamma)] &= \left(\frac{\omega-\gamma}{\omega}\right)^2 \frac{p(1-p)}{\gamma} + p^2 + \frac{p(1-p)}{\omega-\gamma} + p^2 - 2p^2 \\
&= \left(\frac{\omega-\gamma}{\omega}\right)^2 \left(\frac{1}{\gamma} + \frac{1}{\omega-\gamma}\right) p(1-p) \\
&= \frac{(\omega-\gamma)^2}{\omega^2} \frac{\omega-\gamma+\gamma}{\gamma(\omega-\gamma)} p(1-p) \\
&= \frac{\omega-\gamma}{\omega\gamma} p(1-p).
\end{aligned}$$