On the Computation and Application of Prototype Point Patterns

Katherine E. Tranbarger[1] and Frederic Paik Schoenberg[1]

Abstract

This work discusses computational problems related to the implementation of Victor and Purpura's spike time distance metric for point processes. Three algorithms for calculation of spike-time distance are examined as are a number of properties and extensions of the spike-time metric. Extensions include prototype point patterns that can be used for describing a typical point pattern and various clustering algorithms that can be applied to point process data through use of spike-time distance and prototype patterns.

Key words: distance metric, point process, prototype pattern.

[1] Department of Statistics, 8125 Math-Science Building, University of California, Los Angeles, 90095-1554.

1 INTRODUCTION

The problem addressed in this paper is how to define and compute the distance between two observed point patterns. Methods involving distance metrics for point patterns have received increasing use recently. Such methods have had particularly important applications in the analysis of collections of neuron spike trains (Victor and Purpura 1997; Reich et al. 2000). In addition, point pattern distance metrics are required for the computation of a prototype point pattern, a construct introduced in Schoenberg and Tranbarger (2004) and shown to be useful in the description of earthquake aftershock sequences. The present paper investigates the computation and implementation of such distance metrics, especially in the context of their use in prototype point pattern analysis.

The spike-time distance metric proposed by Victor and Purpura (1997) involves matching points in one point pattern to the points in another point pattern. The metric is somewhat analogous to several distance metrics rich with history in mathematics and computer science that are currently in use in computer imaging and other areas. For instance, the concept of Earth Movier's Distance (EMD) introduced by Rubner, Tomasi, and Guibas (1998) evolved from the Hitchcock (1941) solution to the original transportation problem first discussed by Monge (1781). EMD measures the amount of work required to transform a histogram of the values in one image into that of the values in another image using basic operations (see Rubner et al., 2000), and was shown by Levina and Bickel (2004) to be equivalent to Mallows distance on probability distributions when the two image signatures in question are appropriately weighted by their sizes.

In contrast to EMD, the spike-time distance of Victor and Purpura (1997) focuses on matching the points of the two processes rather than their summary histograms. In addition, unlike EMD, the spike-time distance can be used to compare two point patterns of unequal lengths, and no modification involving only partial matching is required. Indeed, the difference in the number of points is a key ingredient in spike-time distance.

In Section 2, we review spike-time distance and explore various properties that aid in its computation. Three different algorithms that can be used to compute spike-time distance are presented here. Prototype point patterns are described in Section 3 and issues in their computation and approximation are discussed. Algorithms for clustering collections of observed point patterns based on their spike-time distances and prototype point patterns are discussed in Section 4. A discussion and suggestions for further research are presented in Section 5.

2 CALCULATION OF SPIKE-TIME DISTANCE

Let X and Y be temporal point patterns, i.e. each is a collection of points on the real line, and corresponds to a σ-finite non-negative integer-valued measure on $\mathbf{R}$ (see Daley and Vere-Jones 2003). (We refer to such a collection of points as a point *pattern*, as distinguished from a point *process*, which is a random variable whose outcomes are point patterns.)

The spike-time distance between X and Y is defined as the total cost of transforming X into Y using a series of basic operations (Victor and Purpura, 1997). For instance, in the standard formulation, points from X can be deleted at a cost p_d, added at a cost of p_a, or moved horizontally a distance Δ at a cost of $p_m\Delta$. The minimum sum of costs for the deletion, addition, and moving operations necessary to transform point pattern X into point pattern Y is the spike-time distance between X and Y. Note that in order for this to be a distance metric, it must be symmetric and thus the constraint $p_a = p_d$ must be imposed. While in some applications it might be desirable for p_a and p_d to take on different values, this paper focuses on the case where the spike-time distance is a symmetric distance metric. Hence in what follows we assume $p_a = p_d$.

One algorithm for determining the minimal sequence of deletion, addition and movement operations needed for transforming one point pattern into another is the Single-Unit (SU) algorithm discussed in Aronov (2003). This work will briefly review the SU algorithm based on dynamic programming algorithms introduced by Sellers (1974) and show a modification on this approach that reduces computation run time from $O(n^2)$ to $O(n^{4/3})$. Also presented here is the Mutual Best Match (MBM) algorithm that is useful for determining distances between patterns of shorter lengths and which has the ability to extend into multiple dimensions as will be discussed in section 5.

2.1 Single-Unit algorithm

In general, the minimal sequence of operations required to transform point pattern X into point pattern Y can be difficult to determine. In the special case where no points of X or Y are added or deleted, however, the computation is quite trivial in view of the following result that is key in the application of all three algorithms to be discussed.

Theorem 1. Suppose that temporal point patterns X and Y each consist of exactly n points and that $p_m << p_a = p_d$, so that in effect addition and deletion of points are not permitted. Then

$$d(X,Y) = p_m \sum_{i=1}^{n} |x_i - y_i|, \tag{1}$$

where $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ are the sorted points of X and Y, respectively.

Proof.

The statement is trivial for $n = 1$.

Suppose $n = 2$, and without loss of generality assume $x_1 \geq y_1$.

If also $x_2 \geq y_2$, then

$$|x_1 - y_2| + |x_2 - y_1| \geq |x_1 - y_1 + x_2 - y_2| = |x_1 - y_1| + |x_2 - y_2|,$$

since both $(x_1 - y_1)$ and $(x_2 - y_2)$ are non-negative.

Alternatively, if $x_2 < y_2$, then $y_2 - x_1 \geq y_2 - x_2 > 0$, since $x_1 \leq x_2 < y_2$. Similarly, $0 \geq y_1 - x_1 \geq y_1 - x_2$. Therefore, $|y_1 - x_1| \leq |y_1 - x_2|$ and $|y_2 - x_2| \leq |y_2 - x_1|$, so $|y_1 - x_1| + |y_2 - x_2| \leq |y_1 - x_2| + |y_2 - x_1|$.

Hence the basic operations that move the point x_1 to y_2 and move the point x_2 to y_1 have a total penalty that is at least as large as the penalties for moving x_1 to y_1 and moving x_2 to y_2.

Now suppose $n = k + 1$ and that the result in Theorem 1 holds for $n = k$. Consider any sequence of elementary operations that includes moving x_{k+1} to y_i and moving x_j to y_{k+1}, where $i, j < k + 1$. By the $n = 2$ case proven above, $|x_{k+1} - y_i| + |x_j - y_{k+1}| \geq |x_j - y_i| + |x_{k+1} - y_{k+1}|$. That is, the elementary operations considered have cost at least as large as those obtained by moving x_{k+1} to y_{k+1} and moving x_j to y_i. Hence the minimal total cost in aligning X and Y is obtained by the elementary operations that involve moving x_{k+1} to y_{k+1}; the cost of aligning the other k points of X with the other k points of Y is given by (1) with $n = k$. The result folllows by induction. □

Utilizing this result, an inductive approach to spike-time distance calculation is possible. The SU algorithm (see Aronov 2003)determines the minimal distance between $i = 1, 2..., n$ points of pattern X and $j = 1, 2..., m$ points of pattern Y by building on the distance result for points $i = 1, 2..., n-1$ of X and points $j = 1, 2..., m-1$ of Y. Let $D_{i,j}$ denote the spike-time distance between the first i points of pattern X and first j points of pattern Y. To determine the minimal distance between patterns X and Y simple calculations to complete an n by m matrix of $D_{i,j}$ values are necessary. At each step $D_{i,j}$ is the minimum of $D_{i-1,j} + p_d$, $D_{i,j-1} + p_d$, and $D_{i-1,j-1} + (p_m * |x_i - y_j|)$ (corresponding to the options of deletion of x_i, deletion of y_j, and the pairing of x_i to y_j, respectively). In the trivial cases of $i = 0$ or $j = 0$, $D_{i,j} = p_d * max(i, j)$.

2.2 Modified Single-Unit algorithm

In spike-time distance calculations, no move greater than $(p_a + p_d)/p_m$ may take place since such a move would be greater in cost than removal and re-insertion of one of the points involved. With this fact in mind, the patterns X and Y can each be broken into patterns of shorter lengths whenever a gap of greater than $(p_a + p_d)/p_m$ is present. In this modified SU (MSU) approach, computations involve the sorting of points $x_1, x_2, ...x_n$ and points $y_1, y_2, ...y_m$ combined to form one pattern, Z, of length $n + m$. Instances in which z_k-z_{k-1} is greater than $(p_a + p_d)/p_m$ are easily identified to determine how X and Y can be partitioned into patterns of shorter lengths. For example, if such a gap was present at z_k=3, pattern X_1 would consist of all points in X less than 3 just as pattern Y_1 would consist of all points in Y less than 3. The SU algorithm as earlier described is then applied to each pair of X_i, Y_i patterns formed in the partitioning process and the final distances for each pair are summed. Advantages of this modification are discussed in Section 2.4.

2.3 Mutual Best Match algorithm

In light of Theorem 1, the problem of determining the best sequence of operations to transform point pattern X into point pattern Y reduces to the problem of determining whether each point in the point patterns will be *removed* (i.e. deleted from one string or equivalently added to the other), or whether it will be *kept*, i.e. *paired* to a point in the other string. Once the points to be kept are determined, the best approach for matching the kept points in X to those in Y is simply sequential, based on Theorem 1. Evaluation of each potential set of kept points is thus remarkably straightforward. In fact, each potential distance is simply the number of points removed summed with the integrated difference between the cumulative functions associated with the remaining points in the two temporal point patterns (Schoenberg and Tranbarger 2004).

Given two temporal point patterns X and Y whose spike-time distance is sought, the following results are useful in determining which points will be kept and which will be removed.

Lemma 2. Any point that is more than $(p_a + p_d)/p_m = 2p_d/p_m$ away from its nearest neighbor in the alternate point pattern will be removed.

Proof. The proof is immediate: for such a point x in X, it is more costly to move x to a point y in Y than to delete x and add to X a point at y. □

Theorem 3. Any pair of points x_i and y_j such that

$$|x_i - y_j| < \min_{k \neq i} \{|x_k - y_j|\} < \min_{k \neq j} \{|x_i - y_k|\} < 2p_d/p_m \tag{2}$$

will be kept.

Proof. There are three possible outcomes for points x_i and y_j satisfying the condition (2). Either both points are removed, one point is kept while the other is removed, or both points are kept. We will show that the first two of these outcomes are excluded.

Suppose that both points are removed. Then the distance penalty function includes the penalty $2 * p_d$ associated with those deletions. Since $|x_i - y_j| < 2 * p_d/p_m$, the penalty could be reduced by keeping both points and moving them to each other, at a cost of $|x_i - y_j| * p_m$. Therefore, removing both x_i and y_j cannot yield the minimum penalty.

Suppose that one point is kept while the other is removed. Assume without loss of generality that x_i is kept while y_j is removed. Then x_i is paired with some point y_k such that $y_k \neq y_j$ and the spike-time distance includes both penalty p_d for removal of y_j and $|x_i - y_k| * p_m$ for the move of point x_i to y_k. Since $|x_i - y_k| > |x_i - y_j|$ the total spike-time distance penalty could be reduced by removing point y_k rather than point y_j and moving x_i to y_j. Therefore, removing one point while keeping the other cannot yield the minimum penalty.

The only option that remains is that both x_i and y_j are kept in the sequence of moves yielding minimal total cost. □

The Mutual Best Match (MBM) algorithm, named for the property shown in Theorem 3, uses the three discussed properties of the optimal sequence of operations in the spike-time distance metric to determine the distance between point patterns X and Y. With these properties, the problem of determining the distance between point pattern X and point pattern Y is simplified to identifying points known to be kept or deleted, then checking the total cost associated with each possible combination of potentially kept points. Calculating the total cost associated with each potential combination of kept points in the two point patterns is not computationally time prohibitive, as the sequential ordering prescribed by Theorem 1 makes this calculation extremely straight-forward.

2.4 Application and penalty selection

It is not uncommon in point process applications to observe a collection of independent realizations of point processes observed on a common space. An example discussed in Schoenberg and Tranbarger (2004) is the collection of observed aftershock sequences following global mainshocks in distinct regions, occurring within a fixed period of time and space of each mainshock. Throughout this work, we refer to such a collection of point processes simply as a *point process dataset*.

The number and proximity of points in each point pattern observed in a point process dataset are important to consider when determining the penalties used in the distance metric. When selecting these penalties, it is the ratio of the deletion (addition) penalty p_d to p_m that governs the results. For simplicity, we suggest setting $p_a = p_d = 1$ and determining p_m by examining the spread of the points in the data. If p_m is too large, then little movement of points will take place in the computation of distances between point patterns, as the cost of deletion and addition will be less than most potential moves. Alternatively, if p_m is too small, then moves will be made between points that are not very close at all. With either extreme, the resulting spike-time distance will measure little more than the sum of, or difference between, the number of points in the two observed point patterns. That is, the spike-time distances will approach the sum of the two point pattern lengths for very large values of p_m, and will approach the absolute difference between these lengths for very small p_m values.

With these extremes in mind, one option is to set p_m to a value such that points closer than the the typical inter-point distance are paired, and points further than this are removed and reintroduced in calculating the spike time distance. This leads to selecting penalties such that:

$$(T/M)p_m \approx 2p_d, \tag{3}$$

where M is the median number of points per observed point pattern and T is the size of the range over which the points are observed.

In simulation studies on point patterns generated from a Poisson process the precise value of the penalties selected has little influence on the overall distribution of pattern distances. Figure 1 shows the results from calculating the 4,950 possible pairwise distances between 100 simulated Poisson point patterns. The results are shown for three different values of p_m: the value as determined by (3), 75% of the penalty recommended in (3), and 125% of the penalty found by (3). In each case, though the mean and standard deviation of the distances has changed as would be expected, the distribution of distances is approximately normal for all three. Normal Q-Q plots are provided below each histogram of distances to further illustrate how closely each follows the normal distribution. Since spike-time distance is useful only for comparisons of similarity rather than as a raw measurement tool (where specific distance values have meaning out of the comparison context) the fact that the distribution of distances remains virtually unchanged in the face of rather substantial alterations to the penalty values suggests that investigations of differences and similarities of point patterns within a dataset may not be overly sensitive to the choice of penalty values, provided the penalties are within some reasonable range, at least in so far as the shape of the distribution of the distances is concerned.

Calculation time is of course a key consideration in selection of which algorithm to use with a particular data set. The main advantage of the MSU approach is that it will often require significantly less time to calculate the spike-time distance of interest. This advantage over the standard SU algorithm can be clearly seen in Figure 2. Here, for lengths $n = 2, 3, 4..., 29, 30, 40, 50, 75, 100, 150, 200$, distances were calculated between 1000 pairs of two randomly generated patterns of n points each using both the SU and MSU algorithms programmed in R. For the modified version, additional results for $n = 300, 400, 500$ are also shown. The patterns used were generated by taking a random sample of size n from a uniform[0,10] distribution. Points in black are the resulting mean CPU times using the standard SU algorithm. The black line included is the best fit least-squared model using n^2 as the only predictor, demonstrating the $O(n^2)$ run time achieved by the SU algorithm. Points in red are fit with a linear model and are the result from using the MSU algorithm. Though the MSU algorithm should run $O(n^{4/3})$ due to the sorting calculation (Sedgewick, 1986), up to $n = 500$ a linear model fits the simulation results quite well as the sorting represents such a small part of all computations involved. In all cases, the penalties were fit as recommended above in (3). The ease of matrix manipulation in R is the main reason R was selected for implementation of the SU and MSU algorithms.

The run time for calculation of the spike-time distance using the MBM algorithm is approximately $O(2^{n_1+n_2})$, where n_1 and n_2 are the lengths of the two point patterns. This is because there are on the order of $O(2^{n_1+n_2})$ combinations of points which might be kept, and each of these is examined individually. Green points in Figure 3 illustrate this for the case $n = n_1 = n_2$, in which case the run time increases approximately as $O(4^n)$. In Figure 3, timing simulations using the MBM approach programmed in C are included with a subset of the same results shown in Figure 2. The overlaid green line corresponds to the best 4^n fit. C was selected for the MBM algorithm because of the looping that takes place in the MBM distance calculation. All of these computations were performed using a 1.33GHz PowerPC G4.

3 PROTOTYPE POINT PATTERNS

With a distance metric clearly defined, it becomes possible to identify a prototype point pattern that can be used for describing a typical observation within the point process dataset. We define this prototype to be the point pattern Y such that the sum

$$\sum_{i=1}^{n} d(X_i, Y), \tag{4}$$

is minimized, where X_i, $i = 1...n$, are the n observed point patterns in the dataset.

3.1 Basic properties of prototype points

Fortunately, one need not search over all possible point patterns in determining the prototype for a given point process dataset. A convenient feature making prototypes easy to identify is described in the next result. Before stating this fact, we first turn to the definition of the median of a sorted list of numbers

$\mathbf{z} = \{\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_m}\}$, whose length m is even. Many texts define the median of such a list as the mean of the two entries $z_{m/2}$ and $z_{m/2+1}$. Instead, let us refer to any value M such that $z_{m/2} \leq M \leq z_{m/2+1}$ as a median of $\mathbf{z}$.

With this convention in mind, we return to the problem of determining prototypes. Suppose that Y is the prototype of a point process dataset consisting of n point patterns $X_1, \ldots, X_n$. For any point p in the prototype, consider the collection of points in the point process dataset $\mathbf{z_p} = \{z_1, z_2, \ldots, z_m\}$ to which p is paired. That is, each point z_i is the point to which p is moved in determining the spike-time distance between Y and X_j, for some j. Note that $m \leq n$, since p might not be kept in the spike-time distance between the prototype and some of the point patterns in the dataset.

Theorem 4. Any point p in the prototype is a median of $\mathbf{z_p}$.

Proof. Fix any prototype point p and the list $\mathbf{z_p} = \{z_1, \ldots, z_m\}$ of points in the dataset to which p is paired. Note that, in order for p to be a point of the prototype, the sum of distances from z_i to p must be less than or equal to the sum of distances from z_i to any other point q. Let q be a median of $\mathbf{z_p}$, and suppose that p is not a median of $\mathbf{z_p}$. We will show that the sum of distances from z_i to q is less than the sum of distances from from z_i to p, contradicting the assumption that p is a point of the prototype.

First suppose that the length m of $\mathbf{z_p}$ is odd, and without loss of generality assume $p < q$. The sum of all m distances from z_i to q is:

$$\begin{aligned}\sum_{i=1}^{m} |z_i - q| &= \sum_{i=1}^{(m-1)/2} |z_i - q| + \sum_{i=(m+3)/2}^{m} |z_i - q| + |z_{(m+1)/2} - q| \\ &= \sum_{i=1}^{(m-1)/2} |z_i - q| + \sum_{i=(m+3)/2}^{m} |z_i - q| \qquad (5)\end{aligned}$$

since $|z_{(m+1)/2} - q| = 0$. The sum of the m distances from z_i to p is:

$$\begin{aligned}\sum_{i=1}^{m} |z_i - p| &= \sum_{i=1}^{(m-1)/2} (|z_i - q| - (q - p)) + \sum_{i=(m+3)/2}^{m} (|z_i - q| + (q - p)) + (z_{(m+1)/2} - p) \\ &= \sum_{i=1}^{m} |z_i - q| - (\frac{m-1}{2})(q - p) + (\frac{m-1}{2})(q - p) + (z_{(m+1)/2} - p) \\ &= \sum_{i=1}^{m} |z_i - q| + (z_{(m+1)/2} - p). \qquad (6)\end{aligned}$$

Therefore, the sum of distances from z_i to p in (6) is greater than the sum of distances to q, which is a contradiction.

If m is even, then without loss of generality assume that $p < z_{m/2}$. The sum of all m distances from z_i to q is:

$$\begin{aligned}\sum_{i=1}^{m} |z_i - q| = & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}| + (q - z_{(m/2)})) + \sum_{i=(m/2)}^{(m/2)+1} (|z_i - q|) \\ & + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}| + (z_{(m/2)+1} - q)) \\ = & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}|) + (z_{(m/2)+1} - z_{m/2}) + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}|) \\ & + (\frac{m}{2}) * (z_{(m/2)+1} - z_{m/2})\end{aligned} \tag{7}$$

and the sum of distances from z_i to p is:

$$\begin{aligned}\sum_{i=1}^{m} |z_i - p| = & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}| - (z_{m/2} - p)) + \sum_{i=(m/2)}^{(m/2)+1} (|z_i - p|) \\ & + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}| + (z_{(m/2)+1} - p)) \\ = & \sum_{i=1}^{(m/2)-1} (|z_i - z_{(m/2)}|) + (z_{(m/2)+1} - z_{m/2}) + (z_{m/2} - p) + \sum_{i=(m/2)+2}^{m} (|z_i - z_{(m/2)+1}|) \\ & + (\frac{m}{2} - 1) * (z_{(m/2)+1} - z_{m/2}) \\ = & \sum_{i=1}^{m} |z_i - q| + (z_{m/2} - p),\end{aligned} \tag{8}$$

$\square$

which again contradicts the assumption that p is a point of the prototype.

A consequence of Theorem 4 is that, for any point process dataset, there exists a prototype made up entirely of points observed in the dataset. That is, in searching for a prototype, one may limit one's search to all possible combinations of entries in the dataset.

3.2 Prototype determination

Theorem 4 may be useful in determining the prototype of a point process dataset. In addition, note that each point of the prototype must be within $(p_a + p_d)/p_m$ of a fraction $p_a/(p_a + p_d)$ of the dataset's point patterns, as discussed in Schoenberg and Tranbarger (2004). Let n be the number of point patterns in the dataset. Then each point z in the dataset for which at least $np_a/[2(p_a + p_d)]$ other points from distinct point patterns in the dataset are in the range $[z - (p_a + p_d)/p_m, z]$ and at least $np_a/[2(p_a + p_d)]$ other points from distinct point patterns in the dataset fall in the range $[z, z + (p_a + p_d)/p_m]$ is a candidate for inclusion in the prototype. For the case where $p_a = p_d$, this is simply $n/4$ points from distinct point patterns in the dataset occurring on either side of z, within a distance $2 * p_d/p_m$. In practice, this observation may significantly decrease the number of candidates to be considered as potential points in the prototype.

3.2.1 Direct algorithms for prototype determination

With a finite number n_{cp} of candidate prototype points to consider, one may compute the sum in equation (4) for each possible prototype, i.e. each possible collection of points that are potentially in the prototype. For sufficiently small p_m, the prototype will generally have length equal to the median length M of the n point patterns, so one may limit one's search to the $\binom{n_{cp}}{M}$ collections of length M only, as candidates for the prototype. Depending on the size of the dataset of interest, this may be a reasonable task to undertake.

Alternatively, if p_m is not small enough to allow for a prototype of length M, then the best prototype of length k can be iteratively sought for $k = 1, 2, \ldots, M$.

In this case, the prototype may be found by considering the optimal prototype candidate of length $k = 1$ and progressively increasing k, ending the search once the sum in (4) for length $k + 1$ is greater than the sum (4) for the best prototype of length k.

3.2.2 Forward and reverse stepwise approximation

Finding the prototype of a set of point patterns as prescribed in Section 3.2.1 may prove to be prohibitively tedious for large datasets. In this case, stepwise methods can be implemented to attain an approximate prototype solution.

A reverse stepwise approach can be implemented by first finding the optimal prototype of median length M. Possible prototypes of shorter length can then be found by eliminating points from the length M prototype one at a time. This strategy continues until the optimal prototype of length $k - 1$ has a larger sum of distances (4) than the best prototype candidate of length k.

While the reverse stepwise approach will entail fewer computations than finding the prototype directly, even the reverse stepwise approximation can prove prohibitively cumbersome due to the long length of the prototype computed in the first stage. An alternative is to use a forward stepwise approach, which can further reduce computation time by beginning with a short prototype candidate and progressively expanding it by one point at each step. In each iteration, the sum of pattern-prototype distances (4) for the prototype candidate of length k is compared with the sum for the prototype candidate of length $k-1$.

Figure 4 illustrates the speed of the forward stepwise prototype approximation approach using the MBM algorithm for distance calculations, again using a 1.33GHz PowerPC G4 processor. Here, $N = 5, 10, ..., 50$ Poisson processes of mean length $n = 1, 2, ...10$ were simulated and the prototypes of the N patterns were determined using the forward stepwise algorithm. Mean CPU times were recorded after 100 repetitions at each level of N and n. As seen in Figure 4, the number of point patterns in a dataset and the average length of those patterns have quite different effects on the calculation time required for prototype determination. While the large perspective plot of Figure 4 shows these two factors together, the two smaller plots to the right illustrate the factors' effects on time separately by averaging the results first over the N levels of number of point patterns and secondly for the n levels of pattern mean length. These marginal plots show that while the mean pattern length has a relationship to run time very similar to that seen in Figure 3 for the time required in distance calculation by the MBM algorithm, the relationship between the number of point patterns and run time is nearly linear. This result is not surprising in light of the fact that collections of longer point patterns will have a prototype longer in length than collections of point patterns that are shorter in length. Longer prototypes require additional steps in the stepwise algorithm and the CPU time required for each step increases rapidly as the algorithm continues.

3.3 Penalty selection considerations

Penalties selected for p_m, p_a, and p_d play a significant role in prototype determination. As discussed in Schoenberg and Tranbarger (2004), a point at time t can only be part of the prototype if at least $p_a/(p_a + p_d)$ of the point patterns in the dataset contain a point within the interval $[t - 2p_a/p_m, t + 2p_a/p_m]$. Selection of moving penalties that are too large will lead to prototypes that seem unusually short, since the range $t \pm 2p_a/p_m$ will be quite small.

When the primary purpose of the prototype pattern is to serve as an example of a typical point pattern, it is useful to set p_m to be quite small compared to what might be advisable for distance calculations. Setting p_m to a small positive value when $p_a = p_d$ enables the prototype to take on the median number of points in the dataset. Because each prototype point must be matched to points within at least $p_a/(p_a + p_d)$ of the observed point patterns, it is typically not possible to achieve a prototype of greater than median length while maintaining the $p_a = p_d$ restriction.

Figure 5 illustrates how the length of the prototype is related to the relative size of the selected p_m and $p_a = p_d$ values. For the first plot on the left, prototypes were calculated for 100 simulated datasets, each consisting of 20 stationary Poisson processes of rate 1 on $[0, T]$, using the forward stepwise algorithm and 100 different p_m values ranging from 0.001 to 2.0. In all cases, $p_a = p_d = 1$. The mean prototype length for each penalty value is seen along the y-axis and is plotted against the corresponding movement penalty value. As expected, smaller values of p_m correspond to longer prototype lengths, with a leveling off across very small p_m values where the maximum prototype length (the median) has been achieved.

The plot at right of Figure 5 shows the 100 prototypes calculated for one randomly selected simulated dataset used for the illustration at left. Of particular interest is the observation that while the number of points in the prototype decreases with increases in p_m, the actual location of many of the prototype points remains essentially constant across a wide range of penalty values. This suggests that for any particular dataset, the locations of the points in the prototype will be rather robust to choices of p_m and $p_a = p_d$. Small changes in the ratio p_m/p_a appear to have surprisingly little effect on either the prototype pattern or the overall distribution of distances as seen earlier in Figure 1.

4 CLASSIFICATION THROUGH CLUSTERING

Using the spike-time distance metric and prototype, various clustering algorithms established for standard multivariate data can be modified for use in point process applications. This Section describes how three such methods can be adapted and discusses issues in the selection of penalties when these clustering algorithms are used.

4.1 HMEANS, KMEANS and Agglomerative clustering

HMEANS and KMEANS clustering are two closely related iterative techniques useful for clustering multivariate data. Both HMEANS and KMEANS clustering begin by randomly assigning each observation to one of c clusters and finding the center of each cluster. While cluster centers are conventionally defined as centroids in standard HMEANS and KMEANS clustering of multivariate data (Spaeth 1980, Anderberg 1973, Bock 1970, Howard 1966), in the case of a point process dataset one may instead use the prototype as the definition of the center of a cluster of point patterns.

A third option is an agglomerative approach that assigns each observed point pattern to its own cluster and then systematically combines clusters. This approach is especially useful in cases where HMEANS and KMEANS fail to converge after numerous iterations.

With each observed point pattern assigned to a cluster of its own, each cluster's prototype is simply the member pattern. The pairwise distances between each cluster prototype can be found and the nearest two clusters joined to form one larger two-pattern cluster. From here, the new cluster prototype is determined, and the new pairwise distances between prototypes are calculated. The process repeats until only c desired clusters remain. If the amount of data makes computing all pairwise distances unfeasible, a similar approach is to consider pairwise distances for only one point pattern at a time in either a randomly assigned order, or in order of length from the longest pattern to the shortest. Our investigations suggest that this procedure of ordering the point patterns from longest to shortest and then progressively merging the point patterns minimizes problems that can occur if the p_a, p_d, and p_m penalties are set such that pattern length overshadows other features in the distance measures.

4.2 Penalty choice considerations

While setting p_m to a small value is useful in determining prototypes, very small p_m values are not desirable in clustering applications. As discussed in Section 2.4, small penalties for movement of points will lead to distances that approach the absolute difference in pattern lengths, while large movement penalties lead to distance calculations that approach the sum of point pattern lengths. In clustering, either extreme will lead to clusters assigned by grouping patterns of similar lengths rather than patterns with similarly placed points.

For the simulation presented in Figure 6, ten point patterns were created for each of two different methods. Ten realizations of a rate 1 Poisson process on [0,6] while another ten point patterns were each created by sampling n points independently from a normal random variable with mean 3.0 and standard deviation 0.25, where n is a Poisson random variable with mean 6.0. Patterns of length zero were excluded. Plotted in Figure 6 is the mean failure rate for HMEANS in correctly identifying the two different types of patterns over 125 simulations. The mean rate of failure is shown for twenty different p_m values ranging from 5% to 200% of the penalty recommended in (3) for distance calculations. It is clear in these results that extremely small values of p_m are less successful at clustering than values similar to that advised in (3) for distance measurements.

5 MULTI-DIMENSIONAL EXTENSIONS AND DISCUSSION

The spike-time distance metric as examined in this work, and as originally proposed by Victor and Pupura (1997), has thus far been applied primarily to temporal point patterns as defined in Section 2. As explored in Schoenberg and Tranbarger (2004), the spike-time metric, the related prototype pattern technique, and related clustering algorithms can be extended to point process data with points occurring in $\mathbf{R}^d$.

In the extension of the definition of the spike-time distance metric to $\mathbf{R}^d$, points can be added with penalty p_a, deleted with penalty p_d, or moved along the i^{th} axis a distance of Δ at a cost of $p_m^{(i)}\Delta$. Moving penalties $p_m^{(1)}, p_m^{(2)}, ...p_m^{(d)}$ may be set independently for movement along each of the d axes as in (3) for distance calculations and clustering application settings. As with the one-dimensional case, smaller values for movement penalties are again useful for prototype determination to enable longer length prototype point patterns. While the ratio of p_a and p_d to moving penalty p_m was of primary importance for temporal point process work, the relative values of the d movement penalties must also be considered when multiple dimensions are involved. These ratios must be considered so as to avoid (or allow) inter-point distances along one or more dimensions being more heavily weighted in distance calculations.

As might be expected, spike-time distance calculations are far more cumbersome in $\mathbf{R}^d$, as the result of Theorem 1 does not hold. Without the result of Theorem 1, neither the SU nor MSU approaches can be implemented. Also, using MBM, multiple pairing arrangements must be considered for each possible set of kept points. The process to determine which points will be kept and which will be removed under the MBM algorithm remains unchanged as Lemma 2 and Theorem 3 (and their results) extend immediately to multiple dimensions.

For prototype pattern determination, a modified version of Theorem 4 applies to the multi-dimensional setting. With more than one dimension, rather than each prototype point p being a median of the points $(z_1, z_2, ..., z_m)$ it is paired with, each *coordinate* of prototype point p will be a median of the corresponding coordinate of the points $(z_1, z_2, ..., z_m)$. Therefore, while the prototype may not contain points in the dataset, there exists a prototype made entirely of points such that each coordinate of each prototype point is a coordinate in one of the points in the dataset.

Clustering algorithms discussed in Section 4 can be applied to multi-dimensional point pattern data without modification. For some datasets, the computation time involved in prototype determination impedes the use of the clustering algorithms as described in Section 4 and slight modifications can be made, such as considering only one of the d dimensions at a time in each step of the prototype and/or distance calculation. Such a monothetic approach is used in Schoenberg and Tranbarger (2004) for clustering earthquake aftershock activity considering the time, magnitude, and location of each aftershock.

While the spike-time distance metric is only one of countless distance metrics for point pattern data [see Victor and Purpura (1997) for others], the concept of a prototype sequence is one that exists independently of distance metric specifics. Therefore, though many of the results presented here apply solely to the spike-time metric, the approaches to determining prototypes and clusters derived from the ability to measure distances between point patterns should remain useful in conjunction with other point pattern distance metrics.

6 ACKNOWLEDGEMENTS

We thank Chuck Woody for introducing us to the work of Victor and Purpura, and we thank Pepe Segundo for introducing us to Chuck Woody. This material is based upon work supported by the National Science Foundation under Grant No. 0306526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7 REFERENCES

Aronov, D. (2003). Fast algorithm for the metric-space analysis of simultaneous responses of multiple single neurons. *Journal of Neuroscience Methods* **124**, 175-179.

Daley, D., and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes, Volume 1: Elementary Theory and Methods, 2nd ed.* Springer-Verlag, New York.

Hitchcock, F.L. (1941). The distribution of a product from several sources to numerous localities. *Journal of Mathematical Physics*, **20**, 224–230.

Levina, E. and Bickel, P.J. (2001). *The Earth Mover's Distance is the Mallows Distance: Some Insights from Statistics.* Proceedings of ICCV 2001, Vancouver, Canada, 251-256.

Monge, G. (1781). *Mémoire sur la Théorie des Déblais et des Remblais.* Histoire de l'Académie Royale des Sciences, Paris.

Reich D., Mechler, F., Purpura, K., and Victor, J. (2000). Interspike intervals, receptive fields, and information encoding in primary visual cortex. *Journal of Neuroscience* **20**(5), 1964-1974.

Rubner, Y., Tomasi, C., and Guibas, L. (1998). A Metric for Distributions with Applications to Image Databases. *IEEE International Conference on Computer Vision*, 59-66.

Rubner, Y., Tomasi, C., and Guibas, L. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* **40**(2), 99-121.

Schoenberg, F.P. and Tranbarger, K.E. (2004). Description of earthquake aftershock sequences using prototype point processes. *In Review*

Sedgewick, R. (1986). A new upper bound for Shell sort. *Journal of Algorithms* **7**, 159-173.

Späth, Helmuth (1980). *Cluster Analysis Algorithms for Data Reduction and Classification of Objects.* E. Horwood, Chichester and Halsted Press, New York.

Victor, J. and Purpura, K. (1997). Metric-space analysis of spike trains: theory, algorithms and application. *Network: Computation in Neural Systems* **8**, 127–164.

Figure 1

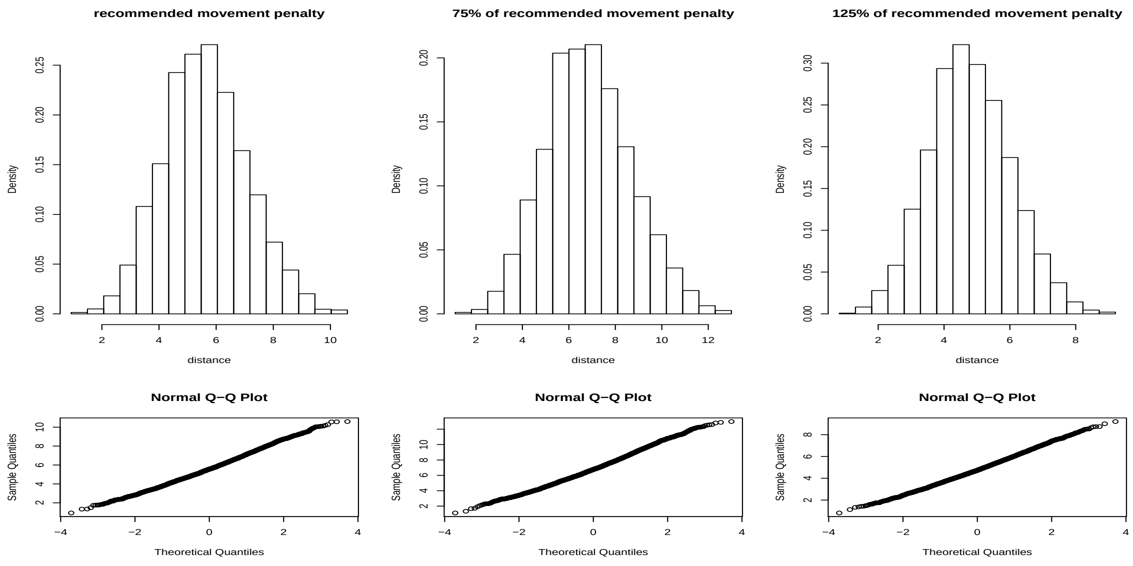


Figure 1: Pairwise distances for 100 simulated Poisson point patterns using three different penalty values. Of note is the observation that the general shape of the distribution of distances remains normal for all three penalty values.

Figure 2

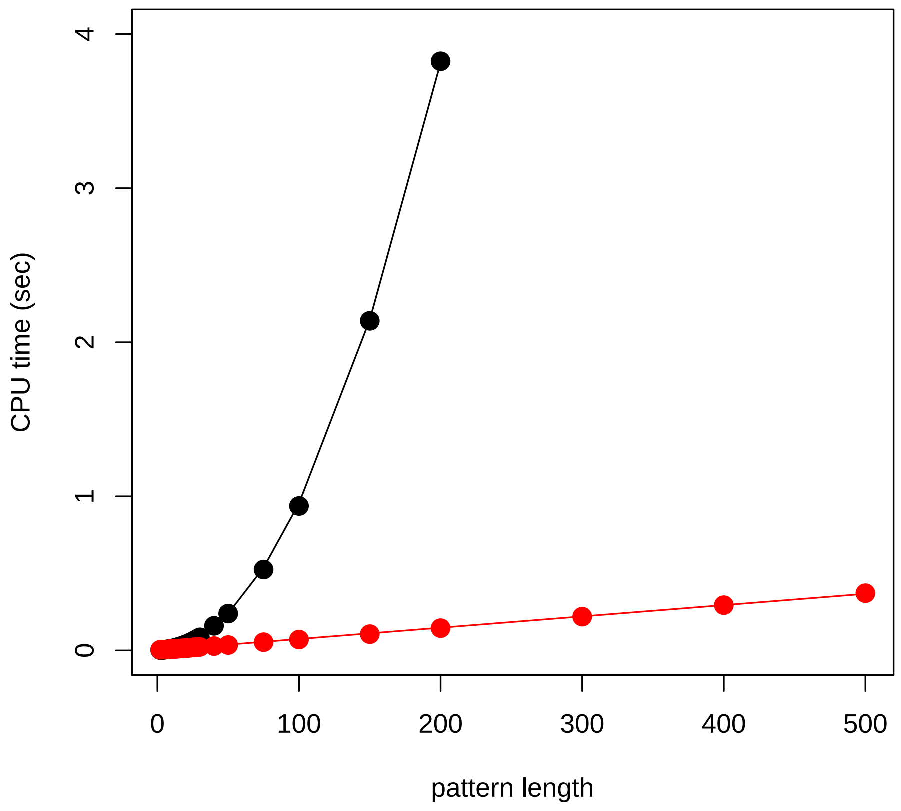


Figure 2: Mean calculation time in R for spike-time distance as pattern length increases. The SU algorithm (in black) is shown with the best-fit n^2 curve, and the MSU algorithm (in red) is shown with a linear fit.

Figure 3

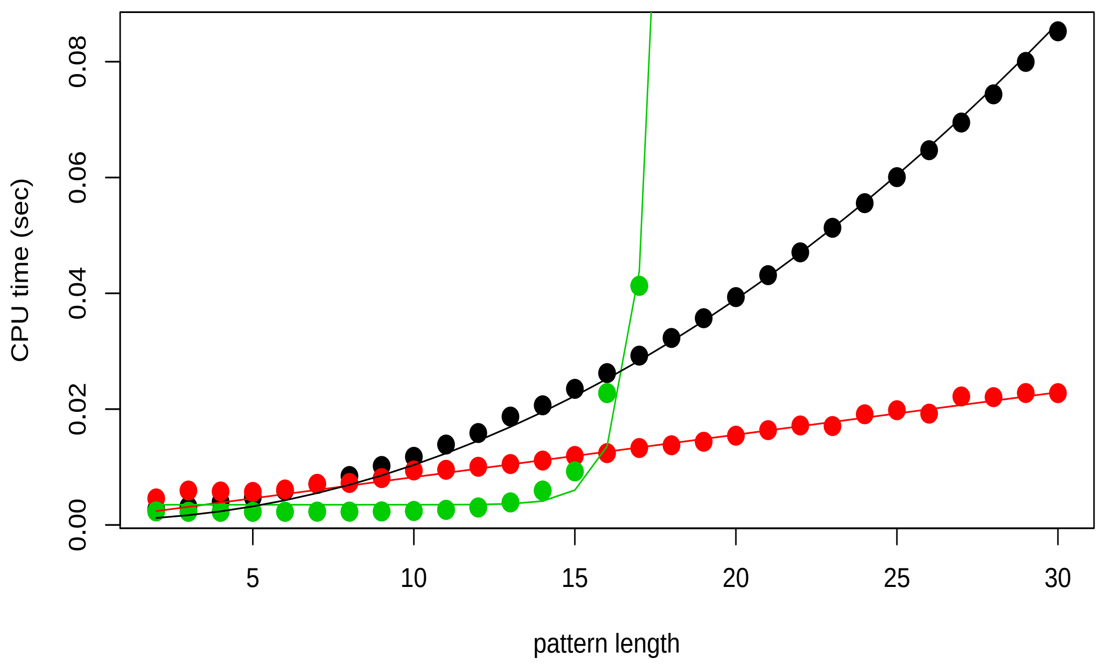
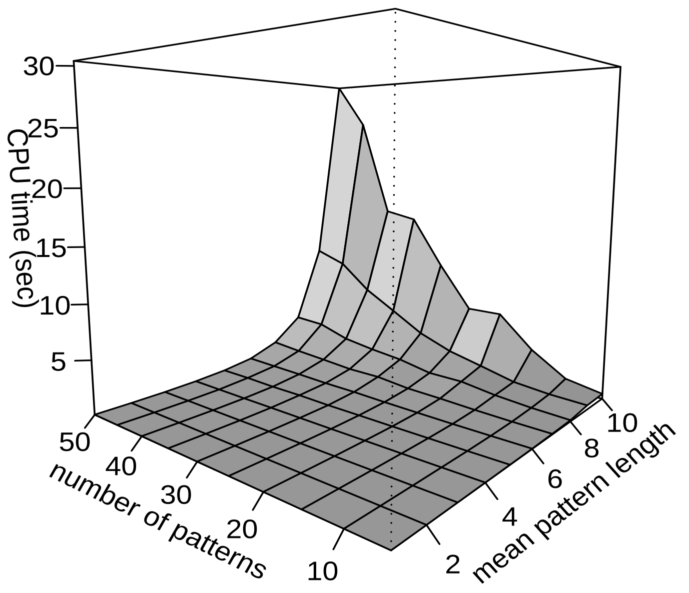
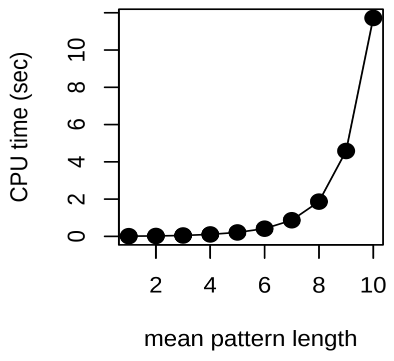


Figure 3: Mean calculation time for spike-time distance as pattern length increases. The MBM distance algorithm (in C) shown in green proves to compute distances more quickly than both the SU and MSU algorithms (in R) in cases where the mean pattern length is less than 16 points (when both MBM and SU are coded in C, MBM performs better for any pattern length less than 9). For patterns of longer lengths, the MSU algorithm is shown to calculate distances most efficiently. In general, the MBM algorithm simulations follow a 4^n curve, the SU algorithm (shown in black) follows a n^2 curve, and the MSU algorithm (in red) is linear with increases in pattern length.

Figure 4





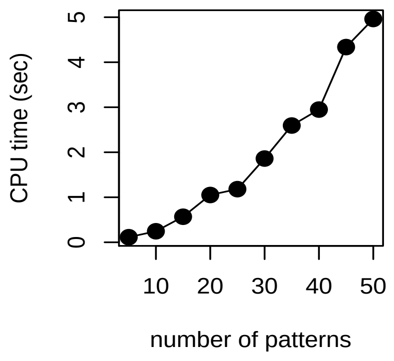


Figure 4: CPU time for forward stepwise prototype approximation of simulated Poisson point patterns of varying length and number. Distances within the prototype algorithm were computed using the MBM algorithm.

Figure 5

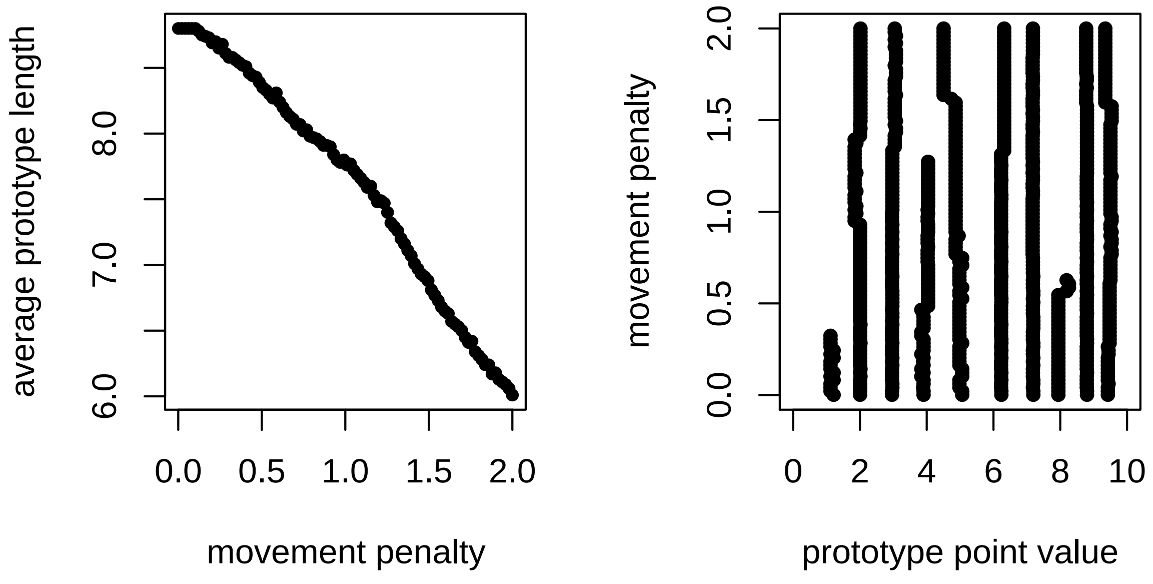


Figure 5: On the left it is seen that as movement penalty values increase, the resulting prototype pattern decreases in length. The graph at right shows Changes in the movement penalty have little influence over location of prototype points. While length decreases with increasing movement penalties, the general location of prototype points is largely consistent across all reasonable penalty values. Plotted are the prototype points for a simulated dataset of 20 Poisson patterns of rate 1 ranging in time from 0 to 10 computed using $p_a = p_d = 1$ and $0.01 \le p_m \le 2$ as plotted on the y-axis.

Figure 6

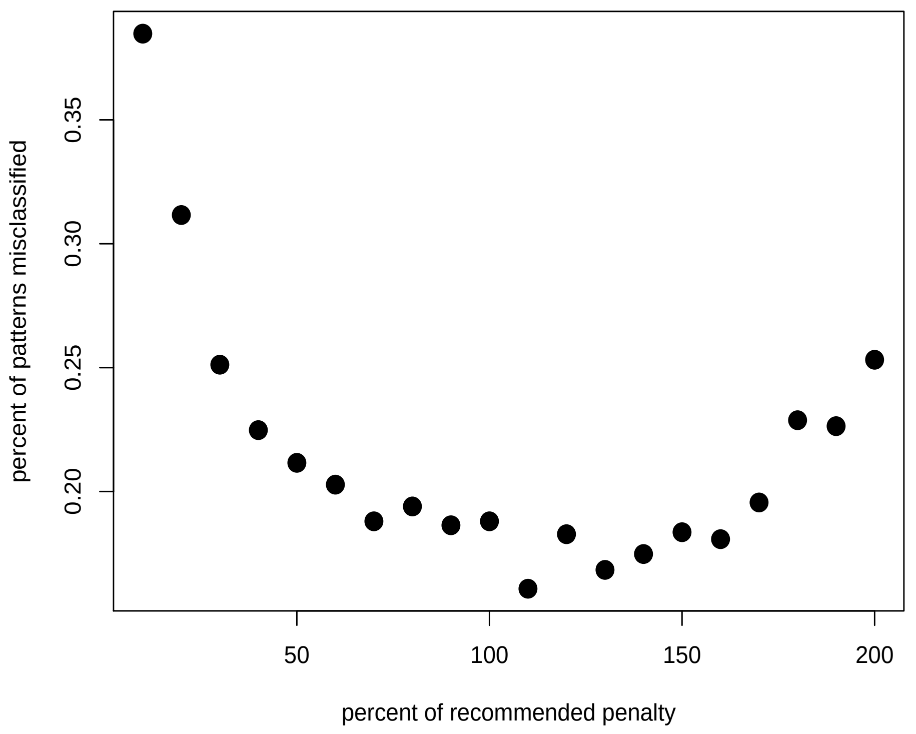


Figure 6: For this simulation (repeated 125 times) ten patterns were created following a Poisson process model of rate 1 with a mean length of six points and ten patterns were created by randomly sampling from a Normal(3, .25) distribution of a size determined by a Poisson(6) random variable. As shown, extremely small penalty values cause HMEANS to incorrectly classify patterns more frequently than penalty values closer to that recommended in (3) of Section 2.4.