On distances between point patterns and their applications

Jorge Mateu∗, Frederic P Schoenberg† and David M Diez‡

Abstract

Various classes of distance metrics between point patterns are outlined, and their uses and pros and cons are discussed. Examples include spike-time distance and its variants, proposed by Victor and Purpura (1997), as well as cluster-based distances and distances based on classical statistical summaries of point patterns, such as the $K$-function or LISA functions. Applications to the summary and description of collections of repeated realizations of a point process via prototypes or multidimensional scaling are explored.

Keywords: $K$-function, LISA functions, point processes, Poisson processes, prototypes, spike-time distance

1 Introduction

A point pattern is a collection of points falling in some space. A random point pattern, that is, a process which assigns different likelihoods to different point patterns, is called a point process. Point process data arise in a wide variety of applications, including epidemiology, ecology, forestry, mining, hydrology, astronomy, ecology, and meteorology.

∗Department of Mathematics, University Jaume I, E-12071 Castellon, Spain. mateu@mat.uji.es
†Department of Statistics, UCLA, 8125 Math-Sciences, Los Angeles, CA 90095, USA. frederic@stat.ucla.edu
‡Department of Statistics, UCLA, 8125 Math-Sciences, Los Angeles, CA 90095, USA. david.m.diez@gmail.com
Point processes evolved naturally from renewal theory and the statistical analysis of life tables dating back to the 17th century, and in the earliest applications, each point represented the occurrence time of an event, such as death or incidence of disease; see Daley and Vere-Jones (2003) for a review and a thorough treatment of point processes. In the mid-20th century, interest spanned to spatial point processes, where each point represented the location of some object or event, such as a tree or sighting of a species. Models for spatial point processes (see e.g. Cressie (1993)) grew quite intricate, and among the names associated with these models are some of the key names in the history of statistics, including Jersey Neyman and David Cox. Today, much attention is paid to space-time point processes, where each point represents the time and location of an event, such as the origination of an earthquake or wildfire, a lightning strike, or an incidence of a particular disease.

Note that point processes are intimately related to time series. As explained in Schoenberg (2011), some data sets that are traditionally viewed as realizations of point processes could in principle also be regarded as time series, and vice versa. For instance, a sequence of earthquake origin times is typically viewed as a temporal point process, though one could also store such a sequence as a time series consisting of zeros and ones, with the ones representing earthquakes. The main difference is that for a point process, the points can occur at any times in a continuum, whereas for time series, the time intervals are discretized. In addition, if the points are sufficiently sparse, one can see that it may be far more practical to store and analyze the data as a point process, rather than deal with a long list containing mostly zeros.

By the mid 1990s, models for spatial-temporal point processes had become plentiful and often quite intricate. Around this time, a wealth of neuronal spike train data began to become available to neuroscientists. Unlike data from seismology or epidemiology, for
instance, where the events were naturally seen as one long instance of a point process, the neuronal data typically consisted of many repeated observations of a point process. For instance, one might observe the times of firings of neurons in a particular part of the brain in the instant immediately following a stimulus, for many different subjects, some of whom have a disease and some of whom do not. The analysis of this type of data required some slightly different methods.

In order to classify neuronal spike trains into clusters or to differentiate between diseased and healthy patients based on their neuronal firing patterns, methods were sought which would require the definition of a distance between two point patterns. The seminal work of Victor and Purpura (1997) proposed several distance metrics, including spike-time distance, which the authors used for describing neuronal spike trains. The list of distances in Victor and Purpura (1997) is far from exhaustive, however, and certain alternative types of distances are more useful for clustered or inhomogeneous point processes or for point processes in high-dimensional spaces.

The purpose of this article is to outline several types of distances between two point patterns, $X$ and $Y$, each observed on the same metric space $S$. We also discuss some examples and applications of the proposed distance metrics to cluster analysis and prototype determination.

The paper is outlined as follows. In Section 2, we discuss spike-time distance and its variants which involve matching individual points of $X$ to corresponding points in $Y$. Section 3 describes distances that might be especially useful for clustered point processes. Distances based on classical functional or scalar descriptors of point processes, including estimators of first and second moments of the processes, or classical test statistics based on these moments, are described in Sections 4 and 5. Applications to summaries of collections of independent realizations of a point process via prototypes or multidimensional scaling
are given in Section 6.

## 2 Pointwise distances

One family of distances is characterized by a raw transformation of one pattern $X$ into another pattern $Y$ by actions on individual points. Individual points in $X$ are moved so that the resulting layout resembles $Y$.

The **spike-time distance** metric developed by Victor and Purpura (1997) is one popular distance metric that has been applied to neuronal, earthquake, and wildfire data (Victor and Purpura, 1997; Tranbarger and Schoenberg, 2010; Schoenberg and Tranbarger, 2008; Diez et al., 2010a; Nichols et al., 2010). Techniques for computing this distance metric were only recently extended to multiple dimensions (Diez et al., 2010a,b). The distance is defined as the minimum cost associated with the transformation of one point pattern $X$ into a pattern $Y$ by deleting, adding, and moving points. The left panel of Fig. 1 represents a transformation of $X$ into $Y$ via actions on individual points of $X$. Those points in $X$ not moved are deleted from $X$, and those points in $Y$ to which no point in $X$ corresponds are added to $X$. If we describe such a transformation as $T$, then the cost associated with $T$ is

$$C(T|X,Y) = p_d|X_{\text{delete}}| + p_a|Y_{\text{add}}| + \sum_{x \in X_{\text{move}}} p_m d_x$$

where $p_a$, $p_d$, and $p_m$ are penalty parameters, $X_{\text{delete}}$, $Y_{\text{add}}$, and $X_{\text{move}}$ are subsets of $X$ and $Y$ that are deleted, added, and moved, respectively; $d_x$ is the distance a point $x$ in $X_{\text{move}}$ is moved, and $|S|$ represents the number of points in a set $S$. The spike-time distance, $d_s(X,Y)$, is defined as the infimum of equation (1) over all such possible transformations $T$. The transformation shown in the left panel of Fig. 1 is the transformation minimizing (1) for a particular set of parameters. Victor and Purpura (1997) showed that spike-
time distance is a well-defined distance metric, and they also considered other closely related distances. Spike-time distance in single or multiple dimensions may be computed using the free R statistical software via the \texttt{stDist} function in the \texttt{ppMeasures} package (R Development Core Team, 2010; Diez et al., 2010b). The C code forming the base of the package is available through the package page on the R Project website, and a brief introduction to \texttt{stDist} function is provided in the Appendix. A spike analysis toolkit for single-dimensional analyses was also made available for MATLAB and Octave (Goldberg et al., 2009).

Another useful distance function that is also defined in terms of such transformations of $X$ into $Y$ is the \textbf{nearest-point distance}, where each point $x$ in $X$ is moved to its nearest neighbor in $Y$; call this nearest point $y_x$. Unlike spike-time distance, nearest-point distances are computed without allowing the addition or deletion of points. The nearest-point distance is defined simply as

$$d_n(X,Y) = \sum_{x \in X} ||x - y_x||$$

where $|| \cdot ||$ may represent Euclidean distance for point patterns in $\mathbb{R}^k$, or some other distance metric for point patterns in a more general metric space. The measure $d_n$ is useful in facility placement. For instance, if points in $X$ represent consumers and points in $Y$ represent facilities, then $d_n$ characterizes the total cost for all consumers to visit their closest facility. Note that for two distinct points $x, x'$ in $X$, we may have $y_x \equiv y_{x'}$, and that in general, $d_n(X,Y) \neq d_n(Y,X)$, so that $d_n$ is not formally a distance metric. However, the sum of these two distances forms a symmetric nearest-point distance metric:

$$d_N(X,Y) = d_n(X,Y) + d_n(Y,X).$$
3 Distances for cluster processes

When a point process is highly clustered, metrics such as spike-time distance tend to yield prototypes that do not reflect a typical sample point pattern. Clustered processes thus require a separate family of distances where movements of collections of points are permitted. For example, consider the following metric, called cluster distance. Let \( T \) represent a transformation of \( X \) into \( Y \) that involves sequentially moving collections of points in \( X \). Thus, \( T \) is itself a sequence of transformations, \( t_i \), where each \( t_i \) moves a subset \( X_i \) of points by a vector \( z_i \) (see the first two panels of Fig. 2). Then the cost associated with \( T \) is defined as

\[
p_d |X_{\text{delete}}| + p_a |Y_{\text{add}}| + \sum_i p_m |X_i|^q ||z_i||
\]

where \( q \) is a parameter in \([0, 1)\). The cluster distance, \( d_c(X, Y) \), is the infimum cost over all such transformations, \( T \). The further each cluster must be moved (the larger \( z_i \) is), and the more points that must be moved in order to match \( X \) up with \( Y \) (the larger \(|X_i|\) is), the larger the cluster distance between \( X \) and \( Y \).
A similar type of distance for clustered point processes is defined by first aligning clusters in $X$ with clusters in $Y$ and subsequently applying simple spike-time distance. For instance, let $\{R_j\}$ represent a set of disjoint and concave regions of the space that contain all the points of $X$. One may translate all of the points in a given region by some fixed vector and repeat for each region, assigning a cost of $p_c$ per unit distance to each translated region, and then spike-time distance may be applied after these regions are moved. This **declustered spike-time distance** is defined as the minimum cost over all such transformations and choices of $\{R_j\}$. An illustration is given in the right panel of Fig. 2.

![Figure 2: The first two steps in the cluster distance are shown in the left and center panels. The right panel represents the process of removing the clustering characteristics prior to applying spike-time distance.](image)

A drawback of both cluster distance and declustered spike-time distance is that methods for their efficient computation appear to be unavailable presently, and searching over all possible translations of all possible subsets of points in $X$ is extremely computationally burdensome. An advantage of cluster distance is that the clusters $X_i$ and their associated translation vectors, $z_i$, may have physical meaning and be of direct interest. For instance, a subset of point pattern $X$ is nearly equivalent to a subset of point pattern $Y$ in the case of a temporal delay in response to stimuli, in the case where the points are observed in time, and in the purely spatial case, the points in $X$ or $Y$ may be translated due
to physical deformation or may require spatial translation due to mis-calibration of the recording instruments.

4 Distances based on functional summaries

4.1 Model-based distances

One way to construct a statistical model (in any field of statistics) is to write down its probability density. One advantages of doing this is that the functional form of the density reflects its probabilistic properties, and the terms or factors in the density often have an interpretation as components of the model. This approach is useful provided the density can be written down, and provided the density is tractable. Probability densities for point processes behave much like probability densities in more familiar contexts. Spatial point process models that are constructed by writing down their probability densities are of interest to the statistical community, and a widely known and used such class of models is called Gibbs processes. To construct spatial point processes which exhibit interpoint interaction (stochastic dependence between points), we need to introduce terms in the density that depend on more than one point.

Each point process model is specified in terms of its conditional intensity rather than its likelihood. This turns out to be an intuitively appealing way to formulate point process models, as well as being necessary for technical reasons. For spatial point processes, the lack of a natural ordering in two-dimensional space implies that there is no natural generalization of the conditional intensity of a temporal or spatiotemporal process given the past or history up to time $t$. Instead, the appropriate counterpart for a spatial point process is the Papangelou conditional intensity (Daley and Vere-Jones (2003)), say $\lambda(u, X)$, which conditions on the outcome of the process at all spatial locations other than $u$. Infor-
mally, \( \lambda(u, X) du \) gives the conditional probability of finding a point of the process inside an infinitesimal neighborhood of the location \( u \), given the complete point pattern at all other locations. The Papangelou conditional intensity of a finite point process uniquely determines its probability density and vice-versa.

Given specific models for the point processes giving rise to the point patterns \( X \) and \( Y \), one may define the distance between \( X \) and \( Y \) in terms of the differences between characteristics of these models. For instance, if the point processes \( X \) and \( Y \) are characterized by their conditional intensities \( \lambda_X(x) \) and \( \lambda_Y(x) \), respectively, then one measure of the difference in these point process models is the integral of the squared difference of the two conditional intensities over the observation region \( S \):

\[
d_{MB}(X, Y) = \int_S (\lambda_X(x) - \lambda_Y(x))^2 dx. \tag{3}
\]

This is illustrated in the left panel of Fig. 3, where the intensity functions have been estimated based on the data shown at the top of the panel. These methods readily extend to a variety of other summaries, such as the integrated squared difference between the overall mean, or second moment measure, or higher moments or cumulants of the processes. Of course, the conditional intensities in (3) may be replaced by their expected values, their overall intensities, or in the spatial point process setting, by the Papangelou intensities (see e.g. Daley and Vere-Jones (2003)).

### 4.2 Distances based on classical functional statistical summaries

Differences in point process behavior can also be characterized by comparing classical summary measures of the first or second moments for each of the point patterns. For instance, given two point patterns \( X \) and \( Y \), one could imagine looking at an estimate of the overall intensity of \( X \), e.g. a kernel estimate, and a similar estimate of the intensity
of $Y$. Alternatively, if each of the point patterns is one-dimensional, then one could look at the empirical cumulative distribution function for each point pattern as a statistical summary of the realization.

In his influential paper Ripley (1977) developed an exploratory analysis of interpoint interaction, assuming that the data are spatially homogeneous. A useful summary statistic is the nonparametric estimator of Ripley’s $K$-function, essentially a renormalized empirical distribution of the pairwise distances between observed points. More pragmatically, for a stationary point process with intensity (mean number of points per unit area) $\lambda$, $\lambda K(r)$ gives the expected number of other points of the process within a distance $r$ of a typical point of the process.

Thus, a further alternative would be to take the estimated $K$-function or its derivative, the estimated reduced 2nd moment measure, for each pattern, and examine the difference. As an illustration, one could compare the integrated squared difference between estimates of Ripley’s $K$-function for two point patterns:

$$\int (\hat{K}_1(r) - \hat{K}_2(r))^2 dr$$

This distance is illustrated in the right panel of Fig. 3.

First and second order measures represent a valuable description of a spatial point process but do not generally uniquely characterize a point process. Additional important summary descriptions are given by distance methods based on measures of some distances between points. The nearest neighbor distance $D$ may be defined as the distance from a point of the pattern to the closest of the other points in the same pattern. The empty space distance $F$ is the distance from an arbitrary fixed location to the nearest point of the pattern. For a given point pattern, a set of distances (nearest neighbor or empty space distance) can be summarized by means of their corresponding distribution functions. Let
Figure 3: Left: A comparison of two intensity models using Equation (3). In this illustration, the intensities $\lambda_X$ and $\lambda_Y$ are estimated by kernel smoothing the points in $X$ and $Y$, respectively. Right: an example using Equation (4) to characterize the difference in clustering behavior of two patterns using Ripley’s K function.

$G(r)$ denote the nearest-neighbor distribution function, and $F(r)$ the corresponding first contact distribution function or empty space function. These distribution functions may be estimated from the observed point patterns $X$ and $Y$ using conventional methods. For example, if $d_i$ define the distance from each of $m$ points in a regular lattice arrangement to the nearest event, then \( \hat{F}(r) = m^{-1} \sum I(d_i \leq r) \), where $I(\cdot)$ is the indicator function. Similarly, if $e_i$ is the distance from each of $n$ events of the point pattern to its nearest neighbour, then \( \hat{G}(r) = n^{-1} \sum I(e_i \leq r) \).

In practice, each point pattern is typically observed in a bounded region $S$, and without precautions, these boundaries can lead to biased estimates. This problem is known in the context of spatial statistics as edge-effects. Different edge-corrected estimators of the functions $F$ and $G$ have been proposed (see e.g. Cressie (1993)). There is a clear analogy with censoring in the context of survival analysis Baddeley and Gill (1997). The different distances observed within a single point pattern are really censored distances. Let $d_i$ denote the observed distance from the $ith$ point of the point pattern to its nearest neighbor within...
the sampling window, \( W \), and \( c_i \) its distance to the complement of the sampling window \( W \). If \( c_i < d_i \) the real nearest neighbor could be outside the window and, in this case, the real and unknown nearest neighbor distance \( d_i \) fulfills \( d_i > c_i \), and the observation is censored. A similar comment applies to empty space distances. In both cases the censoring distance is the distance from the sampling point to the frame window.

Given a statistical summary such as the estimated \( F \) or \( G \)-function of the point processes, the dissimilarity between point processes \( X \) and \( Y \) can be defined as the distance between the corresponding estimated functional summaries. For instance, let \( \hat{F}_X \) and \( \hat{F}_Y \) be the corresponding estimated empty-space distribution functions for \( X \) and \( Y \). The dissimilarity between \( X \) and \( Y \) can thus be defined as

\[
D(X, Y) = d(\hat{F}_X, \hat{F}_Y)
\]

where \( d \) stands for a metric between the functions. For instance, one may use the \( L_2 \) metric

\[
\begin{align*}
D_F^2(X, Y) &= \int_0^r (\hat{F}_X(r) - \hat{F}_Y(r))^2 \, dr
\end{align*}
\]

or the \( L_\infty \) metric:

\[
D_F^\infty(X, Y) = \sup_r \| \hat{F}_X(r) - \hat{F}_Y(r) \|
\]

By replacing the empty-space distribution function with the nearest-neighbor distribution function \( G \) or \( K \)-function \( K \), one can similarly define \( D_G^2(X, Y) \), \( D_G^\infty(X, Y) \), \( D_K^2(X, Y) \), and \( D_K^\infty(X, Y) \), respectively. Note that sampling variability in estimates of \( K(r) \) tend to increase with \( r \) and this can have a great influence on the value of the dissimilarity measure. This suggests the use of the \( L \)-function proposed by Besag (1977), a variance stabilized version of the \( K \)-function.
4.3 Distances based on LISA functions

When estimated from point process data, the empirical product density function provides a description of the density of inter-event distances in an observed point pattern. For instance, high values for small distances are indicative of an overabundance of short inter-event distances (this is a typical situation for cluster processes, where data tend to form groups). Conversely, if short inter-event distances are rare, this will indicate that an inhibitory structure is present, and points tend to separate from each other.

Both the $K$-function and the product density function provide a global measure of the covariance structure by summing over the contributions from each event observed in the process. Now we consider individual contributions to the estimated function that are analogous to the local statistics described by Anselin (1995) and called local indicators of spatial association (LISA). An individual LISA product density function for a given point pattern $X$, $l^X_i(\cdot)$, should reveal the extent of the contribution of the $i$–th event to the global estimate of the product density $l^X(\cdot)$, and may provide a further description of structure in the data (e.g., determining events with similar local structure through dissimilarity measures of the individual LISA functions). A product density LISA function can be constructed in the same manner as the global estimate (Cressie and Collins, 2001).

Given point patterns $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, one may calculate the set of LISA functions for each pattern corresponding to any individual or spatial location. Denote both sets by $\{l^X_i(r), i = 1, \ldots, n\}$ and $\{l^Y_j(r), j = 1, \ldots, m\}$. Then one may derive several possibilities for distances between $X$ and $Y$, such as
(a) Define

$$A_X^L = \frac{1}{n \times (n - 1)} \sum_i \sum_j d(l_i^X(r), l_j^X(r))$$

$$= \frac{1}{n \times (n - 1)} \sum_i \sum_j \int (l_i^X(r) - l_j^X(r))^2 dr$$

(8)

as the average of all pairwise distances between LISA functions of points in $X$. Define the same average for points in $Y$, say, $A_Y^L$. Then, a similarity or dissimilarity measure is given by

$$d_L(X, Y) = A_X^L - A_Y^L.$$ 

(9)

(b) Define the average LISA function coming from the set of LISA functions for points in $X$ and $Y$. Denote them by $l^X(r)$ and $l^Y(r)$. Note that $l^X(r)$ is a function itself. Then one may define a distance measure via

$$d_l(X, Y) = d(l^X(r), l^Y(r)) = \int (l^X(r) - l^Y(r))^2 dr.$$ 

(10)

4.4 The proximity function of an individual to a population based on LISA distances

Let $d_X(i, j) = d(l_i^X(r), l_j^X(r)), i, j = 1, \ldots, n$ the set of pairwise distances between LISA functions for points in $X$. Following Cuadras et al. (1997), define the geometric variability for pattern $X$ as

$$V_{d_X} = \frac{1}{2n^2} \sum_{i, j} d^2_X(i, j)$$ 

(11)
Then the *proximity function* for the \( i - \)th point in \( X_i \) is given by

\[
\Phi^2_X(i) = \frac{1}{n} \sum_{j=1}^{n} d^2_X(i,j) - V_{d_X}
\]  

(12)

Hence the value of the associated proximity-based density function for the \( i - \)th point in \( X \) is given by

\[
f_X(i) = \exp\left\{-\frac{1}{2} \Phi^2_X(i)\right\}
\]  

(13)

One may proceed similarly with pattern \( Y \) defining \( V_{d_Y}, \Phi^2_Y(j) \) and \( f_Y(j) \) for \( j = 1, \ldots, m \).

Finally, the distance between patterns \( X \) and \( Y \) may be defined as a measure of similarity/dissimilarity between the densities \( f_X \) and \( f_Y \), as \( d_P(X,Y) = d(f_X, f_Y) \).

### 5 Distances based on numerical test statistics

One could alternatively compare point processes \( X \) and \( Y \) based on single numerical summaries of each point process. For instance, given the same parametric model for the two processes, one could obtain estimates \( \theta_X \) and \( \theta_Y \) of the parameter vector governing each of the two processes, given the realizations \( X \) and \( Y \), respectively. The distance between \( X \) and \( Y \) could then be defined simply as \( ||\theta_X - \theta_Y|| \).

A related idea is to describe the similarity between \( X \) and \( Y \) in terms of a numerical summary of the degree of clustering or inhomogeneity in each. An example would be a generalization to the point process context of the log rank test (Mantel and Haenszel, 1959) or Kolmogorov-Smirnov type statistic (Fleming and Harrington, 1981), where the time or duration usual in the context of survival analysis is replaced here by the distance observed.
For instance, consider the following generalization of the log rank test of Mantel and Haenszel (1959). In survival analysis, the time instant where an event is observed is frequently called (observed) failure time. Here this phrase is replaced by (observed) failure distance, i.e., the distance between a point of the pattern and its nearest neighbor or the distance between a sampling point and the nearest event of the point pattern, provided that these distances are smaller than the distances to the boundary of the window. In order to construct the log rank test, a sequence of $2 \times 2$ tables is built over the distances (one for each failure distance, $d_j$), where the risk set at that distance is classified into a $2 \times 2$ table, according to group and event status. Let us present some notations: $r_j$ stands for the size of the risk set in the interval $[d_{j-1}, d_j)$ and $f_j$ the number of failures. The size of the risk and failure sets for each group can also be defined. In this way, for the first group, they are denoted by $r_{1j}$ and $f_{1j}$ respectively. Differences between the observed and the expected events (conditioning on the margins of the $2 \times 2$ table) on the first group are added up over distances and squared in order to calculate the numerator of the log rank statistic:

$$U = \sum_j (f_{1j} - g_{1j}), \quad (14)$$

where $g_{1j}$ stands for the expected number of failures in the first group, $g_{1j} = f_j \cdot r_{1j}/r_j$. The denominator is the sum of the variances of the number of events on the first group within each $2 \times 2$ table, which is obtained by using the hyper-geometric distribution:

$$Var(U) = \sum_j t_{1j} \cdot f_j/r_j(1 - f_j/r_j)(r_j - r_{1j})/(r_j - 1) \quad (15)$$

The null distribution of $U^2/Var(U)$ is approximately chi-squared with one degree of freedom. The log rank test is more sensitive to differences at the right tails of the two survival distributions and more suitable for detecting departures when hazards are proportional.
There are cases in which this test is not very effective, and one may use a Kolmogorov-Smirnov type statistic, i.e. the supremum of appropriately scaled empirical processes. This test is sensitive to differences that are large at a particular point. A detailed explanation can be found in Fleming and Harrington (1981).

6 Applying measures to a collection of patterns

Distance measures can be used in the identification of prototypical patterns or with multidimensional scaling for classification. Each technique is considered in the context of a collection of patterns that come from either one or many point processes, and an introduction to referenced R packages is included in the Appendix.

6.1 Prototypical patterns

One application of distance measures is in the construction of a point pattern prototype, which is the typical pattern of a collection. Prototypes are useful for identifying the typical pattern in a collection, much like the median is used to describe the typical observation in a univariate data set. The prototype has generally been used to construct robust representations of a collection of patterns using spike-time distance as the distance metric in applications including neuronal spike behavior, earthquake aftershocks, and wildfire activity in California (Schoenberg and Tranbarger, 2008; Diez et al., 2010a; Nichols et al., 2010).

The prototype $P_d(C)$ of the collection of patterns $C$ based on the distance measure $d$ is defined as the pattern that minimizes the following loss function:

$$\arg \min_{P} \sum_{X_i \in C} \alpha_i d(P, X_i)$$
where $\alpha_i$ is the weight corresponding to point pattern $X_i$. The weights $\alpha_i$ are typically chosen to be unity unless certain point patterns are deemed more important than others, or if the point patterns are measured with differential error. Recall that the median and mean can be defined in a similar way: each is the minimizer of a loss function represented by the sum of distances, the $L_1$ norm for the median and $L_2$ norm for the mean.

Kernel density and prototype summaries for ten simulated patterns are shown in Fig. 4. The kernel density is a pooled estimate based on all patterns and is heavily influenced by pattern 10. The prototype, using spike-time distance, is shown to be resistant to the tenth pattern and characterizes the typical pattern in the collection.

Figure 4: Kernel density and prototype summaries for a collection of ten patterns. Each pattern is represented by a sequence of events at times denoted by circles.

A two-dimensional example of a prototype is shown in Fig. 5. Here the sample collection contains 10 point patterns, nine of which are realizations of an inhomogeneous Poisson process with intensity shown in the plot in the first row and first column. Each pattern is expected to have about 15 points. The last point pattern – row 3, column 1 – is a realization of a homogeneous Poisson process with an overall intensity of three times that
of the original process. Just as in the one-dimensional case, the prototype, shown in the bottom-right panel, is resistant to a single atypical pattern.

Figure 5: The top-left panel shows an intensity function over the space $[0, 1] \times [0, 1]$. The second panel in the first row represents a legend of the intensity, and the next 9 panels show simulations of an inhomogeneous Poisson process based on the intensity, except the bottom-left panel, which is a realization of a homogeneous Poisson process with $\lambda = 42$. The prototype of these 9 point patterns, using spike-time distance with parameters chosen by default according to the recommendations in Diez et al. (2010a), is shown in the bottom-right panel.

Tranbarger and Schoenberg (2010) developed techniques for estimating the prototype in one dimension. This work was extended to multiple dimensions in Diez et al. (2010a); Diez (2010). Prototype methods are freely available through the \texttt{ppPrototype} function in the \texttt{ppMeasures} package on CRAN (R Development Core Team, 2010; Diez et al., 2010b). Many additional examples are included in the package help files.
6.2 Multidimensional scaling

Classical multidimensional scaling (MDS) is useful for pattern classification based on a distance metric. Given a collection of point patterns, \( \mathcal{C} = \{X_1, ..., X_n\} \), a distance matrix \( D \) is computed where \( D_{i,j} = d(X_i, X_j) \) for some distance measure \( d \). MDS uses this distance matrix to estimate relative locations of the patterns in the \( \mathbb{R}^k \) where \( k \) is generally selected by the user. Each pattern is itself represented by a point, and MDS embeds these points in locations of \( \mathbb{R}^k \). The goal of this embedding is to place points representing similar (dissimilar) patterns close to (far from) each other. To achieve this aim, MDS selects a location for each pattern such that the resulting Euclidean distances between the patterns, described by \( \tilde{D} \), approximates the pattern distances \( D \) by minimizing the following loss function:

\[
L(D, \tilde{D}) = \sum_{i \neq j} [D_{ij} - \tilde{D}_{ij}]^2.
\]

See e.g. Venables and Ripley (2002) for additional details on classical MDS and its variants, such as Kruskal’s MDS and Sammon mapping.

MDS can be useful in both identifying groupings of points – in this case, groups of patterns – and in classification. If classifications are provided on a training set, then a distance metric may be applied to compute the distance matrix \( D \). Applying MDS places the patterns into a new space \( \mathbb{R}^k \), where traditional classification techniques may be applied. The MDS methods are implemented in the R package \texttt{smacof} using the \texttt{smacofSym} function (de Leeuw and Mair, 2009).

Fig. 6 shows 15 patterns in one-dimensional space. Each pattern arose from a Poisson process with an overall mean of 12 points per pattern, as described by the figure’s intensities in the left panel. The right panel of Fig. 6 shows MDS applied to these fifteen patterns.
using spike-time distance. One sees a grouping of the patterns based on their original intensity functions.

Figure 6: Left: Two Poisson intensities, each with a total intensity of 12. Middle: 15 patterns, five from the first Poisson process and ten from a second Poisson process depicted in the left panel. Right: Spike time distance was used as the metric to construct the matrix $D$, and multidimensional scaling applied to all 15 patterns. The black circles denote the first five patterns and the red crosses mark those of the second process.

Multidimensional scaling may be applied to any of the distance metrics introduced. Fig. 7 shows a second example of 15 two-dimensional patterns and their representation as points in $[0,1] \times [0,1]$ where the distance $D^K_2$ was used to construct the dissimilarity matrix. Each point pattern is a realization of one of two Neyman-Scott processes with different parameters. Again, one sees that MDS is quite successful at classifying the point patterns into two groups.

Appendix: Spike-time distance, prototypes, and multidimensional scaling in R

We consider the packages \texttt{ppMeasures} and \texttt{smacof} for the R statistical software. These packages and the R statistical software are freely available online from the Comprehensive R Archive Network: r-project.org. The \texttt{ppMeasures} and \texttt{smacof} packages are useful in
Figure 7: The first two rows are realizations of a Neyman-Scott process with parent intensity of 8 and 6 children per parent uniformly distributed within a distance of 0.1. The point patterns in rows 3 and 4 are realizations of a Neyman-Scott process with parent intensity 24 and 2 children per parent within a radius of 0.1. The bottom-right panel shows MDS applied to these 15 patterns using the distance function $D_K^2$, where the black circles denote the eight patterns from the first Neyman-Scott process and the red crosses mark those of the second Neyman-Scott process.
computing spike-time distance, prototypes, and implementing multidimensional scaling. This introduction uses simulated data, and we assume the user is familiar with vectors, matrices, loops, subsetting techniques, and basic plotting in R. To install the packages using an internet connection, use the `install.packages` function, and to load the packages, use the `library` function:

```r
> install.packages(c('ppMeasures', 'smacof'))
> library(ppMeasures)
> library(smacof)
```

Spike-time distance is useful in assessing the similarity or dissimilarity in two point patterns. The `stDist` function in the `ppMeasures` package computes spike-time distance for two patterns. Its required arguments are the two point patterns entered as matrices, where rows represent points, and a moving penalty that can be a single value or a vector where the \( i^{th} \) entry corresponds to movement in the \( i^{th} \) dimension. Additional arguments include addition and deletion penalties, and other arguments that may be modified to compute variants of spike-time distance and other function options. Below two patterns are generated, `p1` and `p2`, using the `rgamma` function to generate the location of the points. The distance is computed using a moving penalty of 10 and the default addition and deletion penalties, which are both 1. A graphic representing how the distance was computed is generated using the `plot` method applied to the output of `stDist`, which is shown in the left panel of Fig. 8. Points connected by a line represent corresponding points in patterns 1 and 2, and isolated points represent added or deleted points.

```r
> p1 <- cbind(rgamma(20, 2), rgamma(20, 1.8))
> p2 <- cbind(rgamma(20, 2), rgamma(20, 1.8))
> (hold <- stDist(p1, p2, 1))
[1] 19.35158
```
Figure 8: Left: A plot representing the optimal transformation of pattern 1 into pattern 2 for the chosen parameters. Center: A collection of 60 patterns and their prototype, shown as pattern 0. Right: Multidimensional scaling applied to the patterns shown in the center panel. These patterns were generated using three random processes, and the coloring and plotting characters denote those groups.

> plot(hold, col=c("#444444", ".BB0000"))
> legend("topright", legend=c("pattern 1", "pattern 2"),
+   pch=c(1,19), col=c("#444444", "#BB0000")

The default colors in the plot method for the stDist output use an alpha level, which may require first opening a Cairo plotting device on a Linux or Windows operating system. The default colors may be changed using extra arguments in the plot method, as shown above.

We next build a point pattern collection and find the pattern’s prototype. A pattern collection is organized using the ppColl function, which has two required arguments. The first argument is a matrix where each row represents the spatial (or spatial-temporal) location of a point; a vector of values may be provided if there is only one spatial dimension. Below 60 patterns are constructed, where the first 15 come from one point process, the next 20 come from another process, and the last 25 are from a third process. We will use these patterns for both the prototype and the multidimensional scaling examples.
```r
> groups <- rep(1:3, c(15, 20, 25))
> nPts <- rpois(60, c(25, 35, 14)[groups])
> pts <- rep(-1, sum(nPts))
> key <- rep(-1, sum(nPts))
> start <- 0
> for(i in 1:60){
+ shape <- c(1.2,1.5,1)[groups[i]]
+ pts[start+1:nPts[i]] <- rgamma(nPts[i], shape=shape)
+ key[start+1:nPts[i]] <- i
+ start <- start + nPts[i]
+ }
> patColl <- ppColl(pts, key, nMissing=sum(nPts==0))

The additional \texttt{nMissing} is provided here to \texttt{ppColl} in the event that one or more patterns
have no points, which would otherwise be undocumented in the pattern collection.

The point pattern prototype is computed using the \texttt{ppPrototype} function. A pattern
collection organized via \texttt{ppColl} is the first required argument; a moving penalty is the
second. Below is code that computes the prototype of the point pattern collection, plots
the collection, and plots the prototype.

```r
> (proto <- ppPrototype(patColl, 10)) # '...' represents omitted output
  dim  1
1  0.2529
2  0.3709
...
13 1.7100
> plot(patColl, addLines=TRUE, col=groups)
```
The resulting plot is shown in the center panel of Fig. 8. The patterns from the collection are shown as rows 1-60, and the prototype is listed as pattern 0. Optional arguments in the `ppPrototype` function adjust the distance function used to compute the prototype (e.g. addition and deletion penalties) and include options for changing parameters in the algorithm used to compute the prototype. For a comparison of the available algorithms, see Diez (2010).

Multidimensional scaling is applied using the `smacofSymm` function in the `smacof` package. This function requires a single argument: a symmetric matrix with entries representing the distance between the objects in the collection. In this instance, the objects are patterns, and the distances are computed below using spike-time distance in a loop. Notice that only the lower-triangle of the matrix is computed, and the upper-triangle is filled in by adding the matrix to its own transpose.

```r
> dists <- matrix(0, 60, 60)
> for(i in 2:60){
+   x <- pts[key == i]
+   for(j in 1:(i-1)){
+     y <- pts[key == j]
+     dists[i,j] <- stDist(x, y, 10)$dist
+   }
+ }
> dists <- dists + t(dists)
> hold <- smacofSym(dists)
> plot(hold$conf, col=groups, pch=c(1,3,19)[groups])
```
The last line of code plots this matrix, coloring the points and also ensuring each group has its own color and plotting character. The right plot in Fig. 8 shows the resulting plot with the locations suggested by smacof. The suggested object locations should only be analyzed relative to their neighbors; the coordinate values for single object holds no meaning when it is not referenced to the other objects. Here we presented only one possible set of parameters in both the distance and multidimensional scaling operations. Changing the default parameters in stdist or smacofSym may result in important changes in the apparent clustering of the objects.

See Diez (2010) and de Leeuw and Mair (2009) for additional details on the functions and algorithms in the ppPrototype and smacof packages, respectively.

References


