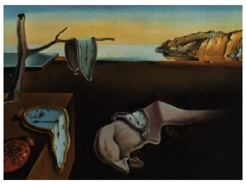

Journal of Statistical Software

October 2015, Volume 67, Issue 5. doi: 10.18637/jss.v067.i05

pdc: An R Package for Complexity-Based Clustering of Time Series

Andreas M. Brandmaier
Max Planck Institute for Human Development

Abstract

Permutation distribution clustering is a complexity-based approach to clustering time series. The dissimilarity of time series is formalized as the squared Hellinger distance between the permutation distribution of embedded time series. The resulting distance measure has linear time complexity, is invariant to phase and monotonic transformations, and robust to outliers. A probabilistic interpretation allows the determination of the number of significantly different clusters. An entropy-based heuristic relieves the user of the need to choose the parameters of the underlying time-delayed embedding manually and, thus, makes it possible to regard the approach as parameter-free. This approach is illustrated with examples on empirical data.

Keywords: complexity, time series, clustering, permutation entropy, R.

1. Introduction

Clustering is an unsupervised technique to partition a data set into groups of similar objects with the goal of discovering an inherent but latent structure. Similarity of static objects is often formalized by embedding objects into a space by means of a suitable measure, e.g., the city-block distance, Euclidean norm, Mahalanobis distance, or correlation. Based on a chosen loss function, clustering algorithms find or approximate an optimal partition of these points. Various approaches exist to construct clusters, mainly differing in whether the number of clusters has to be specified beforehand and whether the solution is partitional or hierarchical. Among the most common and well-understood clustering algorithms is k-means (MacQueen 1967) – or variants of it, such as k-medians or k-centroids – and agglomerative hierarchical clustering (Johnson 1967). Whereas clustering static data is well understood, clustering time series raises additional difficulties. Direct clustering based on the Euclidean space spanned by the raw signals limits the approach to time series of equal length and lacks robustness by restricting similarity to time and shape. Therefore, most approaches reduce the time series to a meaningful description in a low-dimensional feature space by means of geometric, algebraic or symbolic approximations, or cluster the parameters of a model-based description (see, e.g., Liao 2005). A more recent empirical comparison of distance measures was performed by Ding, Trajcevski, Scheuermann, Wang, and Keogh (2008). Fulcher, Little, and Jones (2013) have demonstrated empirically clustering across both time series data and analysis methods. Most clustering algorithms formalize similarity by shape, i.e., as it is apparent to the human eye. However, clusters may arise from structural changes of the underlying dynamical system that may well be hidden under heavy observational noise. This motivates clustering based on the relative complexity between time series, which defines similarity based on statistical properties of the time series instead of apparent shape similarity.

Permutation distribution clustering (PDC; Brandmaier 2012) is a complexity-based approach to clustering time series. The main idea is to formalize similarity as small relative complexity between two time series. As a proxy of complexity, we choose the *permutation distribution* (PD). This distribution was introduced by Bandt and Pompe (2002) who interpret its entropy as a measure of complexity of a univariate time series. Permutation entropy has been successfully employed to analyze the complexity of data sets from various fields, including medicine (Li, Ouyang, and Richards 2007; Bruzzo, Gesierich, Santi, Tassinari, Birbaumer, and Rubboli 2008; Nicolaou and Georgiou 2011), geology (Hao and Zhao 2010) and engineering (Nair, Krishna, Namboothiri, and Nampoori 2010). The importance of clustering time series according to a complexity-based formalization of dissimilarity was also recognized previously in a broader context (Li, Chen, Li, Ma, and Vitányi 2004; Keogh, Lonardi, and Ratanamahatana 2004b). We use the squared Hellinger distance, a metric approximation of the Kullback-Leibler divergence (Kullback and Leibler 1951), between the distributions of signal permutations of embedded time series to obtain a dissimilarity matrix of a set of times series. This then serves as input for further clustering or projection.

The PD has several properties that make it an interesting candidate for a dissimilarity measure. Firstly, it is invariant to all monotonic transformations of the underlying time series. In particular, addition and multiplication of positive constants to the time series do not change its PD, making the PD invariant to monotonic normalizations, e.g., standardization by subtracting the mean and dividing by standard deviation. This property relieves the researcher from deciding to normalize as a preprocessing step which often has a serious impact on the clustering results when common metric dissimilarity measures are used, e.g., the Euclidean distance. Another advantage arising from this property is the robustness against slow drift in a signal (i.e., wandering baseline), which often occurs due to the physical properties of the measurement device. In electroencephalography (EEG), amplifier drifts are observed (Fisch and Spehlmann 1999) and a common problem in accelerometry is thermal drift of the sensors. Drifts of these kinds can be thought of as a local offset in the sensor reading that does not alter the PD. Furthermore, the compressed representation of PD allows the comparison of time series of varying lengths. And last but not least, PD has phase invariance, that is, it is not important when exactly patterns occur but only that they do occur.

PDC is available in package **pdc** (Brandmaier 2015) for the free statistical computing language R (R Core Team 2015), which is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=pdc`. The package **pdc** provides a fast, native implementation to calculate the permutation distribution of time series, together with easily accessible methods to calculate dissimilarity matrices and perform clustering on them.

Importantly, it enables users to either specify meta-parameters, the embedding dimension and time delay, manually or by using an entropy-based heuristic. The package contains secondary functions, e.g., the leave-one-out procedure for cluster evaluation and shape tracing routines for clustering convex shapes. Finally, plotting facilities are provided to render dendrograms of time series. The **TSclust** package (Montero and Vilar 2014) provides implementations of various clustering algorithms. Recently, **TSclust** had a dependency on package **pdc** added, thus allowing comparing PDC to a broad range of alternative approaches. We will illustrate this in the examples later on.

In the remainder of this article, we review the PD and derive a dissimilarity measure based on the Kullback-Leibler divergence between two PDs. We examine a heuristic to automatically choose parameters of the required time-delayed embedding of the time series and a heuristic to determine the number of clusters in a hierarchical PD clustering. We conclude with applications on simulated and real data, and a discussion of limitations and future work.

2. Method

2.1. Permutation distribution

In the following, the calculation of the PD and the construction of a dissimilarity measure between time series based on their PD representation is described. In brief, the PD assigns a probability to the occurrence of certain patterns of the ranks of values in a time series (see Bandt and Pompe 2002). To this end, the time series is partitioned into subsequences of a fixed length, m, which is also referred to as embedding in m-space. To calculate the PD on coarser time scales, the embedding can be time-delayed with delay t, such that only every tth element is regarded when forming subsequences. For each subsequence, the ranks of the values are calculated, for instance, as by-product of sorting the observed values. The permutation distribution is obtained by counting the relative frequency of the distinct rank patterns, also referred to as *ordinal patterns*. Each possible rank pattern can be identified by a permutation of the values 0 to $m - 1$ and hence the name *permutation distribution*. The reliance on ranks is a distinct feature of the PD as the distribution is determined by the observed values relative to each other instead of their absolute values. Figure 1 illustrates ordinal patterns for an embedding of $m = 3$.

Given a time series $X = \{x(i)\}_{i=0}^{T}$, sampled at equal intervals with $x(i) \in \mathbb{R}$, the time-delayed embedding of this time series into an m-dimensional space with time delay t, is $X' = \{[x(i), x(i+t), x(i+2t), ..., x(i+(m-1)t)]\}_{i=0}^{T'}$ with a total of $T' = T - (m-1)t$ elements. The ordinal pattern for an element $x' \in X'$ can be obtained by computing the permutation of indices from 0 to $(m - 1)$ that puts the m values into sorted order. If two elements $x'(i)$, $x'(j)$, $i \neq j$ have the same value, they will keep their original order relative to each other[1]. There are $m!$ unique permutations of length m, and, thus, $m!$ distinct ordinal patterns.

The permutation distribution of a time series is obtained by counting the frequencies of the distinct observed ordinal patterns of the elements $x' \in X'$. Let $\Pi(x)$ be the permutation that

[1]If ties are frequent, adding a small amount of noise to the data can help to avoid bias towards specific ordinal patterns representing these ties.

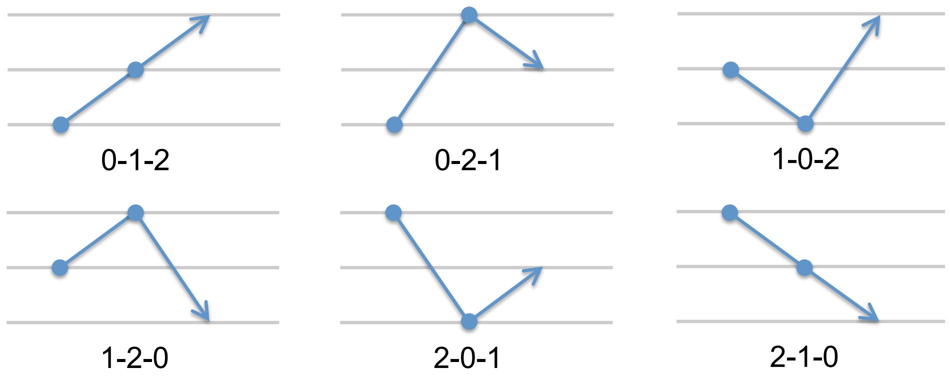

Figure 1: Each time series segment (blue arrows) is a representative of one of the six distinct ordinal patterns for an embedding size of three. For example, the first ordinal pattern (0-1-2) encompasses all patterns for which the first value of the segment is the smallest, and the last one is the largest, irrespective of the actual observed values.

an element $x \in \mathbb{R}^m$ undergoes when being sorted. The permutation distribution of X' is

$$p_\pi = \frac{\#\left\{x' \in X' | \Pi\left(x'\right) = \pi\right\}}{T'}. \tag{1}$$

In the PD, the temporal order of the ordinal patterns is discarded and the distribution solely represents the frequency of unique patterns in the time series. The permutation entropy of order $m \geqslant 2$ as introduced by Bandt and Pompe (2002) is defined as the Shannon entropy of the probability distribution P:

$$H(P) = -\sum_{\pi \in S_m} p_\pi \log p_\pi \tag{2}$$

with S_m being the set of all m-permutations.

The dissimilarity between two time series can be formalized as a dissimilarity of their respective permutation distributions. The Kullback-Leibler (KL) divergence, also known as relative Shannon entropy, is often employed as a measure of divergence between probability distributions and represents a natural expansion of the entropy as a complexity index to calculate relative permutation entropy as a relative complexity index. However, the KL divergence violates the triangle inequality and, thus, is not a metric. Therefore, we employ the squared Hellinger distance instead to embed the permutation distributions of time series into a metric space. Metric spaces open up the possibility to improve runtime considerably (Chávez, Navarro, Baeza-Yates, and Marroquín 2001; Elkan 2003; Moore 2000). Up to scaling, the squared Hellinger distance is equal to the Euclidean norm of the difference of the square root vectors of the discrete probability distributions. Let $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_n)$ be two permutation distributions. The squared Hellinger distance is:

$$D\left(P, Q\right) = \frac{1}{\sqrt{2}} \left\|\sqrt{P} - \sqrt{Q}\right\|_2^2.$$

The squared Hellinger distance can be derived by means of a Taylor approximation of the KL divergence (see Appendix A). It is a metric since it satisfies the triangle inequality, is symmetric and non-negative, and bounded between zero and one.

The pairwise squared Hellinger distances between the set of permutation distributions form a distance matrix that can be fed to a clustering algorithm of the researcher's choice. In this article, sequential agglomerative hierarchical non-overlapping clustering (Johnson 1967) was chosen as the clustering method. Initially, each time series is assigned to a cluster. Based on a dissimilarity measure, clusters are constructed by iteratively merging clusters until there is only a single top cluster left. This leads to a hierarchy of clusters. If not stated otherwise, distances between sets of time series are calculated by the *complete linkage* method, which is defined as the dissimilarity between two clusters C_i and C_j:

$$d_{complete}\left(C_i, C_j\right) = \max_{x \in C_i, y \in C_j} D(x, y). \tag{3}$$

The resulting binary tree is typically visualized in a dendrogram depicting the tree with branch heights that reflect the distance between clusters.

2.2. Entropy heuristic

The choice of parameters m (embedding dimension) and t (time delay) for the time-delayed embedding is crucial for the performance of the clustering. The choice of the embedding dimension necessitates the consideration of two counter-acting effects: the larger the embedding dimension, the larger the representational power (that is, the more distinct ordinal patterns we are able to detect). However, we expect to find fewer observations of each permutation and, thus, the estimate of the frequency will be increasingly unreliable. In the following, we formalize these observations in a heuristic that guides researchers in choosing an embedding dimension if no prior information is available. The heuristic aims at choosing an embedding dimension that is maximally expressable, that is, the PD should be maximally dissimilar to a uniform distribution. This notion is captured by the permutation entropy. Let P be a permutation distribution of embedding dimension m. Let $\log_0(x)$ be 0 if $x = 0$, and $\log(x)$ otherwise. The normalized entropy of the permutation distribution is

$$e_P\left(m\right) = -\frac{\sum_{\pi \in S_m} p_\pi \log_0 p_\pi}{\log\left(m!\right)}.$$

The normalization bounds the estimated entropy between zero and one independently of the embedding dimension. A white-noise time series yields a uniform permutation distribution and an entropy of one. Let $\mathcal{P}$ be a set of permutation distributions of embedding dimension m. The heuristic chooses the embedding dimension with the lowest average, normalized permutation entropy

$$\arg\min_m \sum_{P \in \mathcal{P}} e_P\left(m\right).$$

In order to find the appropriate embedding dimension for clustering a set of time series, we calculate the average normalized entropy of a set of distributions. In a similar vein, different choices of the time delay t can be selected. An illustration of a grid-search on the parameters m and t to determine differences between two autoregressive moving average (ARMA) processes is given in Figure 2. The corresponding code can be found in Section 3.

2.3. Heuristic to determine the number of clusters

PDC offers the possibility to employ a heuristic for the determination of the number of distinct clusters. This is achieved by a top-down procedure subsequently testing null hypotheses of

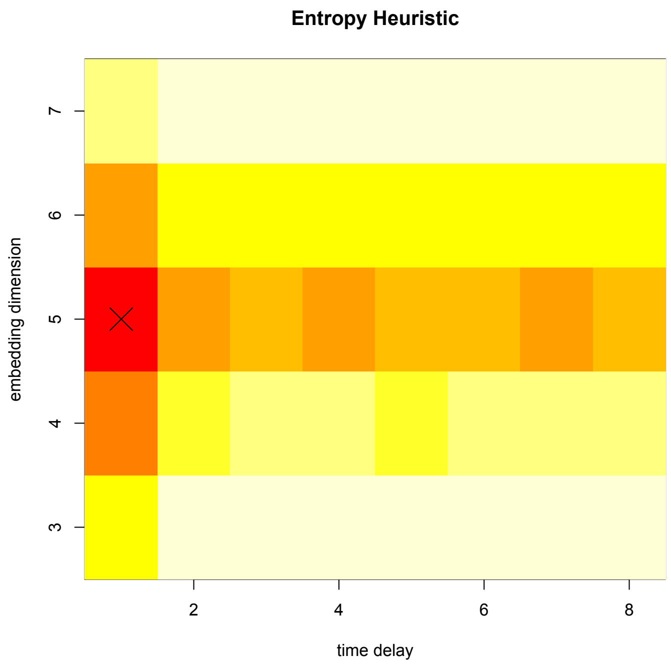

Figure 2: A graphical representation of the average entropy estimates for varying time delays and embedding dimensions on a data set with prototypes generated from two different ARMA processes.

equal PDs in the hierarchy of clusters. Only if it is very unlikely that data were observed under the null hypothesis, the decision to split the cluster into distinct groups is made, and testing is recursively continued. We regard the statistical model of a cluster as a multinomial PD and the model of its two subclusters as a joint model of two multinomial distributions. Since these models are algebraically nested, the likelihood ratio test can be applied. Under the null hypothesis that two codebooks that are to be merged are drawn from the same distribution and the clusters are fixed in advance, the log-likelihood ratio (LLR) is χ^2-distributed with $m!-1$ degrees of freedom. In this framework, we reject the null hypothesis if we find extremely unlikely values of the χ^2-test statistic. Note that clusters are created depending on the chosen dissimilarity measure and linkage method, which may lead to biased results.

Assume two clusters with respective permutation distributions P and Q. Let R be the joint codebook of P and Q representing a fusion of the two clusters. The LLR between a model of two individual clusters and a model of a single joint cluster is given by the following (also see Appendix B)

$$LLR\,(P,Q|R) = 2N \sum_{\pi \in S_m} p_\pi \log\left(\frac{r_\pi}{p_\pi}\right) + 2N \sum_{\pi \in S_m} q_\pi \log\left(\frac{r_\pi}{q_\pi}\right).$$

Notably, the LLR function coincides with the sum of the KL divergences between P and R, and between Q and R up to a scaling factor of $2N$, with N being the number of time series.

2.4. Implementation

The naive implementation for determining the PD of a time series is based on shifting a window of size m over the time series, determining the permutation of the elements within the window by a sorting algorithm, and applying a canonical bijective mapping from permutations to integer indices that are employed to count the frequency of the respective patterns in an array structure. The bijection is provided by Lehmer codes (Lehmer 1960) that are based on a factorial numbering system. The asymptotic time complexity for finding the distribution of a time series of length T is then $O(T \cdot m \log m)$ with a space complexity of $O(m!)$ for storing the array of codeword counts. However, a tremendous gain in processing speed can be gained by optimizing this procedure based on the fact that the actual sorting of items is not needed. Therefore, we use an explicit implementation of a sorting algorithm as a tree of if-clauses with each inner node being a pairwise comparison of items and each leaf having a unique integer label between 0 and $m!$ representing the permutation index. This implementation is more efficient because no items actually have to swap places, no subroutines are called, and no extra memory is required to be reserved on the stack or the heap. Since the typically reasonable embedding dimensions range between 3 and 9, the respective code blocks can be pre-generated. For all larger embedding dimensions, a code generator can generate the necessary code snippets on demand. The code generator works in a recursive fashion. An $m \times m$ matrix H, which is initialized with zeroes, is used to store the transitive closure of the smaller relation between the m items of a pattern $x \in \mathbb{R}^m$. Matrix elements indicate the following relation:

$$H_{ij} = \begin{cases} 1 & x_i > x_j, \\ -1 & x_i \leq x_j, \\ 0 & \text{relation yet unknown.} \end{cases}$$

That is, the permutation index of x can be uniquely determined if the relation between all pairs is known, that is, only if the lower triangle of H is non-zero. Therefore, we iterate through the lower-triangle matrix, e.g., using the Cantor function, and perform the respective pairwise comparisons. After each update of an element in H, we update H to represent the transitive closure, e.g., with the Floyd-Warshall algorithm (see Cormen, Leiserson, Rivest, and Clifford 2004). After each new determination of an entry in H_{ij}, the algorithm recursively calls itself with a copy of H, once having $H_{ij} = 1$, and once having $H_{ij} = -1$. The algorithm terminates when the lower triangle of H contains no more zeroes. This algorithm can be used to create a tree of if-clauses that compares elements of x. The recursive calls correspond to the if-else blocks of the pairwise comparisons, whereas the termination points correspond to the returning of unique integer indices. A sample output of this algorithm can be found in Appendix C.

3. Code examples

3.1. Clustering of autoregressive time series

As a first example, we generate two ARMA processes using `arima.sim`. For the first one, we choose auto-regressive coefficients $ar = (0.8897, -0.4858)$ and moving-average coefficients $ma = (-0.2279, 0.2488)$, and for the second one, $ar = (-0.71, 0.18)$, $ma = (0.92, 0.14)$. We create five time series from each condition with a length of 500 samples each.

```
R> set.seed(69266)
```

```
R> grp1 <- replicate(5, arima.sim(n = 500, list(
+    ar = c(0.8897, -0.4858),  ma = c(-0.2279, 0.2488)), sd = sqrt(0.1796)))
R> grp2 <- replicate(5, arima.sim(n = 500, list(
+    ar = c(-0.71, 0.18), ma = c(0.92, 0.14)), sd = sqrt(0.291)))
R> X <- cbind(grp1, grp2)
```

In a second step, we perform PDC with an automatic selection of the embedding dimension.

```
R> clustering <- pdclust(X)
R> clustering

Permutation Distribution Clustering

Embedding dimension: 5
Time delay          : 1
Number of objects: 10
Clustering method: complete

R> truth.col <- rep(c("red", "blue"), each = 5)
R> plot(clustering, cols = truth.col)
```

The `cutree` command returns the group memberships of the time series for a desired number of clusters, in our case two clusters. We note that the result perfectly matches the expected memberships.

```
R> cutree(clustering, k = 2)

[1] 1 1 1 1 1 2 2 2 2 2
```

Using `loo1nn`, which implements a leave-one-out crossvalidation scheme, we estimate the predictive accuracy of the clustering (Keogh and Kasetty 2003), i.e., the proportion of observations closest to the true class member, and obtain an accuracy of 100%.

```
R> loo1nn(clustering, truth.col)

[1] 100
```

As we see from the summary of the `clustering` object, the embedding dimension was automatically set to 5. The following code yields a graphical representation of the average entropies by embedding dimension and time delay, which are used to determine the hyper-parameters automatically. The resulting plot is shown in Figure 2.

```
R> mine <- entropyHeuristic(X, t.max = 8)
R> summary(mine)

Embedding dimension:  5 [ 3,4,5,6,7 ]
Time delay:  1 [ 1,2,3,4,5,6,7,8 ]
```

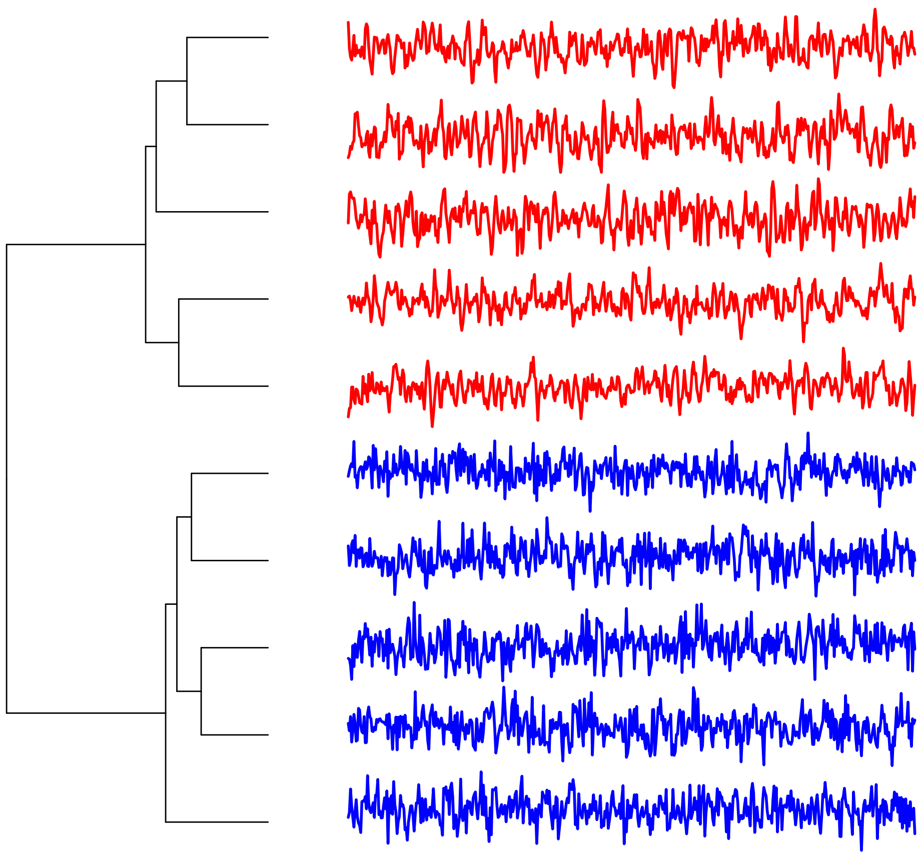

Figure 3: Hierarchical clustering of five time series from two different ARMA models each. Group membership is color-coded in red and blue. The top-level clustering reflects the differences in the generating models.

```
R> plot(mine)
```

The dendrogram of the hierarchical clustering is shown in Figure 3. The top-level clustering perfectly reflects the differences in the generating models. Note that the robustness of PDC to long-term trends can be further explored by adding deterministic or stochastic trend components to the ARMA model.

3.2. A comparison to UCRTSA

Keogh *et al.* (2004b) presented an important data set for the evaluation of clustering algorithms. The authors collected 18 pairs of time series from the *University of California Riverside (UCR) Time Series Archive* (Keogh and Folias 2002) that largely differ in their characteristics. Their ad-hoc Q-measure counts the number of correctly retrieved pairings at the lowest clustering level divided by the total number of pairs. Thus, the measure evaluates the known structure of the lowest clustering level without regarding the unknown higher-level structure. The authors report that more than three quarters of the clustering approaches that they tested yielded the worst possible score of $Q = 0$. The best results were achieved by the simple Euclidean distance with $Q = 0.27$, dynamic time warping with $Q = 0.33$, piecewise

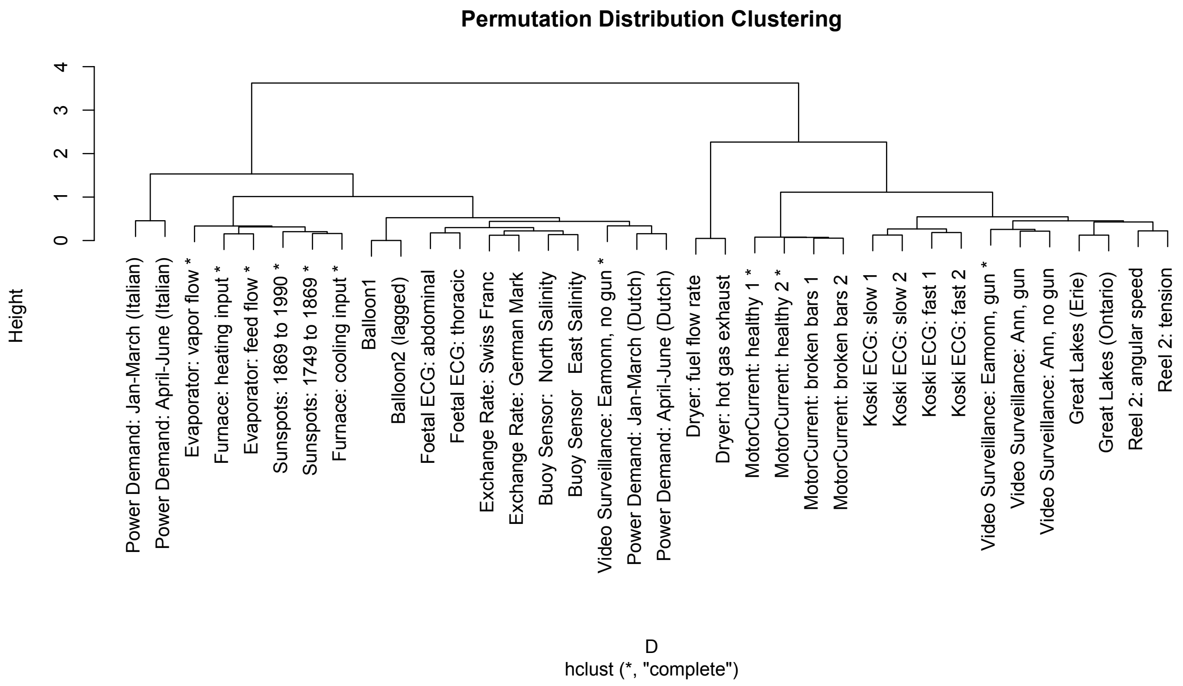

Figure 4: Dendrogram of 18 pairs of time series originating from the UCR Time Series Archive. The clustering retrieves the sources very well. Pairs of time series that are not clustered together at the bottom level are marked with an asterisk.

linear approximation with $Q = 0.33$, linear predictive coding (LPC) cepstra, autocorrelation with $Q = 0.16$, and longest common subsequence with $Q = 0.33$. The authors report that compression-based clustering is superior on the data set, yielding a $Q = 1.0$.

PDC with an automatically determined $m = 5, t = 1$ (see Figure 4 for the dendrogram) can keep up very well with 13 out of 18 time series in perfect pairs at the lowest levels ($Q = 0.72$). The resulting clustering does not perfectly reproduce the pairs on the lower level but, as can be seen in the illustration, similar time series are very close to each other.

This first example shows that PDC can indeed cluster time series that differ in the structure of the generating process. The data set was also made available in package **TSclust**. The following code shows how to obtain the clustering of the data set using package **pdc**.

```
R> library("TSclust")
R> data("paired.tseries", package = "TSclust")
R> truth <- rep(1:18, each = 2)
R> clust <- pdclust(paired.tseries)
R> plot(clust, timeseries.as.labels = FALSE,
+    labels = names(colnames(paired.tseries)))
```

The Q-measure can be obtained by counting the proportion of clusters that are merged at the bottom level:

```
R> merges <- clust$merge[apply(clust$merge < 0, 1, all), ]
R> merges <- merges[(abs(merges[, 1] - merges[, 2]) == 1), ]
```

```
R> merges <- -merges[(merges[, 1] %% 2 == 1), ]
R> num.labels <- length(clust$labels)
R> Q <- nrow(merges) * 2 / num.labels
R> cat("Q = ", round(Q, 2), "\n")

Q =  0.72
```

Package **TSclust** allows us to easily compare[2] other clustering approaches to PDC. For example, to obtain an estimate of clustering performance of dynamic time warping, correlation-based clustering, partial autocorrelation-based clustering, autocorrelation-based clustering, complexity-invariant clustering and PDC (see Montero and Vilar 2014, for details):

```
R> methods <- c("DWT", "COR", "PACF", "ACF", "CID", "PDC")
R> D.list <- lapply(methods, function(x) diss(t(paired.tseries), x))
R> acc <- data.frame(t(sapply(D.list, loo1nn, truth)))
R> names(acc) <- methods
R> round(acc, 1)

   DWT  COR PACF  ACF  CID  PDC
1 38.9 30.6 69.4 69.4 44.4 72.2
```

We obtain that in this case the crossvalidated accuracies of the dissimilarity matrices are all inferior to PDC.

3.3. Clustering EEG data

Permutation entropy has been applied to biomedical signals in previous work, e.g., it was used as a feature in a supervised learning task to predict absence seizures (Li *et al.* 2007) or the fetal behavioral state from biomagnetic recordings (Frank, Pompe, Schneider, and Hoyer 2006). In this section, we apply PDC on resting state EEG recordings from participants who were recorded once with their eyes open and once with their eyes closed. PDC was previously applied to cluster EEG data (Brandmaier 2012) but the data set could not be made available to the public. Therefore, a data set of recordings of brain electrical activity from a study by Andrzejak, Lehnertz, Mormann, Rieke, David, and Elger (2001) was obtained, which is freely available to download (Andrzejak 2014). For the following illustration, we used data sets *A* and *B*, each containing 100 single-channel electroencephalographic recordings of 23.6s duration, sampled at 173.61Hz, and resulting in time series of 4096 samples each. The sets correspond to segments from five healthy participants each during eyes-open (A) and eyes-closed (B) conditions. Generally, power in the 8–12Hz frequency band (α-power) is suggested as a measure of resting-state arousal (Barry, Clarke, Johnstone, Magee, and Rushby 2007). We examined whether PDC can find partitions of the underlying dynamics of the two conditions. The following code blocks assume the data sets to reside in `path` in respective subfolders `A-EyesClosed` and `B-EyesOpen`. For purposes of illustration, we load only the first 10 instances of each class.

[2]The set of reasonable candidate algorithms should be influenced by the researchers' prior knowledge about the time series, e.g., some algorithms might be considered appropriate for aligned shapes only.

```
R> tmp <- tempdir()
R> files.A <- paste(tmp, "setA.zip", sep = "/")
R> files.B <- paste(tmp, "setB.zip", sep = "/")
R> download.file("http://ntsa.upf.edu/system/files/download/Z.zip", files.A)
R> download.file("http://ntsa.upf.edu/system/files/download/O.zip", files.B)
R> instances.each <- 10
R> X <- matrix(NA, nrow = 4097, ncol = instances.each * 2)
R> for (i in 1:instances.each) {
+    z <- unz(files.A, paste("Z", formatC(i, width = 3, flag = "0"), ".txt",
+      sep = ""))
+    X[, i] <- as.numeric(readLines(z))
+    close(z)
+    z <- unz(files.B, paste("O", formatC(i, width = 3, flag = "0"), ".txt",
+      sep = ""))
+    X[, instances.each + i] <- as.numeric(readLines(z))
+    close(z)
+ }
R> colnames(X) <- rep(c("closed", "close"), each = instances.each)
```

We calculate the entropy heuristic for time delays up to 10 and across the default embedding dimensions. We perform the clustering and plot the entropy surface using the `plot` command and plot the clustering dendrogram.

```
R> mine <- entropyHeuristic(X, t.max = 10)
R> clust <- pdclust(X)
R> cols <- rep(c("red", "blue"), each = instances.each)
R> plot(mine)
R> plot(clust, cols = cols)
```

The following code allows us to compare the clustering accuracy over different methods.

```
R> library("TSclust")
R> methods <- c("DWT", "COR", "PACF", "ACF", "CID", "PDC")
R> D.list <- lapply(methods, function(x) diss(t(X), x))
R> acc <- data.frame(t(sapply(D.list, loo1nn, cols)))
R> names(acc) <- methods
R> acc

  DWT COR PACF ACF CID PDC
1  60  55   65  70  55  90
```

We obtain a leave-one-out accuracy of 90% for PDC. This is in line with the previous finding of Brandmaier (2012), who reported an average accuracy of 92% averaged across five participants, and again seems to be superior to the set of candidate clustering approaches.

3.4. Shape complexity

Shape clustering is a fundamental problem in computer vision, e.g., for speeding up database retrieval or automatic labeling of objects (Shen, Wang, Bai, Wang, and Latecki 2013). In the following, we illustrate the potential usefulness of PDC for clustering objects with planar shapes. Shape signatures of objects can be created by unrolling their contour around its centroid across "time" (Gonzalez and Woods 1992; Keogh, Wei, Xi, Vlachos, Lee, and Protopapas 2009; Batista, Wang, and Keogh 2011). The resulting time series represents distance-to-center of points on the contour versus radial angle. The length of the resulting time series N is related to the angular resolution $\Delta\phi = 2\pi/N$ in radians. To this end, package **pdc** provides the function `traceImage`.

In order to be applicable, object descriptions must be invariant with regard to translation, rotation, and scale, and should have low computational complexity (Gonzalez and Woods 1992; Zhang and Lu 2004). Permutation distribution clustering of such shapes fulfills all of these requirements. The computational complexity is linear in the length of the time series and, thus, in the sampling rate that is used to trace the objects' shapes. If objects are centered, e.g., by positioning them on their center of mass, linear changes in scale of an object result in a linear rescaling of the time series. In a similar vein, rotations of an object will phase-shift the time series on the x-axis with parts exceeding 360° reappearing at 0° and vice versa. That is, with minimal deviations attributable to discretization errors, the permutation distribution will be invariant to object scale, rotation, and location.

We illustrate our considerations with a small data set of two shapes, a four-point and a five-point regular star, that were created each in two sizes (100% and 150%) and in two rotations (0° and 45°). The shapes were discretized to time series with an angular resolution of 3.6° per time point. In agreement with our expectations, each of the two top clusters found when PDC is applied represents one of the two classes of stars (see Figure 5, on the left). The clustering reflects the fact that the rotation leads to slightly more discretization errors than the scaling, that is, scaled versions of the same shape are clustered nearly perfectly, while there is a small inter-cluster variance between different rotations. We set a larger minimum time-delay of 5 to increase robustness over discretization errors when searching for the optimal delay. The embedding dimension was automatically selected to be $m = 4$ and the time-delay to be $t = 9$. The following code loads the data set containing the eight star shapes, shows how the time series representation of these plots can be converted to the original shape using a polar plot from package **plotrix** (Lemon 2006), and how clustering accuracy and a textual dendrogram can be obtained.

```
R> data("star.shapes", package = "pdc")
R> library("plotrix")
R> oldpar <- polar.plot(star.shapes[, 1])
R> par(oldpar)
R> ent <- entropyHeuristic(star.shapes, t.min = 5, t.max = 10)
R> clust <- pdclust(star.shapes, m = ent$m, t = ent$t)
R> truth <- sapply(colnames(star.shapes), function(x) substr(x, 1, 1))
R> loo1nn(clust, truth)
R> str(clust)

[1] 100

--[dendrogram w/ 2 branches and 8 members at h = 1.18]
  |--[dendrogram w/ 2 branches and 4 members at h = 0.025]
  |  |--[dendrogram w/ 2 branches and 2 members at h = 0]
```

```
 |  |  |--leaf "4-2-1"
 |  |  `--leaf "4-2-2"
 |  `--[dendrogram w/ 2 branches and 2 members at h = 0.00605]
 |     |--leaf "4-1-1"
 |     `--leaf "4-1-2"
 `--[dendrogram w/ 2 branches and 4 members at h = 0.371]
    |--[dendrogram w/ 2 branches and 2 members at h = 0.00389]
    |  |--leaf "5-2-1"
    |  `--leaf "5-2-2"
    `--[dendrogram w/ 2 branches and 2 members at h = 0.00477]
       |--leaf "5-1-1"
       `--leaf "5-1-2"
```

Package **pdc** also contains the raw image data, which can be used to plot a dendrogram augmented by the images instead of the time series. To this end, the package provides the `rasterPlot` function

```
R> data("star.shapes.raw", package = "pdc")
R> rasterPlot(clust, star.shapes.raw$images, aspect = 0.5)
```

To further illustrate the potential usefulness, another data set including objects of three different classes was created: glasses, bottles, and fish. The files were selected from `http://openclipart.org/`. The resulting clustering is shown in Figure 5 (on the right). PDC perfectly retrieves the object classes in the top three clusters within the created taxonomy. Shape signatures of glasses and bottles are similar because the glasses' shapes are merely bottles with a waistline; this is reflected in the clustering. Also, not surprisingly, the intra-class variance of the fish cluster is larger than in any of the other two reflecting the larger difference between the fish prototypes in comparison to the other two classes. These theoretical and empirical results promise a new efficient approach to shape signature clustering under location, scale, and rotation invariance.

```
R> data("complex.shapes", package = "pdc")
R> ent <- entropyHeuristic(complex.shapes, t.min = 5, t.max = 10)
R> summary(ent)

Embedding dimension:  3 [ 3,4,5,6,7 ]
Time delay:  5 [ 5,6,7,8,9,10 ]

R> clust <- pdclust(complex.shapes, m = ent$m, t = ent$t)
R> truth <- rep(c("fish", "bottle", "glasses"), c(5, 4, 5))
R> loo1nn(clust, truth)
R> data("complex.shapes.raw", package = "pdc")
R> rasterPlot(clust, complex.shapes.raw$images)
```

The following snippet illustrates a projection of the dissimilarity matrix obtained from PDC onto two components by minimizing a loss function called "strain." This is also known as multidimensional scaling or principal coordinates analysis. Also, the convex hull of each cluster is plotted as shaded area in the principal coordinate space. The resulting plot is shown in Figure 6 and can be created using the command `mdsPlot` provided by package **pdc**.

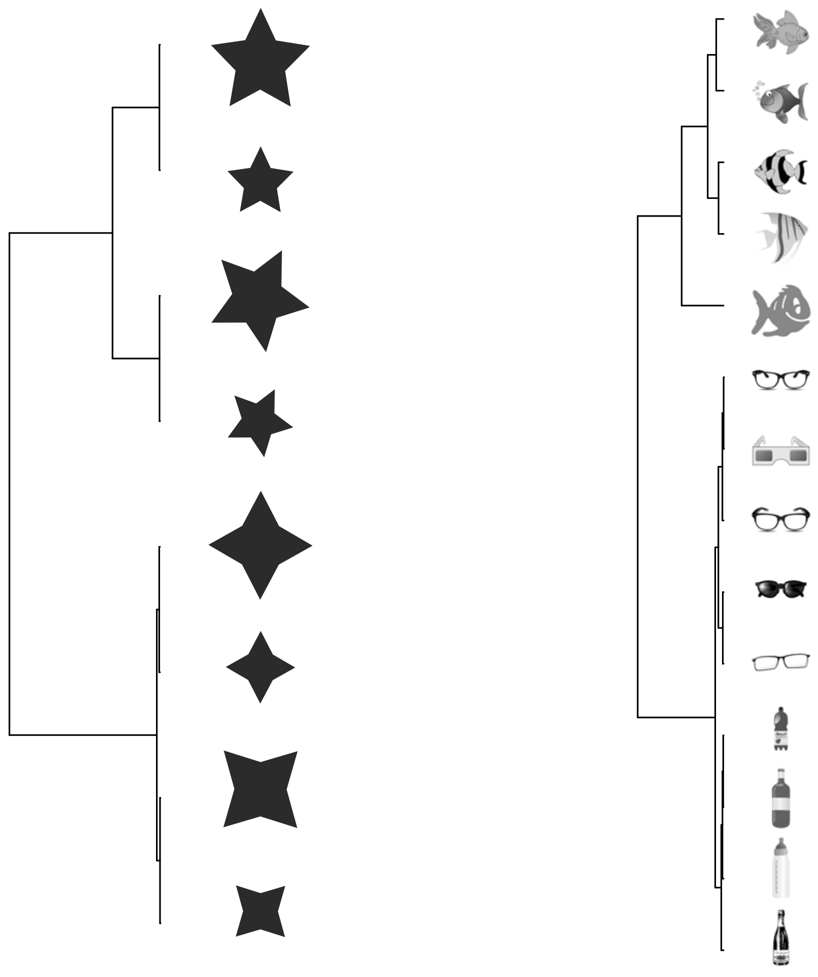

Figure 5: PDC of shape complexities. Left: A four-point and a five-point star, each in four different transformations of scale and rotation are correctly clustered into two different clusters. Right: Three different object categories: bottles, glasses, and fish.

```
R> mdsPlot(clust, truth, col = c("lightgray", "lightblue", "lightgreen"))
```

4. Discussion

Clustering plays a central role in deriving knowledge from data and can serve as a tool for data reduction, hypothesis generation, hypothesis testing, and prediction based on the resulting group structure (Halkidi, Batistakis, and Vazirgiannis 2001). To find a formalization of dissimilarity which accounts for task-specific invariances is crucial for successful clustering. We introduced a dissimilarity measure between time series that is based on a complexity-based divergence of the time series. We proposed applying this measure in a hierarchical clustering scheme to obtain tree-like partitions of a set of multivariate time series. In a unified framework, PDC combines criteria for selecting representational complexity that determines the number of distinct ordinal patterns by which a time series is represented, as well as the number of

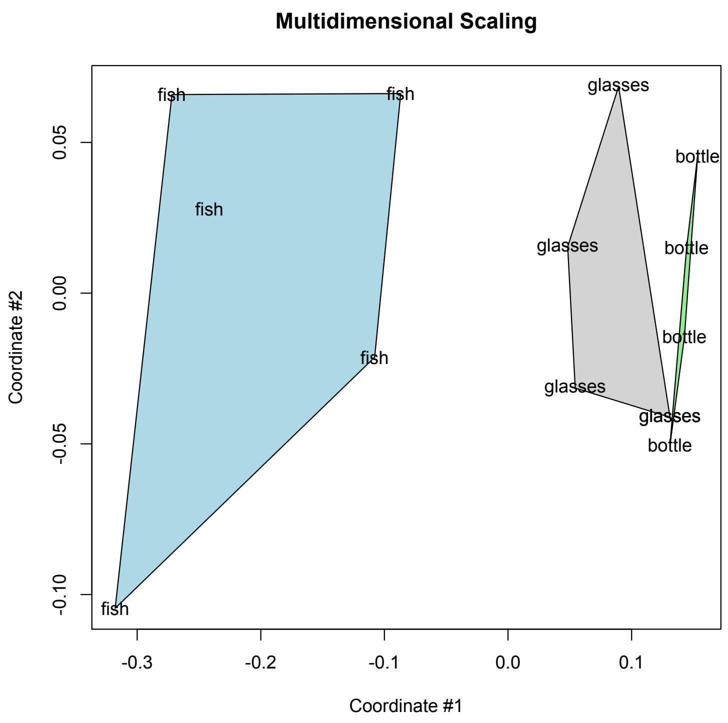

Figure 6: Multidimensional scaling of the dissimilarity matrix of the complex shapes data sets including pictures of bottles, fish, and glasses. Gray, blue, and green polygons illustrate the convex hulls of clusters.

distinct clusters. The proposed dissimilarity measure is invariant to the phase of the time series, invariant to monotonic transformations of the time series, particularly constant shifts and linear scalings, and is robust under drift and to outliers. Furthermore, it naturally allows the comparison of time series of differing length. Last, but not least, the runtime complexity is linear in the length of the time series and, thus, a highly efficient measure.

The (dis)similarity measure of PDC can be interpreted in two major ways. From a pattern recognition perspective, time series are similar if their local patterns of up- and down-movements match, irrespective of when and in what order these patterns occur. From an algorithmic-complexity point of view, time series are reduced into a lossy compressed representation, and time series are deemed similar if their compressed version is similar.

Time series segmentation is a challenging problem in many domains. Beyond clustering a set of different time series, clustering algorithms can be employed to detect anomalies in time series (Keogh *et al.* 2004b) or segment time series into similar and different parts. This can be achieved by sliding a window over the time series and treating the resulting segments as independent time series (Keogh, Chu, Hart, and Pazzani 2004a). Alternatively, top-down and bottom-up approaches or combinations thereof can be used (Keogh *et al.* 2004a). The window approach adds another parameter, the window size w, to the algorithm. Preselecting an appropriate window size is often considered as feasible (Keogh *et al.* 2004b).

Beyond taking a statistical approach, information-theoretic criteria can be employed to determine the number of clusters. As far as these criteria are likelihood-based, such as the well-known Akaike information criterion (AIC; Akaike 1973) or the Bayesian information criterion (BIC; Schwarz 1978), the criteria can be seen as automatic choices of the significance threshold of the likelihood ratio test. The decision of how many distinct clusters appear to be in the data is then seen as a sequential and nested model selection problem. Future work must examine whether the application of such criteria is beneficial for clustering.

The main limitation of PDC is its weakness in clustering time series with similarity in time and, less so, with similarity in geometry. Most time series clustering approaches aim to match similar shapes, trends, seasonality, or other algebraic basis functions of the time series. If differences really lie in crude shape patterns, PDC will perform less well than an algorithm that explicitly considers geometric-visual similarity. For example, if the moments of the time series are thought to represent discriminative information about time series, this information is discarded by PDC. Although this can be advantageous for some problems, it might be problematic for others. A prominent example is the synthetic control chart data set (Alcock and Manolopoulos 1999). The data sets consist of six noisy prototypical patterns: a constant, a cyclic pattern, an increasing trend, a decreasing trend, a sudden upward shift, and a sudden downward shift. The final patterns are obtained by mixing white noise to the basic patterns. In principle, PDC can succeed in determining the constant, the cycle, and the trends. However, both patterns with a sudden shift in either direction are by definition not captured by PDC. In domains where such a shift is considered irrelevant information, for example, if arising from a sensor misreading, this property is clearly beneficial. Whenever information of the moments should be preserved, nothing speaks against taking into account multiple features for clustering that capture the specific definition of similarity in the idiosyncratic context. The dissimilarity function can also be extended to include further features, like the signal's mean and the variance. Wang, Smith, Hyndman, and Alahakoon (2004) present such a feature-based approach, which also included complexity measures, but they did not use the PD. It would be highly interesting to examine the extent to which PDC can improve their cluster results.

PDC depends on the choice of two parameters, the embedding dimension and the time delay. We presented a heuristic that aims at selecting an embedding with maximal representational power. Of course, we cannot exclude cases in which the heuristic chooses a parameter that leads to a more discriminative representation between time series that does not match the intention of the researcher. This addresses a fundamental problem in clustering. Different researchers have different objectives. A sociologist is likely to expect a different clustering on a socio-economic panel than a banker who is granting loans. In this vein, different domains might necessitate different heuristics to choose embedding parameters. Our heuristic is meant to be a first take at automatically determining the parameters without prior information. Whenever there is some known ground-truth, the analysis paradigm shifts to semi-supervised analysis, and parameters can be chosen to maximize the performance on available labeled training data.

Fast and efficient searching in large databases is important for prediction, hypothesis testing, and rule generation (Faloutsos, Ranganathan, and Manolopoulos 1994). This is why lower bounding has become important. Lower bounding allows the creation of efficient, low-memory approximations of complex objects that can be held in main memory to allow efficient search and retrieval. By construction, the permutation distribution is a highly compressed version of the original time series and there is no need for further lower bounding and, thus, efficient search and retrieval is possible. Generally, observing time series for a longer time raises statistical power as this increases the precision with which codebooks and differences between them can be estimated. Statistical power to detect differences can in principle be simulated when hypotheses about the true effect size can be formed. Calculating the PD for time series in the range of millions of observations is still feasible on current desktop computers, e.g., a time series with 100 million observations is typically (based on a 1.6GHz Intel Core i7 processor) processed in a couple of seconds.

Clearly, the selection of the distance measure is the most important choice for a successful clustering. In this spirit, PDC is not meant to replace existing techniques; it merely adds another tool to the researcher's toolbox that promises success whenever similarity is expected regarding the complexity of time series, respectively regarding differences in the complexity of the underlying generators. Whenever researchers hypothesize that differences of dynamic processes pertain to specific properties like, e.g., trends or seasonality, these need to be incorporated in the decision of what clustering to use.

Acknowledgments

I would like to thank Timo von Oertzen for many helpful discussions. I am grateful to the anonymous reviewers who helped to improve both the manuscript and the R package. Thanks are also due to Julia Delius for her helpful assistance in language and style editing.

References

Akaike H (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In B Petrov, F Csáki (eds.), *Second International Symposium on Information Theory*, volume 1, pp. 267–281. Akademiai Kiado, Budapest.

Alcock R, Manolopoulos Y (1999). "Time-Series Similarity Queries Employing a Feature-Based Approach." In *7th Hellenic Conference on Informatics*, pp. 27–29. Ioannina, Greece.

Andrzejak R (2014). "Data Files." URL `http://ntsa.upf.edu/downloads/andrzejak-rg-et-al-2001-indications-nonlinear-deterministic-and-finite-dimensional`.

Andrzejak R, Lehnertz K, Mormann F, Rieke C, David P, Elger C (2001). "Indications of Nonlinear Deterministic and Finite-Dimensional Structures in Time Series of Brain Electrical Activity: Dependence on Recording Region and Brain State." *Physical Review E*, **64**(6), 061907. `doi:10.1103/physreve.64.061907`.

Bandt C, Pompe B (2002). "Permutation Entropy: A Natural Complexity Measure for Time Series." *Physical Review Letters*, **88**(17), 174102–1–174102–4. `doi:10.1103/physrevlett.88.174102`.

Barry R, Clarke A, Johnstone S, Magee C, Rushby J (2007). "EEG Differences Between Eyes-Closed and Eyes-Open Resting Conditions." *Clinical Neurophysiology*, **118**(12), 2765–2773. `doi:10.1016/j.clinph.2007.07.028`.

Batista G, Wang X, Keogh E (2011). "A Complexity-Invariant Distance Measure for Time Series." In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 699–710. Curran Associates, Inc., RedHook, NY.

Brandmaier A (2012). *Permutation Distribution Clustering and Structural Equation Model Trees*. Dissertation, Saarland University, Saarbrücken.

Brandmaier A (2015). ***pdc**: Permutation Distribution Clustering*. R package version 1.0.3, URL `http://CRAN.R-project.org/package=pdc`.

Bruzzo A, Gesierich B, Santi M, Tassinari C, Birbaumer N, Rubboli G (2008). "Permutation Entropy to Detect Vigilance Changes and Preictal States from Scalp EEG in Epileptic Patients. A Preliminary Study." *Neurological Sciences*, **29**(1), 3–9. `doi:10.1007/s10072-008-0851-3`.

Chávez E, Navarro G, Baeza-Yates R, Marroquín J (2001). "Searching in Metric Spaces." *ACM Computing Surveys (CSUR)*, **33**(3), 273–321. `doi:10.1145/502807.502808`.

Cormen T, Leiserson C, Rivest R, Clifford S (2004). *Introduction to Algorithms*. MIT Press, Cambridge, MA.

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008). "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures." *Proceedings of the VLDB Endowment*, **1**(2), 1542–1552. `doi:10.14778/1454159.1454226`.

Elkan C (2003). "Using the Triangle Inequality to Accelerate k-Means." In *International Conference on Machine Learning*, volume 20, pp. 147–153.

Faloutsos C, Ranganathan M, Manolopoulos Y (1994). "Fast Subsequence Matching in Time-Series Databases." In R Snodgrass, M Winslett (eds.), *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pp. 419–429. Minneapolis, MN.

Fisch B, Spehlmann R (1999). *Fisch and Spehlmann's EEG Primer: Basic Principles of Digital and Analog EEG*. Elsevier Science, Amsterdam, The Netherlands.

Frank B, Pompe B, Schneider U, Hoyer D (2006). "Permutation Entropy Improves Fetal Behavioural State Classification Based on Heart Rate Analysis from Biomagnetic Recordings in Near Term Fetuses." *Medical and Biological Engineering and Computing*, **44**(3), 179–187. `doi:10.1007/s11517-005-0015-z`.

Fulcher B, Little M, Jones N (2013). "Highly Comparative Time-Series Analysis: The Empirical Structure of Time Series and Their Methods." *Journal of the Royal Society Interface*, **10**(83), 20130048. `doi:10.1098/rsif.2013.0048`.

Gonzalez R, Woods R (1992). *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts.

Halkidi M, Batistakis Y, Vazirgiannis M (2001). "On Clustering Validation Techniques." *Journal of Intelligent Information Systems*, **17**(2), 107–145. `doi:10.1023/a:1012801612483`.

Hao C, Zhao T (2010). "Regional Differentiation Based on Permutation Entropy and Its Geographical Explanation." In *Proceedings of the Third International Symposium on Computer Science and Computational Technology*, pp. 5–8. Jiaozuo, China.

Johnson S (1967). "Hierarchical Clustering Schemes." *Psychometrika*, **32**(3), 241–254. `doi:10.1007/bf02289588`.

Keogh E, Chu S, Hart D, Pazzani M (2004a). "Segmenting Time Series: A Survey and Novel Approach." In M Last, A Kandel, H Bunke (eds.), *Data Mining in Time Series Databases*, volume 57, pp. 1–22. World Scientific Publishing, Singapore.

Keogh E, Folias T (2002). "The UCR Time Series Data Mining Archive." *Computer Science & Engineering Department, University of California, Riverside, CA.* URL http://www.cs.ucr.edu/~eamonn/time_series_data/.

Keogh E, Kasetty S (2003). "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration." *Data Mining and Knowledge Discovery*, **7**(4), 349–371. doi:10.1145/775047.775062.

Keogh E, Lonardi S, Ratanamahatana C (2004b). "Towards Parameter-Free Data Mining." In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 206–215.

Keogh E, Wei L, Xi X, Vlachos M, Lee SH, Protopapas P (2009). "Supporting Exact Indexing of Arbitrarily Rotated Shapes and Periodic Time Series Under Euclidean and Warping Distance Measures." *The VLDB Journal*, **18**, 611–630. doi:10.1007/s00778-008-0111-4.

Kullback S, Leibler R (1951). "On Information and Sufficiency." *The Annals of Mathematical Statistics*, **22**(1), 79–86. doi:10.1214/aoms/1177729694.

Lehmer D (1960). "Teaching Combinatorial Tricks to a Computer." In *Proceedings of Symposia in Applied Mathematics, Combinatorial Analysis*, volume 10, pp. 179–193. New York.

Lemon J (2006). "**plotrix**: A Package in the Red Light District of R." *R News*, **6**(4), 8–12.

Li M, Chen X, Li X, Ma B, Vitányi P (2004). "The Similarity Metric." *IEEE Transactions on Information Theory*, **50**(12), 3250–3264. doi:10.1109/tit.2004.838101.

Li X, Ouyang G, Richards D (2007). "Predictability Analysis of Absence Seizures with Permutation Entropy." *Epilepsy Research*, **77**(1), 70–74. doi:10.1016/j.eplepsyres.2007.08.002.

Liao T (2005). "Clustering of Time Series Data: A Survey." *Pattern Recognition*, **38**(11), 1857–1874. doi:10.1016/j.patcog.2005.01.025.

MacQueen J (1967). "Some Methods for Classification and Analysis of Multivariate Observations." In M Cam, J Neyman (eds.), *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University of California Press, Berkeley, CA.

Montero P, Vilar J (2014). "**TSclust**: An R Package for Time Series Clustering." *Journal of Statistical Software*, **62**(1), 1–43. doi:10.18637/jss.v062.i01.

Moore A (2000). "The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data." In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 397–405. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Nair U, Krishna B, Namboothiri V, Nampoori V (2010). "Permutation Entropy Based Real-Time Chatter Detection Using Audio Signal in Turning Process." *International Journal of Advanced Manufactoring Technology*, **46**, 61–68. doi:10.1007/s00170-009-2075-y.

Nicolaou N, Georgiou J (2011). "The Use of Permutation Entropy to Characterize Sleep Electroencephalograms." *Clinical EEG and Neuroscience*, **42**(1), 24–28. doi:10.1177/155005941104200107.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Shen W, Wang Y, Bai X, Wang H, Latecki L (2013). "Shape Clustering: Common Structure Discovery." *Pattern Recognition*, **46**, 539–550. doi:10.1016/j.patcog.2012.07.023.

Wang X, Smith K, Hyndman R, Alahakoon D (2004). "A Scalable Method for Time Series Clustering." *Technical report*, Monash University, Australia.

Zhang D, Lu G (2004). "Review of Shape Representation and Description Techniques." *Pattern Recognition*, **37**(1), 1–19. doi:10.1016/j.patcog.2003.07.008.

A. Squared Hellinger distance approximating KL divergence

The squared Hellinger distance is the second-order approximation of the KL divergence:

$$\begin{aligned}
-\sum_{\pi\in S_m} p_\pi \log\frac{p_\pi}{q_\pi} &= 2\sum_{\pi\in S_m} p_\pi \log\sqrt{\frac{q_\pi}{p_\pi}} = 2\sum p_\pi \log\left[1+\left(\sqrt{\frac{q_\pi}{p_\pi}}-1\right)\right] \\
&\cong 2\sum p_\pi\left[\left(\sqrt{\frac{q_\pi}{p_\pi}}-1\right)-\frac{1}{2}\left(\sqrt{\frac{q_\pi}{p_\pi}}-1\right)^2\right] \\
&= 2\sum\left[\sqrt{q_\pi p_\pi}-p_\pi\right]-\sum p_\pi\left(\frac{\sqrt{q_\pi}-\sqrt{p_\pi}}{\sqrt{p_\pi}}\right)^2 \\
&= 2\sum\left(\sqrt{p_\pi q_\pi}\right)-2-\sum\left(\sqrt{q_\pi}-\sqrt{p_\pi}\right)^2 \\
&= \left(2-\sum\left(\sqrt{q_\pi}-\sqrt{p_\pi}\right)^2\right)-2-\sum\left(\sqrt{q_\pi}-\sqrt{p_\pi}\right)^2 = -2\sum\left(\sqrt{q_\pi}-\sqrt{p_\pi}\right)^2
\end{aligned}$$

B. Likelihood-ratio between codebooks

Let $L\left(x|P\right)$ be the multinomial likelihood of model P given observations x:

$$L(x|P) = \left(\frac{N!}{\prod_{\pi\in S_m} x_\pi}\right)\prod_{\pi\in S_m} p_\pi^{x_\pi} = N!\prod_{\pi\in S_m}\frac{p_\pi^{x_\pi}}{x_\pi!}.$$

The log-likelihood is thus:

$$-LL(x|P) = -\log(N!) - \sum_{\pi\in S_m} x_\pi \log\left(p_\pi\right) + \sum_{\pi\in S_m}\log\left(x_\pi!\right).$$

The log-likelihood ratio is:

$$\begin{aligned}
\Lambda(x|P,Q) = \log\left(\frac{-L\left(x|P\right)}{-L\left(x|Q\right)}\right) &= \left(-LL\left(x|P\right)\right)-\left(-LL\left(x|Q\right)\right) \\
&= -\log(N!) - \sum_{\pi\in S_m} x_\pi\log\left(p_\pi\right) + \sum_{\pi\in S_m}\log\left(x_\pi!\right) \\
&\quad + \log(N!) + \sum_{\pi\in S_m} x_\pi\log\left(q_\pi\right) - \sum_{\pi\in S_m}\log\left(x_\pi!\right) \\
= \sum_{\pi\in S_m} x_\pi\log\left(q_\pi\right) - \sum_{\pi\in S_m} x_\pi\log\left(p_\pi\right) &= -\sum_{\pi\in S_m} x_\pi\log\left(\frac{q_\pi}{p_\pi}\right) = \sum_{\pi\in S_m} x_\pi\log\left(\frac{p_\pi}{q_\pi}\right)
\end{aligned}$$

C. Permutation index

The following fragment shows explicit, nested if-clauses for retrieving the permutation index of an array of size 3, implemented in the Python language.

```
function y = get_codeword(x)
 if x(3) < x(1)
        if x(2) < x(1)
                if x(3) < x(2)
                        y=1;
                else
                        y=2;
                end
        else
                y=3;
        end
 else
        if x(2) < x(1)
                y=4;
        else
                if x(3) < x(2)
                        y=5;
                else
                        y=6;
                end
        end
 end
```

Affiliation:

Andreas M. Brandmaier
Center for Lifespan Psychology
Max Planck Institute for Human Development
Lentzeallee 94
14195 Berlin, Germany
E-mail: brandmaier@mpib-berlin.mpg.de
URL: http://www.brandmaier.de/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2015, Volume 67, Issue 5 *Submitted:* 2014-02-25
doi:10.18637/jss.v067.i05 *Accepted:* 2014-12-28