Circular Silhouette and a Fast Algorithm

Yinong Chen, Tathagata Debnath, Andrew Cai, and Mingzhou Song*

Abstract—Circular data clustering has recently been solved exactly in sub-quadratic time. However, the solution requires a given number of clusters; methods for choosing this number on linear data are inapplicable to circular data. To fill this gap, we introduce the circular silhouette to measure cluster quality and a fast algorithm to calculate the average silhouette width. The algorithm runs in linear time to the number of points on sorted data, instead of quadratic time by the silhouette definition. Empirically, it is over 3000 times faster than by silhouette definition on 1,000,000 circular data points in five clusters. On simulated datasets, the algorithm returned correct numbers of clusters. We identified clusters on round genomes of human mitochondria and bacteria. On sunspot activity data, we found changed solar-cycle patterns over the past two centuries. Using the circular silhouette not only eliminates the subjective selection of number of clusters, but is also scalable to big circular and periodic data abundant in science, engineering, and medicine. The resulting software package 'CircularSilhouette' is open source, freely available at https://cran.r-project.org/package=CircularSilhouette

Index Terms—Circular clustering, silhouette, circular genome, bacterial genome, mitochondria, periodic data, solar cycle

1 Introduction

Circular and periodic data are abundant. Synthetic aperture radar [1], wind direction measurement [2], and 24-hour emergency room arrival time [3] are sources of circular data. Bacterial, mitochondrial, and chloroplast genomes are also circular. Periodic activity is observed from circadian rhythms of living organisms to solar cycles. Circular data clustering reveals recurrent patterns. It is a two-dimensional special case of the spherical k-means problem using the cosine distance [4]. Its solutions had been heuristic [5], [6], [7], [8], [9], [10], [11], until Debnath and Song [12] designed a fast and optimal circular clustering (FOCC) algorithm that exactly solves the problem in time linear-polylogarithmic to the number of points. The mean shift-based methods [5], [6] cluster circular data by locating density function maxima. The solution of Abraham et al. [9] uses simulated annealing. Lagona et al. [8] employ the hidden Markov model and the von Mises distribution; they also optimized mixture probabilistic models of multivariate circular and linear data via expectation maximization (EM) [7]. Most solutions are based on the EM algorithm and continue to appear [10], [11]. However, the user is often required to specify the number of clusters, which can be subjective. As this figure reveals vital properties about the grouping patterns in circular data, it is desirable to select this number objectively, optimally, and quickly.

To our knowledge, the literature lacks any method to detect the number of clusters in circular or periodic data, despite many approaches for linear data. Rousseeuw first introduced the silhouette, applicable to assessing the cluster quality of one-dimensional linear data [13]. The number of clusters can also be detected by elbow methods [14], penalized likelihood criteria such as AIC, BIC, DIC [15], information-theoretic approaches [16], and physics-based models like EF-Index [17]. However, most methods are not easily applicable to circular data because calculation of either distance- or density-based criteria will be different on a circle from along a line.

Here, we introduce the circular silhouette and design a fast algorithm to calculate the average silhouette width; subsequently we use it to detect the number of circular clusters. We expand the original silhouette to cover clustered circular data. The distance between two points on a circle creates complications for computing the circular silhouette. We handle them with guaranteed correctness in linear time on sorted data. We evaluated the algorithm on simulated data of various distributions in contrast to the elbow method. Our algorithm, but not the elbow method, correctly returned numbers of clusters in the simulated data. Next, we applied it on circular genomes of human mitochondria, *Candidatus* Carsonella ruddii, and *Lactobacillus curieae*. We also applied the circular silhouette algorithm on sunspot activity data to estimate the period of solar cycle, traditionally known to be about 11 years [18]. Our finding suggests that the period has reduced by about seven months during the past two centuries. At solar cycle peaks, the sun can radiate energy that affects lifespan via mutating genomes [19], and a faster cycle may imply a stronger negative impact on human life in the foreseeable future. Such examples demonstrate how the circular silhouette may contribute to scientific discoveries.

The circular silhouette algorithm, together with the FOCC algorithm [12], can efficiently determine an optimal circular clustering. With manual tuning eliminated, the clustering process is no longer subjective. This advance in the capacity of circular data analysis expands the range of applications to gigantic sample sizes.

- *Y. Chen is with Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD 21218, United States.*
- *T. Debnath and M. Song are with Department of Computer Science, New Mexico State University, Las Cruces, NM 88003, United States.*
- *A. Cai was with School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, United States. He is currently with Apple Inc., Austin, TX 78746, United States.*
- *M. Song is also with Molecular Biology and Interdisciplinary Life Sciences Graduate Program, New Mexico State University, Las Cruces, NM 88003, United States. E-mail: joemsong@nmsu.edu*

: Corresponding author.

2 METHODS

2.1 The circular silhouette

We view circular points as one-dimensional points periodically distributed along a line. Let x and y be coordinates of two points along the circle of circumference L. Assuming that x and y, modulo L, fall within $[0, L)$, we define the *circular distance* between the two points as

$$d(x, y) = \min\{|y - x|, L - |y - x|\} \tag{1}$$

The silhouette information [13] assesses cluster quality by compactness within and separation between clusters. It has been widely used to determine the optimal number of clusters. The *silhouette width* of object u_n among N objects is defined as [13]

$$S_n = \frac{b_n - a_n}{\max\{a_n, b_n\}}, \quad n = 1, \ldots, N \tag{2}$$

where a_n is the *average intra-cluster dissimilarity* of object u_n to all other object u_i in the same cluster with m objects:

$$a_n = \frac{1}{m-1} \sum_{i=0}^{m-1} d(u_n, u_i), \quad m > 1 \tag{3}$$

If $m = 1$, $a_n = 0$. b_n is the *average inter-cluster dissimilarity* of object u_n to all objects v_i in a cluster nearest to u_n:

$$b_n = \min_{k \in \{0,\ldots,K-1\} - \{k(n)\}} \frac{1}{W_k} \sum_{v_i \in \text{ cluster } k} d(u_n, v_i) \tag{4}$$

where $k(n)$ is the cluster of object u_n and W_k is the number of objects in cluster k among K clusters. The *average silhouette width* is defined by

$$\bar{S} = \frac{1}{N} \sum_{n=1}^{N} S_n \tag{5}$$

which is a statistic of overall cluster quality, often used to determine the number of clusters.

We quantify the dissimilarity of two circular points by their circular distance defined in Eq. (1). To compute a_n and b_n by definition, one needs to go through all other points in the cluster of point u_n and all points in its nearest cluster. As a result, it takes linear time of the two cluster sizes to calculate S_n for a single point u_n. Overall, to compute all S_n for $n = 1, \ldots, N$ takes quadratic time $\mathcal{O}(N^2)$.

2.2 A fast circular silhouette algorithm

Now we introduce a fast algorithm to calculate circular silhouette information in worst-case log-linear time. Instead of independently computing intra- and inter-cluster distances for each object, we update the distances of an object based on those of a neighboring object using respective recurrence equations. Although one such update may take more than constant time, our algorithms run in amortized constant time for each object, attaining overall linear time on sorted data.

Let O be an array of N circular data points. Let C be the cluster labels of each point in O. Given point n, O_n is its coordinate on the circle and C_n is its cluster label. Let L be the circumference of the circle. We linearize the original input data by sorting O to $X = (x_0, \ldots, x_{N-1})$ with cluster labels C rearranged accordingly into $Y = (y_0, \ldots, y_{N-1})$. We calculate intra-cluster distance a_n and inter-cluster distance b_n for each point and save them in arrays $\mathbf{a}$ and $\mathbf{b}$.

Algorithm 1 CIRCULAR-SILHOUETTE is the fast circular silhouette algorithm. The input is circular data O, cluster labels C, and circumference L. The output is the average silhouette width over the N points. It first sorts, extends, and shifts the original data O and C into linearized arrays X and Y. The purpose of lines 7–8 is to shift X to avoid calculation of the mean for any cluster that crosses the origin. Then it calls Algorithm 2 to calculate the average silhouette width over all points.

Algorithm 1 CIRCULAR-SILHOUETTE(O, C, L)

1: N is the number of points in array O
2: $O \leftarrow O \mod L$
3: Array I: indices to elements in O, arranged in increasing order
4: Sort O in order of $O[I[i]] < O[I[i+1]]$ for $i = 0, \ldots, N-1$
5: Linearize: $X \leftarrow O[I[0]], \ldots, O[I[N-1]]$; $Y \leftarrow C[I[0]], \ldots, C[I[N-1]]$
6: Extend: $X \leftarrow X, X[0] + L, \ldots, X[N-1] + L$; $Y \leftarrow Y, Y$
7: Find the first point of the first cluster as $X[i] = Z$.
8: $X \leftarrow (X[i] - Z, X[i+1] - Z, \ldots, X[i+N-1] - Z)$
9: $Y \leftarrow (Y[i], Y[i+1], \ldots, Y[i+N-1])$
10: $\bar{S} \leftarrow$ CIRCULAR-SIL-SORTED(X, Y, L)
11: **return** average silhouette width $\bar{S}$

Algorithm 2 CIRCULAR-SIL-SORTED calculates the average silhouette width over all sorted and shifted points. The input includes linearized data $X = (x_0, \ldots, x_{N-1})$, the cluster of each point $Y = (y_0, \ldots, y_{N-1})$, and the circumference L. The output is the average silhouette width $\bar{S}$ of the N points. For each cluster k, it calculates intra- and inter-cluster distances, respectively. Inter-cluster distances from cluster k to its two neighbor clusters are computed. For each point in cluster k, the smaller of the two inter-distances is assigned to b_n. Then it uses the intra- and inter-distances to calculate the average silhouette width over all points.

Algorithm 2 CIRCULAR-SIL-SORTED(X, Y, L)

K: the number of clusters; N: the number of points
$W[k]$: the number of points in cluster k obtained from Y
$I[k]$: index to the first point of cluster k obtained from Y
$\mu[k]$: the mean of cluster k from X and Y
$Sil \leftarrow 0$
for $k \leftarrow 0$ *to* $K - 1$ **do**
    $\mathbf{a} \leftarrow$ INTRACLUSTERDIST(Points in cluster k, L)
    k_l cluster label to the left side of cluster k
    k_r cluster label to the right side of cluster k
    $\mathbf{b}_l \leftarrow$ INTERCLUSTERDIST(X, L, I, W, μ, k, k_l)
    if $k_r \neq k_l$ **then**
        $\mathbf{b}_r \leftarrow$ INTERCLUSTERDIST(X, L, I, W, μ, k, k_r)
    else
        $\mathbf{b}_r \leftarrow \mathbf{b}_l$
    end if
    for $n \leftarrow 0$ *to* $W[k] - 1$ **do**
        $b[n] \leftarrow \min(b_l[n], b_r[n])$
        $Sil \leftarrow Sil + (b[n] - a[n]) / \max(a[n], b[n])$
    end for
end for
$\bar{S} \leftarrow Sil/N$
return average silhouette width $\bar{S}$

2.3 Calculating intra-cluster distances

Now we calculate intra-cluster distance a_n for all m points within a given cluster. For clarity, we shift the m points

to start at zero represented by $\mathbf{u} = (u_0, \ldots, u_{m-1})$, where u_0=0. a_n is evidently shift-invariant.

Let A_n be the sum of distances between u_n and all other points in the cluster. If the cluster is within half a circle (case 1), one can compute A_n using linear distance; otherwise (case 2), one must use circular distance.

Case 1. The cluster range is less than $L/2$. By definition,

$$\begin{aligned} A_n &= \sum_{i=0}^{n-1}(u_n - u_i) + \sum_{i=n+1}^{m}(u_i - u_n) \\ &= (2n - m + 1)u_n + \sum_{i=n+1}^{m-1} u_i - \sum_{i=0}^{n-1} u_i \quad (n > 0) \end{aligned} \tag{6}$$

with $A_0 = \sum_{i=0}^{m-1}(u_i - u_0)$. Equivalently, A_n can be computed from A_{n-1} in constant time by recurrence equation:

$$A_n = A_{n-1} + (2n - m)(u_n - u_{n-1}) \tag{7}$$

which is derived in Supplementary Materials 1.1.

Case 2. The cluster range is at least $L/2$. We cut points before and after $L/2 + u_n$ by a line into two groups. The two points bordering the cut line can be progressively updated. Using both points, A_n can be updated from A_{n-1} in amortized constant time, instead of going over all the points, as illustrated in Fig. 1.

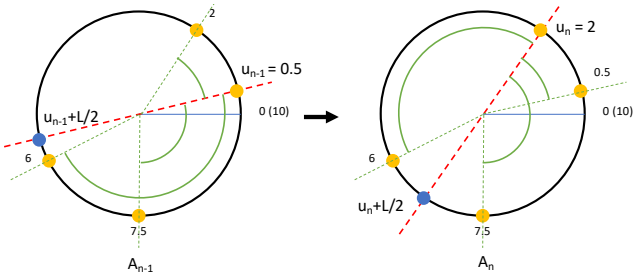

Fig. 1: **Updating intra-cluster distance sums.** Four points (yellow) are in a cluster on a circle of circumference $L = 10$. Green arcs are the distance from current point to other points. At $u_{n-1} = 0.5$, the corresponding A_{n-1} is the length sum of all green arcs in the left circle. Moving from $u_{n-1} = 0.5$ to $u_n = 2$, we can calculate the distance sum A_n from A_{n-1} instead of going over all four points. In the right circle, the cut line (red dashed) from u_n to $u_n + L/2$ (blue point) divides the points to two sides. Within each side, the distance sum can be updated consistently. Going from $n-1$ to n, the blue point will move and some yellow points (e.g., 6) may change side, which is accounted for in recurrence equation (10).

Here is a summary of the main steps:

1) Append $\mathbf{u} + L$ to $\mathbf{u}$ to have an array of $2m$ points: $(u_0, \ldots, u_{m-1}, u_0 + L, \ldots, u_{m-1} + L)$. $h(n)$ is the halfway index between n and $n+m$ such that $u_{h(n)-1} < u_n + L/2 \leq u_{h(n)}$. There are $h(n) - n$ points from u_n to $u_{h(n)-1}$ and $m - h(n) + n$ points from $u_{h(n)}$ to u_{m+n-1}.

2) Calculate intra-cluster distance sum A_0 for point $u_n = 0$ by the circular distance definition in Eq. (1):

$$\begin{aligned} A_0 &= \sum_{i=0}^{h(0)-1} u_i + \sum_{i=h(0)}^{m-1}(L - u_i) \\ &= [m - h(0)]L + \sum_{i=0}^{h(0)-1} u_i - \sum_{i=h(0)}^{m-1} u_i \end{aligned} \tag{8}$$

3) Calculate intra-cluster distance sum A_n for other points u_n $(n > 0)$ within the cluster:

$$A_n = \sum_{i=n}^{h(n)-1}(u_i - u_n) + \sum_{i=h(n)}^{n+m-1} L - (u_i - u_n) \tag{9}$$

Equivalently, A_n can be computed from A_{n-1} by

$$\begin{aligned} A_n = A_{n-1} &+ \left[2\sum_{i=h(n-1)}^{h(n)-1} u_i\right] + [2n + m - 2h(n)]u_n \\ &+ [2h(n-1) - m - 2n]u_{n-1} + [h(n-1) - h(n)]L \end{aligned} \tag{10}$$

which is derived in Supplementary Materials 1.2.

Next, we calculate the intra-cluster distance $a_n = A_n/(m-1)$ for every point in the cluster for both cases.

These steps give rise to Supplementary Algorithm S1 IntraClusterDist to calculate intra-cluster distances of points in a given cluster. The input is shifted points in the cluster $\mathbf{u} = (0, u_1, \ldots, u_{m-1})$ and the circumference L. The output $\mathbf{a} = (a_0, \ldots, a_{m-1})$ is the average intra-cluster distances from every u_n to all other points in the cluster. The intra-cluster distance is calculated in two cases as discussed above. Under each case, the intra-cluster distance for each point is calculated based on the previous point in amortized constant time. The overall runtime is thus linear to the number of points in the cluster.

2.4 Calculating inter-cluster distances

Now, we calculate inter-cluster distances. Let $\mathbf{u}$ contain all m points in cluster k: $\mathbf{u} = (u_0, \ldots, u_{m-1})$, where $u_0 = 0$. For point u_n, let cluster k' contain q points $\mathbf{v} = (v_0, \ldots, v_{q-1})$, where $\mathbf{v} \geq 0$. We precompute the means of each cluster as μ_k for $k = 0, \ldots, K-1$. Let B_n be the inter-cluster distance sum for u_n. We draw a cut line from u_n to $u_n + L/2$ to divide points in $\mathbf{v}$ into two sides, each inside half a circle (Fig. 2). We consider the inter-cluster distance in two cases:

Case 1. If cluster k' points are all on one side of the cut line from u_n, $b_n = \min\{|u_n - \mu_{k'}|, L - |u_n - \mu_{k'}|\}$—the circular distance between u_n and $\mu_{k'}$, the mean of cluster k', derived in Supplementary Materials 2.1.

Case 2. If some points in cluster k' are on the opposite sides of the cut line, we will calculate B_n by updating from B_{n-1}.

We summarize the algorithm into four steps:

1) Let $h(n)$ be the halfway index between 0 to $q-1$ such that $v_{h(n)-1} < u_n + L/2 \leq v_{h(n)}$. There are $h(n)$ points from v_0 to $v_{h(n)-1}$ and $q - h(n)$ points from $v_{h(n)}$ to v_{q-1}.

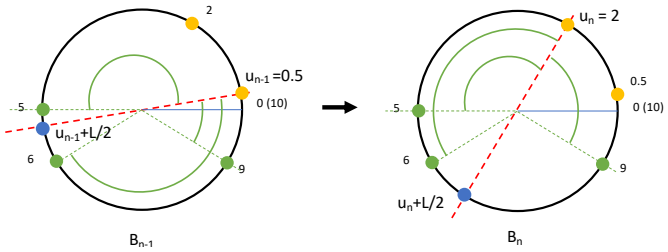

Fig. 2: **Updating inter-cluster distance sums.** A current cluster (two yellow points) and another cluster (three green points) are on a circle of circumference $L = 10$. For the left circle, B_{n-1} is the inter-cluster distance sum from point $u_{n-1} = 0.5$ to all green points, as indicated by the green arcs. Moving from $u_{n-1} = 0.5$ to $u_n = 2$, we can calculate the distance sum B_n from B_{n-1} instead of going over all three points. For the right circle, the cut line (red dashed) from u_n to $u_n + L/2$ (blue point) divides the green points to two sides. Within each side, the distance sum can be updated quickly. Going from $n-1$ to n, the blue point will move and some yellow points (e.g., 6) may change side and others stay (e.g., 5 and 9), which are accounted for in recurrence equation (13).

2) Calculate inter-distance sum B_0 from $u_0 = 0$, the first point in cluster k, to all points in cluster k':

$$\begin{aligned} B_0 &= \sum_{i=0}^{h(0)-1} v_i + \sum_{i=h(0)}^{q-1} L - v_i \\ &= [v - h(0)]L + \sum_{i=0}^{h(0)-1} v_i - \sum_{i=h(0)}^{q-1} v_i \end{aligned} \tag{11}$$

which is calculated regardless of u_0 being case 1 or case 2.
3) Calculate inter-cluster distance sum B_n from u_n ($n > 0$) to cluster k'. Rather than by definition—linear time in q:

$$B_n = \sum_{i=0}^{h(n)-1} (v_i - u_n) + \sum_{i=h(n)}^{q-1} L - (v_i - u_n) \tag{12}$$

we compute B_n from B_{n-1} in amortized constant time iteratively as derived in Supplementary Materials 2.2:

$$\begin{aligned} B_n = B_{n-1} &+ \left[2 \sum_{i=h(n-1)}^{h(n)-1} v_i \right] + [q - 2h(n)]u_n \\ &+ [2h(n-1) - q]u_{n-1} + [h(n-1) - h(n)]L \end{aligned} \tag{13}$$

4) Calculate inter-cluster distance $b_n = B_n/q$ for each point in cluster k.

These steps constitute Supplementary Algorithm S2 INTERCLUSTERDIST that computes the inter-cluster distance from each point in cluster k to cluster k'. The input includes linearized data $X = (x_0, \ldots, x_{N-1})$, the circumference L, cluster start index I, cluster size W, cluster means μ, and cluster labels k and k'. The output $\mathbf{b} = (b_0, \ldots, b_{m-1})$ is average inter-cluster distances from u_n in cluster k to all points in cluster k'. Each inter-cluster distance is calculated in two cases aforementioned: either directly or based on the previous point in amortized constant time. Case 1, a special situation of case 2, is included for a lower overhead than using case 2. A cluster can have both case 1 and case 2 points. Upon encountering a case 2 point for the first time, B_0 must be calculated.

2.5 Time complexity

We have argued that computing the silhouette by definition takes $\mathcal{O}(N^2)$ time in the end of section 2.1. In contrast, the fast circular silhouette algorithms reduce the runtime to be sub-quadratic in N.

Theorem 1. *Algorithm 1* CIRCULAR-SILHOUETTE *takes $\mathcal{O}(N \log N)$ time on unsorted data; Algorithm 2* CIRCULAR-SIL-SORTED *takes $\mathcal{O}(N)$ time on sorted data.*

Proof. Supplementary Lemma S1 shows that the intra-cluster distance calculation by Alg. S1 takes $\Theta(m)$ time for a cluster of m points, or amortized constant time $\mathcal{O}(1)$ for each point. In Supplementary Lemma S2, we derive $\mathcal{O}(q+m)$ time for Alg. S2 to compute inter-cluster distances between two given clusters of m and q points, respectively, with amortized constant time $\mathcal{O}(1)$ for each point in the two clusters. Then we establish the stated runtime for Alg. 1 and Alg. 2 in Supplementary Materials section 3.3. □

3 RESULTS

To demonstrate the effectiveness of circular silhouette and the efficiency of the fast algorithm, we show empirical runtime, number-of-cluster selection correctness, and an application in period finding. First we will compare the runtime of the algorithm with the algorithm using the circular silhouette definition on both simulated and real data. Then we will examine the correctness of the detected number of clusters by maximizing the average circular silhouette width on simulated data. We also apply the algorithms to identify molecular events on circular genomes. Lastly, we will estimate the period of solar cycle using the circular silhouette.

3.1 Empirical runtime

Here, we evaluate the runtime of the fast circular silhouette algorithm along side the silhouette algorithm by definition. Both are implemented in the function `circular.sil()` in the R package 'CircularSilhouette', where the former algorithm is specified by setting the method argument to `"linear"` and the latter by `"quadratic"`. The latter implementation does not exhaustively find the minimum inter-cluster distance as given in Eq. (4). Instead, it used only the two clusters immediately adjacent to the current cluster, theoretically having the same worst-case runtime with but practically faster than using the definition in Eq. (4).

We simulated circular data from a Gaussian mixture model wrapped around a circle with a circumference of 1000. The model has five components with means of 0, 200, 400, 600, and 800 and a variance of 1. We used the `rnorm()` function in R to generate data points for each component, merged the data points, and performed an operation of modulo 1000 to derive circular data. Each point is assigned a cluster label by the component it originated from.

We compared the runtime of the two algorithms on five sets of simulated data of 200,000, 400,000, 600,000, 800,000, and 1,000,000 points, respectively. We used a computer with AMD Ryzen 7 5800H with Radeon Graphics at 3.20 GHz and 16 GB RAM. Figure 3a shows the runtime as a function of the number of points with fixed five clusters. The runtime of the fast algorithm grows slowly, while the silhouette-by-definition algorithm becomes much slower with increasing numbers of points. At one million points, the fast algorithm took 0.3291 second; the algorithm by definition spent 1051 seconds. So the fast algorithm achieved a speedup of 3,193 times. Figure 3b displays the runtime as a function of the number of clusters for the two algorithms, at fixed 102,400 points. Both algorithms ran faster as the number of clusters increased. This counter-intuitive saving is because the number of point pairs involved in computing both intra- and inter-cluster distances is likely to decrease as the number of clusters increases. For example, computing circular silhouette on two clusters took 0.32 second but only 0.029 second on 1024 clusters using the fast algorithm. Between the two algorithms, the fast algorithm's time saving is most pronounced at smaller numbers of clusters. In summary, the fast circular silhouette algorithm is at an evident advantage on large datasets.

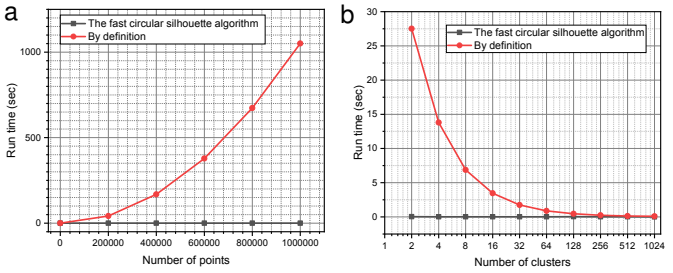

Fig. 3: **The runtime of the fast algorithm versus the algorithm by circular silhouette definition.** (a) With an increasing number of circular data points in five clusters, the runtime of the fast algorithm grows sub-quadratically, whereas the growth rate of the slow algorithm is quadratic. (b) At fixed 102,400 points, the runtime of both algorithms decreases with increasing numbers of clusters, with the fast algorithm being quicker to complete.

3.2 Selecting the number of clusters on simulated data

We evaluate the performance of the fast circular silhouette algorithm on three simulated datasets where the numbers of circular clusters are known. To select an optimal number of clusters on a given dataset, we run FOCC and then calculate the circular silhouette on the resulting clustering for each given k. The k that maximizes the average circular silhouette width is declared as an optimal number of clusters for the given dataset. The algorithm is implemented as R function `find.num.of.clusters()` in the 'CircularSilhouette' package. We also used the elbow method to select the number of clusters based on the within-cluster sum of squared distances (WSS) on linearized circular data. The knee point is determined by the 'kneedle' algorithm [20], as implemented in R package 'etam4260/kneedle' on GitHub. We evaluate their performance by comparing detected numbers of clusters with ground-truth numbers of clusters.

The first dataset was manually created with nine well-separated clusters (Fig. 4a). The maximum silhouette width is achieved at $K = 9$ clusters (Fig. 4d), but the elbow method incorrectly detected 5 clusters (Fig. 4g).

The second simulated dataset includes five clusters of points (Fig. 4b) randomly generated from five von Mises distributions. The circumference is 360. The means of each cluster are 0, 72, 144, 216, 288, respectively. The correct number of clusters (5) is successfully detected by both the circular silhouette (Fig. 4e) and the elbow method (Fig. 4h).

The third dataset has 17 clusters (Fig. 4c). Clusters 4, 6, 9, and 13 are normally distributed. Clusters 3, 5, 14, and 17 are gamma distributed. Clusters 1, 2, and 12 are exponentially distributed. Clusters 8, 15, and 16 are beta distributed. Clusters 7 and 11 are Student's t-distributed. The points were transformed to be circular by modulo the circumference of 600. Using the circular silhouette, we properly identified the number of clusters (Fig. 4f), but the elbow method failed again (Fig. 4i).

Therefore, we conclude that the circular silhouette can be properly used to select numbers of clusters on data with well-separate clusters and is a better option than the elbow method especially for large numbers of clusters.

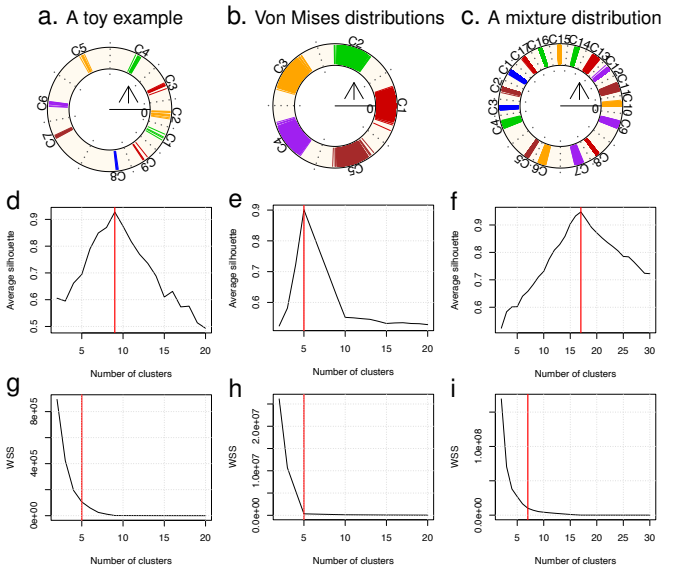

Fig. 4: **Circular silhouette versus an elbow method to select number of clusters.** (a) A toy example with 9 clusters. (b) Simulated data from 5 von Mises distributions. (c) Simulated data from a mixture with 17 components. (d–f) The average silhouette width is maximized (red vertical lines) at (d) 9 clusters for the toy example, (e) 5 clusters for the von Mises data, and (f) 17 clusters for the mixture data. (g–i) The elbow method detected knee points at (g) 5 (wrong), (h) 5, and (i) 7 (wrong) clusters for each dataset, indicated by the red lines.

3.3 Cluster patterns in round genomes

Next, we use the fast circular silhouette algorithm to discover patterns in round genomes. We first studied the CpG

sites on the human mitochondrial genome [21] and the *Candidatus* Carsonella ruddii strain BT chromosome. Their respective clusters are shown in Fig. 5. It is apparent that the 435 CpG sites are quite scattered on human mitochondria, falling into 70 clusters; in contrast, the 524 CpG sites along the BT chromosome are concentrated in 16 clusters.

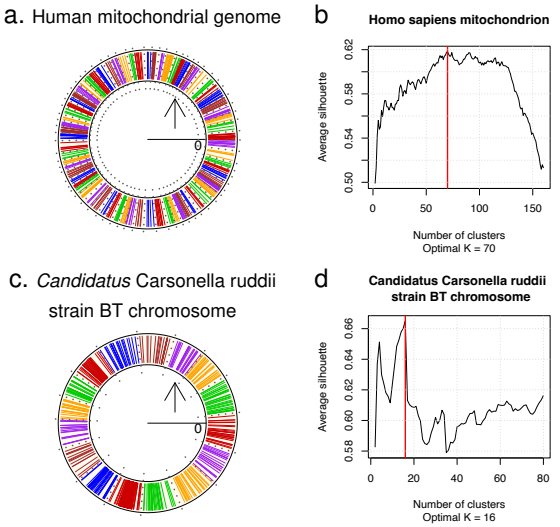

Fig. 5: **CpG-site clusters around two circular genomes.** (a) CpG sites group into 70 clusters in the human mitochondrial genome. (b) Average silhouette width, as a function of number of clusters, is maximized at 70 clusters (red line). (c) CpG sites on chromosome BT of the *Candidatus* Carsonella ruddii strain form 16 clusters. (d) The estimated number of clusters 16 maximizes average silhouette width (red line).

We also examined the start locations of 232 genes along the *Candidatus* Carsonella ruddii strain BT chromosome [22] and 2010 genes along the bacterium *Lactobacillus curieae* genome [23]. Although both have similar numbers of gene clusters (38 versus 39), the gene clusters are more uneven in the former genome than the latter genome (Fig. 6). The cluster unevenness can also be indicated by the sharp peaks in the curve that shows the average silhouette width as a function of number of clusters.

3.4 Estimating the period of sunspot activity

Although the period of sunspot activity was estimated to be about 11 years [18], we used circular silhouette to identify an optimal period from public data with a higher precision and also studied the trend of any change in the period. The algorithm is implemented as R function `estimate.period()` in the 'CircularSilhouette' package. Using this algorithm, we have estimated the average period of sunspot activity to be about 10 years and 313 days in the past 200 years. Most strikingly, we find that the period reduced by about seven months over the past two centuries.

Daily numbers of sunspots recorded from 1818 to 2019 [24] were used in our analysis (Fig. 7a). We converted the longitudinal sunspot activity data to circular data using a range of circumferences, between 2×10 and 2×12 years to include at least two sunspot activity peaks on the circle. Given a circumference, we use the FOCC algorithm and

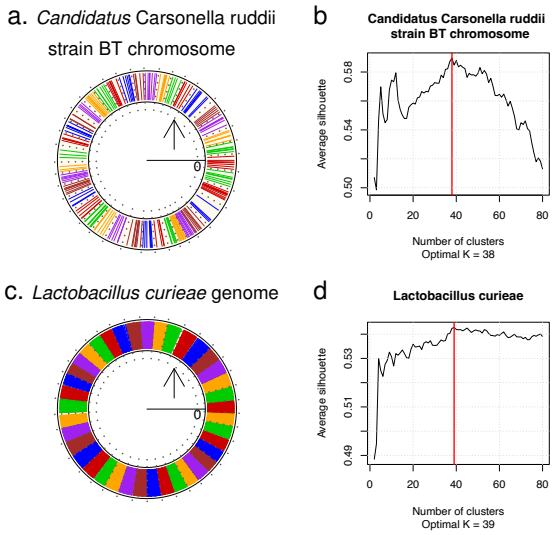

Fig. 6: **Gene clusters by start locations in two circular bacterial genomes.** (a) The gene locations of the *Candidatus* Carsonella ruddii strain BT chromosome. (b) The number of clusters that maximizes silhouette width is 38 (red line). (c) Gene locations in the *Lactobacillus curieae* genome. (d) The optimal number of clusters is 39 (red line).

circular silhouette to find the optimal number of clusters in the range of 2–12. For a given number of clusters, we tried circumferences between 20 and 24 years at a resolution of 0.2 year. Figure 7b identified the maximum silhouette width at two clusters. So, we estimated the sunspot activity period by dividing the optimal circumference by two.

Using the same range of 10 to 12 years for the period, we increased the circumference resolution to 0.1 year to get an optimal period of 10.85 years (Fig. 8a). To further improve the precision, we used an even higher circumference resolution of 0.02 year in the range between 10.70×2 and 11×2 in Fig. 8b. Then we obtained a more precise period estimate of 10.857 years, which is about 10 years and 313 days.

To examine the cluster qualify, we superpose the sunspot data over two periods of the estimated solar cycle. Figure 8c shows a histogram of the data in a 1D Cartesian coordinate system; Figure 8d shows a circular histogram of the data in a polar coordinate system. Evidently, the clusters are well overlapped within and separate inbetween, suggesting a strong periodic signal of sunspot activity.

Next, we studied the sunspot activity trend by 100-year sliding windows. Specifically, we estimated the periods of sunspot activity in the early, middle, and late 100 years to be 11.239 (Fig. 8c), 10.814 (Fig. 8d), and 10.632 (Fig. 8e) years, respectively. The reduction in period from the early to late 100 years is 221 days, more than seven months, or 5.4% of the period of the early 100 years. As the last sunspot peak was observed in April 2014, our finding suggests that it is possible that the next peak would occur in 2024, instead of July 2025—the forecast by National Weather Service.

4 Discussion

Only until recently, a sub-quadratic-time exact solution to circular data clustering becomes available [12]. The circular

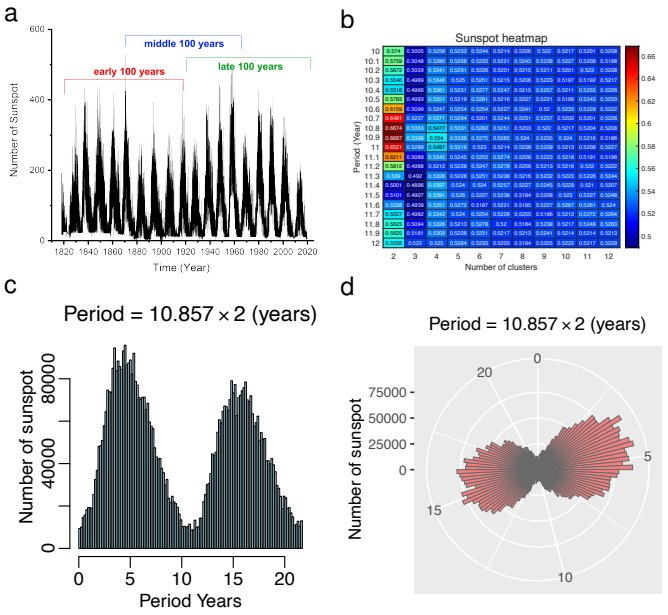

Fig. 7: **Sunspot activity data, circular silhouettes using various periods and numbers of clusters, and optimal clusters.** (a) Numbers of sunspot activity as a function of date. (b) The heat map shows silhouette widths corresponding to combinations of number of clusters (horizontal axis) and period (vertical axis). The color gradient legend maps to silhouette width. (c) Evident periodic sunspot signals as shown by a 128-bin histogram of the data in a one-dimensional Cartesian coordinate system in two solar cycles (10.857×2 years). (d) A strong circular sunspot signal captured by a 128-bin circular histogram in a polar coordinate system with a circumference of 10.857×2 years.

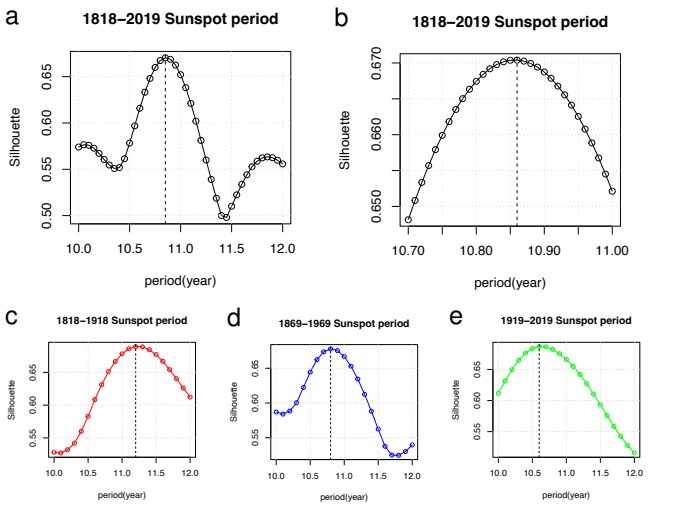

Fig. 8: **Estimating the period of sunspot activity over the last two centuries from 1818 to 2019.** (a) Maximum silhouette widths as a function of period between 10 and 12 years at a circumference resolution of 0.1 year obtained on data from 1818 to 2019. (b) Zooming into the peak region between 10.70 and 11 years in (a), we calculated maximum silhouette widths at a circumference resolution of 0.02 year on the same 200-year data. (c) The sunspot activity period detection for the first 100 years (1818–1918), giving the largest silhouette width at a period of 11.2 years. (d) The sunspot activity period detection for the middle 100 years (1869–1969), giving the largest silhouette width at 10.8 years. (e) The sunspot activity period detection for the last 100 years (1919–2019), giving the largest silhouette width at 10.6 years. For (a–e), dots represent sampled period values. Dashed lines indicate the optimal period among those sampled.

silhouette and the fast algorithm described here further expand optimal solutions related to circular clustering. These solutions together open the door to applications in circular, periodic, angular, looped, and phase data analysis.

Although we can exactly and efficiently solve both linear [25] and circular [12] clustering problems, it is unclear if clustering points on a cylindrical surface is efficiently solvable by combing the linear and circular algorithms.

The original silhouette is undefined for data with only one cluster. This is also the case for circular silhouette. If one cluster is a valid option, some amendments will be necessary. One possible solution is the Bayesian information criterion, which promotes likelihood penalized by the number of clusters.

5 Conclusion

The presented circular silhouette and the fast algorithm provide a foundation to choose the optimal number of clusters automatically for circular or periodic data. It guarantees the optimality at a runtime linear to the input size on sorted data. We demonstrated its usage on simulated and real data. We found optimal clusters in circular genomes. Most interestingly, we found that the solar-cycle period is reducing in the last 200 years. As the methodology is unbiased, optimal, and fast, it has now enabled a scale-up of applications, such as circadian rhythm characterization, Internet traffic analysis by periods of day, week, or month, abnormal climate change detection, and pattern discovery in circular DNA and RNA molecules. We anticipate fast circular data analysis to play a wide range of roles in pattern discovery for science, engineering, and medicine.

Software and data availability

Circular silhouette algorithms are implemented in C++ and R in package 'CircularSilhouette' via https://cran.r-project.org/package=CircularSilhouette. Supplementary Materials contain code and data to reproduce result figures.

Acknowledgments

The reported work is partially funded by US National Science Foundation grant 1661331.

References

[1] L. Perlovsky, R. Ilin, R. Deming, R. Linnehan, and F. Lin, "Moving target detection and characterization with circular SAR," in *2010 IEEE Radar Conference*, 2010, pp. 661–666.

[2] A. Obermann, S. Bastin, S. Belamari, D. Conte, M. A. Gaertner, L. Li, and B. Ahrens, "Mistral and Tramontane wind speed and wind direction patterns in regional climate simulations," *Climate Dynamics*, vol. 51, no. 3, pp. 1059–1076, 2018.

[3] A. Choudhury and E. Urena, "Forecasting hourly emergency department arrival using time series analysis," *British Journal of Healthcare Management*, vol. 26, no. 1, pp. 34–43, 2020.
[4] K. Hornik, I. Feinerer, M. Kober, and C. Buchta, "Spherical k-means clustering," *Journal of Statistical Software*, vol. 50, no. 10, pp. 1 – 22, 2012.
[5] S.-J. Chang-Chien, M.-S. Yang, and W.-L. Hung, "Mean shift-based clustering for directional data," in *Third International Workshop on Advanced Computational Intelligence*, 2010, pp. 367–372.
[6] S.-J. Chang-Chien, W.-L. Hung, and M.-S. Yang, "On mean shift-based clustering for circular data," *Soft Computing*, vol. 6, no. 6, pp. 1043–1060, 2012.
[7] F. Lagona and M. Picone, "Model-based clustering of multivariate skew data with circular components and missing values," *Journal of Applied Statistics*, vol. 39, no. 5, pp. 927–945, 2012.
[8] ——, "A Gaussian–von Mises hidden Markov model for clustering multivariate linear-circular data," in *Statistical Models for Data Analysis*, P. Giudici, S. Ingrassia, and M. Vichi, Eds. Heidelberg: Springer International Publishing, 2013, pp. 171–179.
[9] C. Abraham, N. Molinari, and R. Servien, "Unsupervised clustering of multivariate circular data." *Stat Med*, vol. 32, no. 8, pp. 1376–1382, Apr 2013.
[10] A. Roy, S. K. Parui, and U. Roy, "SWGMM: a semi-wrapped Gaussian mixture model for clustering of circular–linear data," *Pattern Analysis and Applications*, vol. 19, no. 3, pp. 631–645, 2016.
[11] A. Roy, A. Pal, and U. Garain, "JCLMM: A finite mixture model for clustering of circular-linear data and its application to psoriatic plaque segmentation," *Pattern Recognition*, vol. 66, pp. 160–173, 2017.
[12] T. Debnath and M. Song, "Fast optimal circular clustering and applications on round genomes," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 6, pp. 2061–2071, 2021.
[13] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.
[14] F. Liu and Y. Deng, "Determine the number of unknown targets in open world based on elbow method," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 5, pp. 986–995, 2021.
[15] K. P. Burnham and D. R. Anderson, "Multimodel inference: Understanding AIC and BIC in model selection," *Sociological Methods & Research*, vol. 33, no. 2, pp. 261–304, 2004.
[16] C. A. Sugar and G. M. James, "Finding the number of clusters in a dataset," *Journal of the American Statistical Association*, vol. 98, no. 463, pp. 750–763, 2003.
[17] M. K. Bhowmik, T. Debnath, D. Bhattacharjee, and P. Dutta, "EF-index: Determining number of clusters (K) to estimate number of segments (S) in an image," *Image and Vision Computing*, vol. 88, pp. 29–40, 2019.
[18] A. K. Srivastava, S. W. McIntosh, N. Arge, D. Banerjee, M. Dikpati, B. N. Dwivedi, M. Guhathakurta, B. Karak, R. J. Leamon, S. K. Matthew, A. Munoz-Jaramillo, D. Nandy, A. Norton, L. Upton, S. Chatterjee, R. Mazumder, Y. K. Rao, and R. Yadav, "The extended solar cycle: Muddying the waters of solar/stellar dynamo modeling or providing crucial observational constraints?" *Frontiers in Astronomy and Space Sciences*, vol. 5, 2018.
[19] W. E. Lowell and G. E. Davis, "The light of life: Evidence that the sun modulates human lifespan," *Medical Hypotheses*, vol. 70, no. 3, pp. 501–507, 2008.
[20] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a 'kneedle' in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*. Minneapolis, MN, USA: IEEE, 2011, pp. 166–171.
[21] R. M. Andrews, I. Kubacka, P. F. Chinnery, R. N. Lightowlers, D. M. Turnbull, and N. Howell, "Reanalysis and revision of the Cambridge reference sequence for human mitochondrial DNA," *Nature Genetics*, vol. 23, no. 2, pp. 147–147, 1999.
[22] L. Katsir and O. Bahar, "Genome sequence of 'Candidatus Carsonella ruddii,' strain BT from the psyllid Bactericera trigonica," *Genome Announcements*, vol. 06, no. 4, pp. e01 466–17, 2018.
[23] Y. Wang, Y. Wang, C. Lang, D. Wei, P. Xu, and J. Xie, "Genome sequence of Lactobacillus curieae CCTCC M 2011381T, a novel producer of gamma-aminobutyric acid," *Microbiology Resource Announcements*, vol. 3, no. 3, pp. e00 552–15, 2015.
[24] SILSO World Data Center. (1818-2019) The international sunspot number. Royal Observatory of Belgium, avenue Circulaire 3, 1180 Brussels, Belgium. [Online]. Available: http://www.sidc.be/silso/
[25] M. Song and H. Zhong, "Efficient weighted univariate clustering maps outstanding dysregulated genomic zones in human cancers," *Bioinformatics*, vol. 36, no. 20, pp. 5027–5036, 2020.

Yinong Chen received the Bachelor of Engineering degree in agricultural engineering from Zhejiang University in 2022. She is pursuing an MS degree focusing on neuroengineering in the Department of Biomedical Engineering at Johns Hopkins University. Her research interests are brain-computer interface (BCI), electrode and sensor fabrication, and pattern recognition algorithms.

Tathagata Debnath received the Bachelor of Technology degree in computer science and engineering from the National Institute of Technology, Agartala, Tripura, India. He completed the Masters of Technology degree in computer science and engineering from Tripura University, a central university in India with the highest scores. He is pursuing a PhD degree from the Department of Computer Science at New Mexico State University (NMSU), USA. He has received a PhD tuition scholarship and Biopattern scholarship at NMSU. His research interests include genomics, proteomics, proteogenomics, bioinformatics, biological network analysis, computer vision, image processing, and machine learning including deep neural networks.

Andrew Cai is a former undergraduate student at Cornell University studying electrical and computer engineering as well as computer science, primarily interested in computer architecture. He is currently working in design verification at Apple Inc.

Mingzhou Song received the BS degree in electrical engineering from the Beijing University of Posts and Telecommunications, and the MS and PhD degrees from the Department of Electrical Engineering, University of Washington at Seattle. He was an assistant professor in the Department of Computer Science, Queens College of City University of New York. Later, he joined New Mexico State University, where he is a professor in the Department of Computer Science and a faculty member in the Graduate Program in Molecular Biology and Interdisciplinary Life Sciences. In 2019, he received a Fulbright scholar award and visited Charles University and Czech Technical University in Prague, Czech Republic. His research interests include statistical foundations for pattern discovery, data science algorithms for network inference, and applications to molecular biological systems. His lab has released eight standalone software packages, all open-source and freely available to the public; the two most popularly downloaded are 'Ckmeans.1d.dp' and 'FunChisq'.