Partitioned Local Depth (PaLD) Community Analyses in R

by Lucy D'Agostino McGowan, Katherine Moore, and Kenneth S. Berenhaut

Abstract Partitioned Local Depth (PaLD) is a framework for holistic consideration of community structure for distance-based data. This paper describes an R package, **pald**, for calculating Partitioned Local Depth (PaLD) probabilities, implementing community analyses, determining community clusters, and creating data visualizations to display community structure. We present essentials of the PaLD approach, describe how to use the **pald** package, walk through several examples, and discuss the method in relation to commonly used techniques.

0.1 Introduction

Partitioned Local Depth (PaLD) is a framework for holistic consideration of community structure for distance-based data. Leveraging a socially inspired perspective, the approach provides network-based community information which is founded on new measures of local depth and pairwise cohesion (partitioned local depth). The method does not require distributional assumptions, optimization criteria, nor extraneous inputs. A complete description of the perspective, together with a discussion of the underlying social motivation, theoretical results, and applications to additional data sets is provided in Berenhaut et al. (2022). A brief technical description is included below, directly following the introduction.

As suggested in Berenhaut et al. (2022), a main goal of PaLD is to "transform input dissimilarity comparisons into output pairwise relationship strengths (or cohesion) and resulting weighted networks". This is intended to provide within- and between-community structural information which goes beyond simple cluster labeling. Building on existing approaches to (global) depth, local depth expresses features of centrality via an interpretable probability which is free of parameters and robust to outliers. Partitioning the probabilities which define local depth, we then obtain a measure of cohesion between pairs of individuals. Both local depth and cohesion reflect aspects of relative position (rather than absolute distance) and provide a straightforward way to account for varying density across the space. As shown in Berenhaut et al. (2022), provided that two sets are separated (in the sense that the minimum between-set distance is greater than the maximum within-set distance), cohesion is invariant under the contraction and dilation of the distances within each set. This property may be particularly valuable when one has reason to believe that there is heterogeneity in density across the space.

As cohesion captures a sense of the relationship strength between points, we can then analyze and visualize the resulting community structure via a network whose edges are weighted by (mutual) cohesion. The underlying social framework motivates a straightforward yet elegant threshold for distinguishing between strongly and weakly cohesive pairs.

Networks obtained from cohesion can be displayed using a force-directed graph drawing algorithm; here we will graphically emphasize the strong ties (colored by connected component). We refer to the connected components of the network of strong ties as community "clusters". Note that to qualify as a cluster in this definition, one may not have any strong ties with those outside the cluster, and thus the existence of disjoint groups is a strong signal for separation. Here, clusters are identified without additional user inputs nor optimization criteria. If one wishes to further break the community graph into groups, one may consider using community detection methods (such as spectral clustering or the Louvain algorithm), as available, say, in the **igraph** package. The collection of strong ties may be used in place of (weighted) k-nearest neighbors in settings such as classification and smoothing. Overall, the structural information obtained from local depth, cohesion and community graphs can provide a holistic perspective to the data which does not require the use of distributional

assumptions, optimization criteria nor additional user inputs.

It is important to emphasize at the outset that community analyses go beyond simple cluster labeling to address intra- and inter-cluster structure, and can supplement results from other methods for clustering, embedding, data depth, nearest neighbors, etc. In addition, the method does not require distributional assumptions, optimization criteria, nor extraneous inputs. Theoretical considerations (see Berenhaut et al. (2022)) and examples point to distinctive properties that assist in considerations of complex data, in particular with respect to varying density and high dimensions. See the section Cultural and Psychological distance analysis, below, for some discussion in the context of density-based methods.

After a brief introduction to the partitioned local depth approach (including a concrete and instructive example), below, we present a new package, **pald**, for calculating partitioned local depths, implementing community analyses, and creating data visualizations to display community structure. This paper describes how to use the package, walks through several examples, and contrasts the method results with commonly used techniques. Together, these demonstrate both the novelty of the method and utility of the implementation in the package described.

0.2 The partitioned local depth approach

The PaLD methodology as introduced in Berenhaut et al. (2022) offers a parameter-free approach to analyzing community structure in distance-based data. First, consider a ground set $\mathcal{S}$ equipped with a meaningful measure of pairwise distance (or dissimilarity), $d : \mathcal{S} \times S \to \mathbb{R} \cup \infty$. We require only that d satisfies the requirements that for all $x, y \in S$ with $y \neq x$,

$$d(x,x) \leq d(y,x) \quad \text{and} \quad d(x,x) < d(x,y). \tag{1}$$

Note that d need not necessarily be symmetric nor positive (see (i) below).

We begin with a concept formalizing locality to a pair. For any pair $(x, y) \in \mathcal{S} \times \mathcal{S}$, the *local focus*, $U_{x,y}$ (illustrated in Figure 1 for two-dimensional Euclidean data) is defined via

$$U_{x,y} \stackrel{\text{def}}{=} \{z \in S \mid d(z,x) \leq d(y,x) \text{ or } d(z,y) \leq d(x,y)\}. \tag{2}$$

The set $U_{x,y}$ is comprised of elements that are "locally" relevant to the pair (x, y). Eq. (1) guarantees that both x and y are elements of $U_{x,y}$. For discussion from a social perspective, wherein the data are embedded in a latent social space, see Social Framework in Berenhaut et al. (2022). Recall that in (2), the distance d need not be symmetric; leveraging the social perspective, we are interested in the direct closeness of z *to* x and y, not the reverse (see also (iv), below).

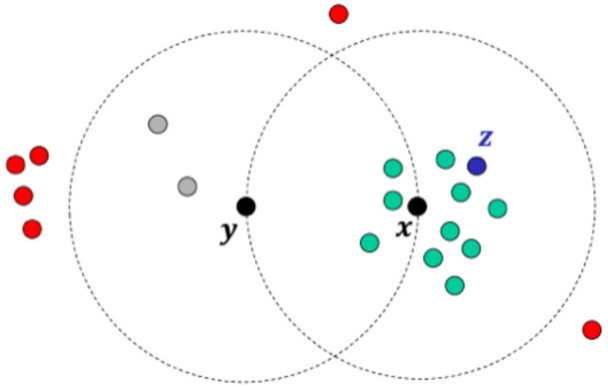

Figure 1: The local focus for two points, x and y, where $\mathcal{S}$ is a subset of $\mathbb{R}^2$, under Euclidean distance (reproduced with permission from Berenhaut et al. (2024)). The points in red are outside the focus, $U_{x,y}$. Those in green (and z in blue) are in the focus and closer to x, while those in gray are closer to y.

Now, suppose that x is fixed, and select Y and Z uniformly at random from $\mathcal{S} \setminus \{x\}$ and $U_{x,Y}$, respectively. The *local depth* of x, $\ell_{\mathcal{S}}(x)$, is defined as the probability that Z is closer to x than it is to Y (with a coin flip in the case Z is equidistant), i.e.

$$\ell_{\mathcal{S}}(x) \stackrel{\text{def}}{=} P(d(Z,x) < d(Z,Y)). \tag{3}$$

The local depth of x can be thought of as a measure of the extent to which x is (relatively) surrounded by other elements of $\mathcal{S}$ (under d).

Finally, the *cohesion*, $C_{x,w}$, of w to x, for a given $w \in \mathcal{S}$, is obtained through a partitioning of the probability defining local depth in (3). In particular,

$$C_{x,w} \stackrel{\text{def}}{=} P(Z = w \text{ and } d(Z,x) < d(Z,Y)). \tag{4}$$

The cohesion of w to x can be viewed as the contribution of w to the local depth of x. Eq. (1) guarantees that when $Z = x$, $d(Z,x) < d(Z,Y)$. For some theoretical properties of cohesion, see Berenhaut et al. (2022). Note that the sum of all cohesions is conserved at $n/2$, where $n \stackrel{\text{def}}{=} |\mathcal{S}|$; we thus have a constant mean local depth of 0.5 (independent of n).

A universal threshold for strong cohesion (see Berenhaut et al. (2022)) is defined via

$$T \stackrel{\text{def}}{=} P(Z = W \text{ and } d(Z,X) < d(Z,Y)), \tag{5}$$

where X, Y, Z, and W are selected uniformly at random from $\mathcal{S}$, $\mathcal{S} \setminus X$, $U_{X,Y}$, and $U_{X,Y}$, respectively. The threshold, T, can be conveniently calculated as one half the average of the diagonal of the matrix of pairwise cohesion values (see Berenhaut et al. (2022) for details), i.e.

$$T = \frac{1}{2}\left(\frac{1}{n}\sum_{x \in \mathcal{S}} C_{x,x}\right). \tag{6}$$

A strong relationship connection between x and w is then established when:

$$\min\{C_{x,w}, C_{w,x}\} \geq T. \tag{7}$$

For further details on PaLD, including theoretical results, discussion of the underlying social perspective and applications, see Berenhaut et al. (2022) (see also Berenhaut and Moore (2022)).

Before moving on to a simple concrete example, we provide a few remarks, for the

interested reader, on the concepts introduced above.

(i) (Distances and dissimilarities) For the purposes of the package, we require only the inequalities in (1). This implies the convenient form of the threshold in (7), see Berenhaut et al. (2022), and is crucial to the underlying social perspective of conflict. For a more general mathematical PaLD framework, wherein the concepts of locality and support are further refined see Berenhaut et al. (2024). As the definitions in (2), (3) and (4) only depend on triplet distance comparisons, local depth and cohesion are invariant under monotone transformations of distance, and negative distances are fine. If meaningful in context, distances need not be symmetric and self-distances, $d(x, x)$, may vary over x (subject to the constraint in (1)). The triangle inequality may be violated to some extent, mirroring ideas of strong triadic closure (Granovetter (1973)).

(ii) (Local foci) The concept of local foci in (2) is crucial to all that follows. It allows for consideration of the data at varying scales without the need for localizing parameters. The selection of Z in (3) and (4) reflects the larger presence of individuals in smaller foci (see Feld (1981) for discussion is social settings). The results in Berenhaut et al. (2022) regarding limiting irrelevance of density and separation under increasing concentration are driven by local comparisons. Self-cohesions (i.e., diagonal elements of C) are not necessarily equal. This is driven by the fact that by (4),

$$C_{x,x} = \frac{1}{n-1} \sum_{y \neq x} \frac{1}{|U_{x,y}|}, \tag{8}$$

and $C_{x,x}$ is dependent on local density around x.

(iii) (Non-monotonicity of the size of local foci) As distance away from an element, x, increases, monotonicity in volume of local foci is not necessary. In particular, consider the one-dimensional set $\mathcal{S} = \{B, x, A, C, D\}$, with $B = 5$, $x = 11$, $A = 16$, $C = 18$, $D = 19$. Under Euclidean distance, $d(x, A) = 5 < 6 = d(x, B)$ but $|U_{x,A}| = |\{x, A, C, D\}| = 4$ and $|U_{x,B}| = |\{x, B, A\}| = 3$. This further emphasizes the capturing of local density, an important feature of PaLD.

(iv) (Assymmetry) Note that the definitions in (2), (3) and (4) are stated to allow for asymmetric distances, as may occur for instance when considering distances on graphs. Note that even symmetric distances can easily lead to asymmetric cohesion. This is the case even for the simple one-dimensional data considered in Example 1. In considering strong relationships based on cohesion, though, symmetrization is employed in (7).

(v) (The use of weighted networks) In what follows, we will at times have occasion to consider a network with node set $\mathcal{S}$ and pairwise edges weighted via the respective values of cohesion. When viewed as a weighted (connectivity) network in this sense,

   (a) the weighted nodal degrees are the values of local depth, addressing considerations of (local) data depth,

   (b) the network may be embedded in low-dimensional Euclidean space via standard network embedding techniques, providing visualization of the data that adapts to relative density,

   (c) the isolated nodes of either the original network or that restricted to strong ties can provide information regarding outliers,

   (d) the connected components of the network restricted to strong ties (i.e. those with weights above the threshold in (6)) can be taken as “clusters" in the classical sense,

   (e) weighted or unweighted network neighbors may be used in place of k-nearest neighbors in settings such as classification and smoothing, and

   (f) some network analyses can be informative regarding data structure.

Ideas of distance on networks can be complex, though, and this is particularly so for general weighted networks (e.g. correlation or social networks). As is the case here, edge weights may reflect intensity of connection as opposed to simple pairwise distance, and it is important to exercise caution when applying naïve functions intended for unweighted networks (or simple distance-based weighted networks) in **igraph** and other software.

(vi) (Matrix methods) For convenience, we use the **igraph** package to analyze weighted networks, where appropriate. Matrix-based methods are also possible. For instance, we employ `igraph::components` to obtain community clusters from networks, but alternatively one could use blocks extraction directly from the cohesion matrix using packages such as **lintools**.

(vii) (Density-based methods and localized approaches) Complimentary density-based methods such as DBSCAN (density-based spatial clustering of applications with noise) and its hierarchical variant HDBSCAN (see for instance Campello et al. (2020)), seek to identify high-density regions as clusters. We discuss these in the context of a concrete example below (see Cultural and Psychological distance analysis). Determination of required parameter values can be challenging. There are also extent methods for considering data depth which probe local structure (see Paindaveine and Van Bever (2013) and Agostinelli and Romanazzi (2011)). These often also require localizing parameters. In Berenhaut et al. (2022) (see Theorem 2: Limiting Irrelevance of Density), it is shown that local depth and cohesion account for varying density in the sense that, provided subsets are sufficiently separated, the cohesion is maintained, as within-subset distances are contracted or dilated, without the need to search over a parameter space.

(viii) (Social latent spaces) The definitions in (2), (3), (4), and (6), are developed from a social perspective. The interested reader can refer to the section *Social Framework* in Berenhaut et al. (2022). For further discussion see Berenhaut et al. (2024) and Berenhaut and Moore (2022).

Example 1. As a concrete example demonstrating the PaLD framework, consider the one-dimensional set $\mathcal{S} \overset{\text{def}}{=} \{1, 3, 7, 8, 9, 13, 17\}$, with pairwise (Euclidean) distances given in the array below.

```
    1  3  7 8 9 13 17
1   0  2  6 7 8 12 16
3   2  0  4 5 6 10 14
7   6  4  0 1 2  6 10
8   7  5  1 0 1  5  9
9   8  6  2 1 0  4  8
13 12 10  6 5 4  0  4
17 16 14 10 9 8  4  0
```

The local depths (rounded to three decimal places) are

```
    1     3     7     8     9    13    17
0.409 0.463 0.588 0.675 0.600 0.471 0.294
```

while the pairwise cohesions (in matrix form) are given by

```
       1     3     7     8     9    13    17
1  0.210 0.127 0.036 0.024 0.012 0.000 0.000
3  0.137 0.220 0.048 0.036 0.024 0.000 0.000
7  0.048 0.048 0.220 0.137 0.109 0.028 0.000
8  0.024 0.052 0.163 0.218 0.163 0.056 0.000
```

```
9  0.024 0.024 0.109 0.137 0.220 0.063 0.024
13 0.000 0.000 0.012 0.036 0.103 0.188 0.133
17 0.000 0.000 0.000 0.000 0.012 0.107 0.175
```

where (non-diagonal) cohesion values above the threshold of

$$0.1036 = (1/14)(0.210 + 0.220 + 0.220 + 0.218 + 0.220 + 0.188 + 0.175) \tag{9}$$

are indicated in blue; see (7)). The resulting weighted network of pairwise cohesion values is displayed in Figure 2; here strong ties (with cohesion values above the threshold) are colored according to community clusters (the connected components of the network of strong ties), i.e. $\{1,3\}$, $\{7,8,9\}$ and $\{13,17\}$.

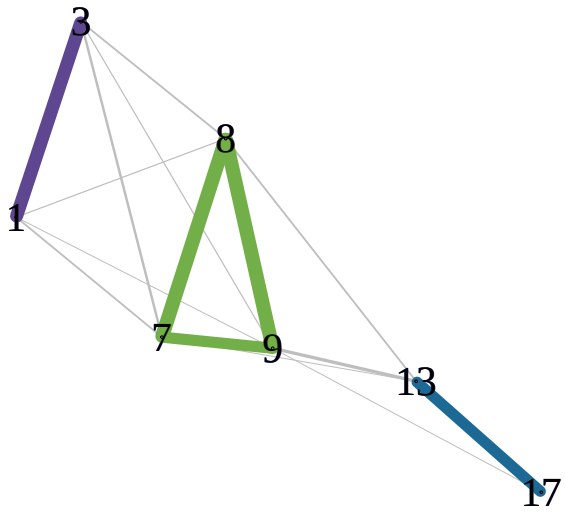

Figure 2: The community cluster network for the simple one-dimensional data considered in Example 1. Strong ties (with cohesion values above the threshold) are colored according to community clusters (the connected components of the network of strong ties), while weak ties are indicated in grey.

Note that the cohesion of the element 17 to the element 1 is zero. This reflects the fact that when an element, Y, other than 1 is selected uniformly at random from $\mathcal{S}$, and an element Z is selected uniformly at random from $U_{x,Y}$, 17 is never, at once, equal to Z and closer to 1 than it is to the opposing element Y. In fact, 17 is not an element of $U_{1,y}$, for $y \in \{3,7,8\}$. For $y \in \{9,13,17\}$, $17 \in U_{1,y}$, but in each case, $d(17,y) < d(17,1)$. This zero cohesion is reflected in the absence of any edge (grey or colored) from the node for 17 to the node for 1, in Figure 2. □

We now turn to discussion of the **pald** package.

0.3 pald

The main functions in the **pald** package can be split into 3 categories:

1. A function for computing the matrix of cohesion values, as defined in (4).
2. Functions for extracting useful information from the cohesion matrix, such as local depths, neighbors, community clusters, and graph objects.
3. Plotting functions for community graphs.

In addition, the package provides a number of pertinent example data sets which may be used to demonstrate community analysis, including a synthetic data set of two-dimensional

points created by Gionis et al. to demonstrate aggregation, clustering data generated from the scikit-learn Python package (Pedregosa et al., 2011), data describing cognate relationships between words across 87 Indo-European languages (Dyen et al., 1992), data compiled by Love and Irizarry (2015) of tissue gene expressions, data employing information from the World Values Survey (Inglehart et al., 2014) on cultural values regarding family, religion, education, and institutions for several regions (Muthukrishna et al., 2020), and three example data sets generated for the Berenhaut et al. (2022) paper.

While it is not a necessity, the **pald** package is designed to function well with the pipe operator, `|>`, particularly for those who are familiar with tidyverse-based approaches. This functionality will be demonstrated briefly below.

Creating the cohesion matrix

For the purposes of the **pald** package, the sole input for the Partitioned Local Depth (PaLD) computations is a distance matrix or `dist` object. Recall that the collection of input distances (or dissimilarities) is assumed to satisfy the requirements in (1). More generally, the method only requires triplet distance comparisons, as opposed to exact numeric distances (see Berenhaut et al. (2022)).

For demonstration purposes, we first show how one can compute a distance matrix from an input data frame with, say, two variables `x1` and `x2`. The input data may be of any dimension; in fact the PaLD framework provides advantages when considering high-dimensional data (see the **Examples** section as well as Berenhaut et al. (2022)).

```
library(pald)
df <- data.frame(
  x1 = c(6, 8, 8, 16, 4, 14),
  x2 = c(5, 4, 10, 8, 4, 10)
)
rownames(df) <- c("A", "B", "C", "D", "E", "F")
```

The `dist` function returns a (default Euclidean) pairwise distance matrix for an input data frame, as demonstrated below. If the data are already provided as a distance matrix (or `dist` object), the user can skip to the next step. Note that the distance matrix needed for the subsequent functions does not need to be a `dist` object and *need not* be symmetric.

```
d <- dist(df)
```

The function above creates a `dist` object. If converted to a matrix, this will be an $n \times n$ distance matrix, where n corresponds to the number of observations in the original data frame (in this example $n = 6$).

This `dist` object, or a distance matrix, can then be passed to the `cohesion_matrix` function in order to calculate pairwise cohesion values.

```
cohesion_matrix(d)
```

```
#>            A          B          C         D          E         F
#> A 0.25000000 0.18333333 0.06666667 0.0000000 0.18333333 0.0000000
#> B 0.14000000 0.24000000 0.05000000 0.0000000 0.10666667 0.0000000
#> C 0.07333333 0.07333333 0.20333333 0.0000000 0.03333333 0.0800000
#> D 0.00000000 0.00000000 0.00000000 0.2333333 0.00000000 0.1333333
#> E 0.14000000 0.10666667 0.03333333 0.0000000 0.24000000 0.0000000
#> F 0.00000000 0.00000000 0.05000000 0.1400000 0.00000000 0.2400000
#> attr(,"class")
#> [1] "cohesion_matrix" "matrix"          "array"
```

Equivalently, the user can use the native pipe `|>` as follows.

```
df |>
  dist() |>
  cohesion_matrix()
```

```
#>            A          B          C         D          E         F
#> A 0.25000000 0.18333333 0.06666667 0.0000000 0.18333333 0.0000000
#> B 0.14000000 0.24000000 0.05000000 0.0000000 0.10666667 0.0000000
#> C 0.07333333 0.07333333 0.20333333 0.0000000 0.03333333 0.0800000
#> D 0.00000000 0.00000000 0.00000000 0.2333333 0.00000000 0.1333333
#> E 0.14000000 0.10666667 0.03333333 0.0000000 0.24000000 0.0000000
#> F 0.00000000 0.00000000 0.05000000 0.1400000 0.00000000 0.2400000
#> attr(,"class")
#> [1] "cohesion_matrix" "matrix"          "array"
```

The *cohesion matrix* output by the `cohesion_matrix` function is the main input for the majority of the remaining functions.

Functions for extracting information from the cohesion matrix

From the *cohesion matrix*, a variety of useful quantities can be computed. Below, we create a cohesion matrix using the functions described in the previous section.

```
df |>
  dist() |>
  cohesion_matrix() -> cohesion
```

The `local_depths` function calculates the *depth* of each point, as defined in (3), outputting a vector of local depth probabilities.

```
local_depths(cohesion)
```

```
#>         A         B         C         D         E         F
#> 0.6833333 0.5366667 0.4633333 0.3666667 0.5200000 0.4300000
```

In this case, the (locally) deepest point is `A`.

The `strong_threshold` function will calculate the cohesion threshold for strong ties given in (5), which reflects typical cohesion for local points (see Berenhaut et al. (2022)). Computationally, this is equal to half the average of the diagonal of the cohesion matrix, and is a threshold that may be used to distinguish between strong and weak ties.

```
strong_threshold(cohesion)
```

```
#> [1] 0.1172222
```

Here, the threshold is a little above `0.117`.

The function `cohesion_strong` will update the cohesion matrix to set all weak ties to zero (via the `strong_threshold` function). Reflecting (7), by default, the matrix will also be symmetrized, using the entry-wise (parallel) minimum of the cohesion matrix and its transpose.

```
cohesion_strong(cohesion)
```

```
#>      A    B         C         D    E         F
#> A 0.25 0.14 0.0000000 0.0000000 0.14 0.0000000
#> B 0.14 0.24 0.0000000 0.0000000 0.00 0.0000000
#> C 0.00 0.00 0.2033333 0.0000000 0.00 0.0000000
```

```
#> D 0.00 0.00 0.0000000 0.2333333 0.00 0.1333333
#> E 0.14 0.00 0.0000000 0.0000000 0.24 0.0000000
#> F 0.00 0.00 0.0000000 0.1333333 0.00 0.2400000
#> attr(,"class")
#> [1] "cohesion_matrix" "matrix"          "array"
```

The `community_graphs` function takes the cohesion matrix and creates **igraph** objects, graphs that describe the (symmetrized) relationship structure between points. This function will output a list of three objects:

- `G`: the weighted (community) graph whose edge weights are mutual cohesion
- `G_strong`: the weighted (community) graph consisting of edges for which mutual (symmetrized) cohesion (i.e. the minimum of the two directed cohesion values for any given pair) is greater than the threshold for strong ties
- `layout`: the graph layout. By default this is provided by the Fruchterman Reingold (FR) force-directed graph drawing algorithm for the graph `G`, as implemented in the **igraph** package.

```
graphs <- community_graphs(cohesion)
graphs[["G_strong"]]
```

```
#> IGRAPH c4d81a7 UNW- 6 3 --
#> + attr: name (v/c), weight (e/n)
#> + edges from c4d81a7 (vertex names):
#> [1] A--B A--E D--F
```

Here we see that there are three connected components, ties `A-B` and `A-E` form the first community cluster, and the tie `D-F` which forms another.

The `any_isolated()` function will check whether there are any isolated points (according to cohesion).

```
any_isolated(cohesion)
```

Here, there are no isolated points, i.e. points having zero cohesion with all other points in the data (an extreme form of outlier).

The "community clusters" identified by PaLD are the connected components of the graph of strong ties, `G_strong`. To directly calculate these, we can use the `community_clusters` function. This will output a data frame with two columns; the first corresponds to the individual (`point`), as identified by the row name of the original input data frame, `df`, the second identifies the `community` that the individual belongs to.

```
community_clusters(cohesion)
```

```
#>   point community
#> A     A         1
#> B     B         1
#> C     C         2
#> D     D         3
#> E     E         1
#> F     F         3
```

In this example, three communities are identified with these six points. Points `A`, `B`, and `E` fall into Community 1. Point `C` is in Community 2 (a community of size 1) and points `D` and `F` fall into Community 3.

0.4 Plotting functions

The final category of function is that for data visualization. We can begin by visualizing the points in the data frame `df` (Figure 3). When visualizing these points, it is important to have the aspect ratio of the x and y axes equal to 1 so as to not distort distances. When using the **ggplot2** package for this visualization, one can use the command `coord_fixed(ratio = 1)`. If using the `plot` function included in the base library, one can use the `asp = 1` argument.

```
library(ggplot2)
ggplot(df, aes(x1, x2)) +
  geom_text(label = rownames(df)) +
  coord_fixed(ratio = 1) +
  xlim(c(4, 16)) +
  ylim(c(4, 16))
```

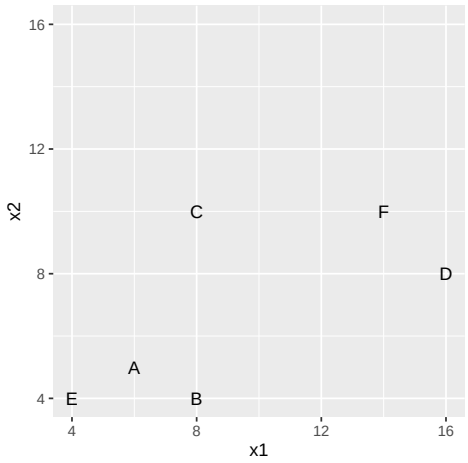

Figure 3: Visualization of the points from the data frame `df`

We can pass the cohesion matrix to the `plot_community_graphs` function to view the relationship between points (Figure 4). The function will also permit parameters that can be passed to `plot.igraph` via the `...` argument.

```
plot_community_graphs(cohesion,
                      vertex.label.cex = 2,
                      vertex.label.dist = 0.9)
```

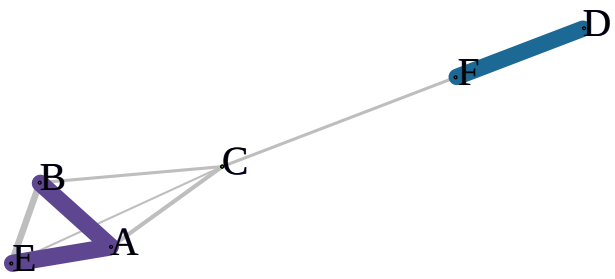

Figure 4: PaLD graph displaying the relationship between the points in data frame `df`

Notice in this plot the force-directed layout does not match that of the original data frame as seen in Figure 3. Since our original data is two-dimensional, it may be reasonable

to use the latter as the layout for plotting. Figure 5 includes this update as well as others addressing some of the aesthetics, such as employing more readable labels. The `layout` argument allows the user to pass a matrix to dictate the 2-dimensional layout of the graph. For example, if we wanted the graph to match the visualization displayed in Figure 3, we could pass `as.matrix(df)` (a matrix of the data frame `df`) to the `layout` argument (Figure 5). Here, we increase the vertex size and change the vertex label color, through the argument specifications `vertex.size = 100` and `vertex.label.color = "white"`. Additionally, to allow axes, we use `axes = TRUE`, and to put these back on the original scale we set `rescale = FALSE`, resetting the axis limits using `xlim` and `ylim`. The `par(pty = "s")` function forces the subsequent plot to be square.

```
par(pty = "s")

plot_community_graphs(cohesion,
                      layout = as.matrix(df),
                      vertex.size = 100,
                      vertex.label.color = "white",
                      axes = TRUE,
                      rescale = FALSE,
                      asp = 1,
                      xlim = c(4, 16),
                      ylim = c(4, 16))
```

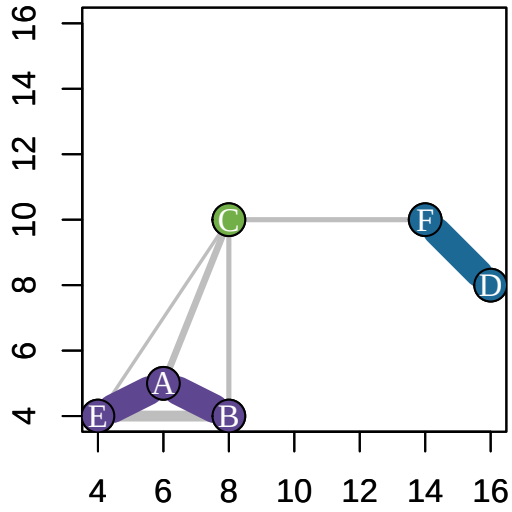

Figure 5: PaLD graph displaying the relationship between the points in data frame `df`, matching the original layout in Figure 3

0.5 Examples

We will demonstrate the utility of the **pald** package through several illustrative examples.

Community analysis for tissue gene expression data

The first example is from a subset of data from Zilliox and Irizarry (2007), McCall et al. (2011), and McCall et al. (2014), obtained from the **tissuesGeneExpression** bioconductor package (Love and Irizarry, 2015) consisting of 22,215-dimensional gene expression data from 189 tissue samples. A (Euclidean) `dist` object was created using this data set and is included in the **pald** package in an object called `tissue_dist`.

The `tissue_dist` object is a `dist` object resulting in a distance matrix with 189 rows and 189 columns.

We can create the cohesion matrix using the `cohesion_matrix` function.

```
tissue_cohesion <- cohesion_matrix(tissue_dist)
```

We can display relationships between tissue samples, both locally and globally through the `plot_community_graphs` function (Figure 6). For clarity of the display, we show how to remove the labels using `show_labels = FALSE`. We will instead color according to the labels by passing these to the `vertex.color` argument for the `plot.igraph` function (via the `...` argument). Similarly, we can add a legend using the `legend()` function, as you would for an **igraph** visualization. Additionally, we use the `edge_width_factor` and `emph_strong` arguments to adjust the width of the lines between and within PaLD communities.

```
labels <- rownames(tissue_cohesion)
plot_community_graphs(tissue_cohesion,
                      show_labels = FALSE,
                      vertex.size = 4,
                      vertex.color = as.factor(labels),
                      edge_width_factor = 35,
                      emph_strong = 5)
legend("topleft",
       legend = unique(as.factor(labels)),
       pt.bg = unique(as.factor(labels)),
       col = "black",
       pch = 21)
```

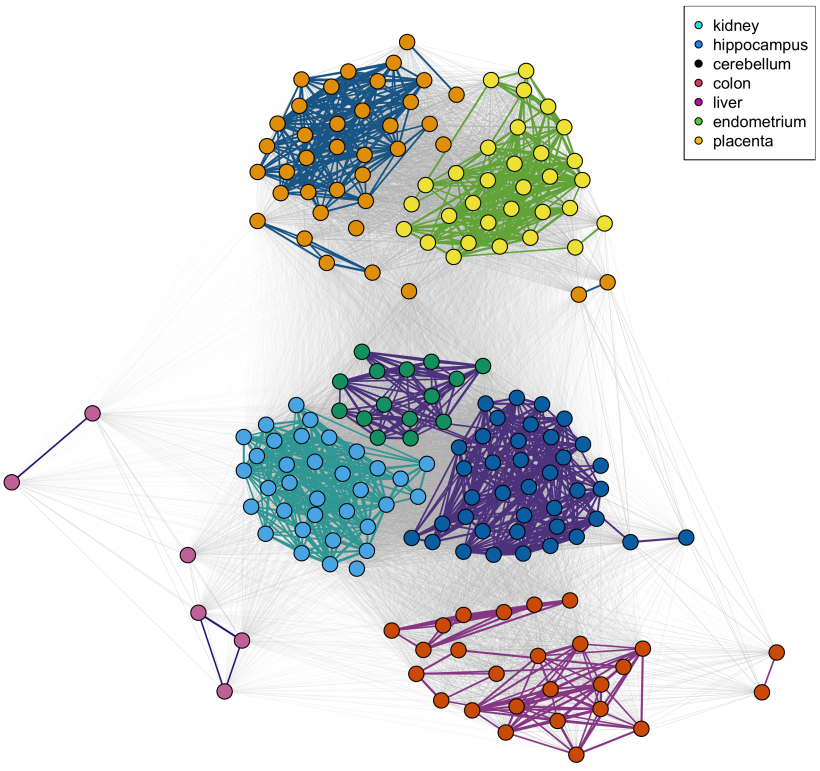

Figure 6: Community cluster network for the tissue data. The line colors indicate the PaLD communities; the point colors indicate the tissue classification.

A summary of community strong ties can be produced via the following code. Note that `tissue_graph_strong` is an **igraph** object corresponding to the graph displayed in Figure 6, restricted only to strong ties.

```
tissue_graphs <- community_graphs(tissue_cohesion)
tissue_graph_strong <- tissue_graphs[["G_strong"]]
E <- igraph::get.edgelist(tissue_graph_strong)
tissue_strong_ties <- data.frame(
  strong_ties = apply(E, 1, paste, collapse = ",")
)
tissue_strong_ties |>
  table() |>
  sort(decreasing = TRUE)
```

```
#> strong_ties
#>           kidney,kidney
#>                     273
#>             colon,colon
#>                     241
#> hippocampus,hippocampus
#>                     191
#>   cerebellum,cerebellum
#>                     186
#>             liver,liver
#>                      85
#> endometrium,endometrium
#>                      81
#>       placenta,placenta
#>                       4
#>      kidney,endometrium
#>                       3
```

Note that there are only three strong ties between different tissue types (kidney and endometrium) in the community cluster network of 1,064 strong ties total.

0.6 Cognate-based Language Families

This example explores a data set from Dyen et al. (1992) that summarizes relationships between 87 Indo-European languages from the perspective of cognates, coded using 2,655-dimensional binary vectors. A `dist` object was created from this data set and is included in the **pald** package in an object called `cognate_dist`.

Here we will demonstrate how one can further apply functions in the **igraph** package to objects output from the **pald** package. We can first use the `cohesion_matrix` function to calculate the cohesion matrix and the `community_graphs` function to create a list with the weighted community graph, the weighted community graph with only strong ties included, and the layout. From this, we can extract the graph with only the strong ties, here called `cognate_graph_strong`.

```
cognate_cohesion <- cohesion_matrix(cognate_dist)
cognate_graphs <- community_graphs(cognate_cohesion)

cognate_graph_strong <- cognate_graphs[["G_strong"]]
```

We can then use the `neighbors` function from the **igraph** package to extract the strong neighbors in this graph. For example, we can extract all neighbors for the language "French", via the following code.

```
french_neighbors <- igraph::neighbors(cognate_graph_strong, "French")
french_neighbors
```

```
#> + 8/87 vertices, named, from 5c6bb3a:
#> [1] Italian
#> [2] Ladin
#> [3] Provencal
#> [4] Walloon
#> [5] French_Creole_C
#> [6] French_Creole_D
#> [7] Spanish
#> [8] Catalan
```

Similarly, we can sort and print the associated neighborhood weights by subsetting the cohesion matrix.

```
cognate_cohesion["French", french_neighbors] |>
  sort(decreasing = TRUE)
```

```
#>         Walloon
#>      0.03258771
#>       Provencal
#>      0.02871174
#> French_Creole_C
#>      0.02406057
#> French_Creole_D
#>      0.02406057
#>           Ladin
#>      0.02094596
#>         Italian
#>      0.01997696
#>         Catalan
#>      0.01859688
#>         Spanish
#>      0.01679733
```

We can again use the `plot_community_graphs` function to visualize the community clusters (Figure 7). One may note the commonly identifiable language clusters and that, under a slight rotation, some of the underlying geography is mirrored in the plot.

```
plot_community_graphs(
  cognate_cohesion,
  edge_width_factor = 30,
  emph_strong = 3,
  vertex.size = 3,
  vertex.label.cex = 0.7,
  vertex.label.dist = 1
)
```

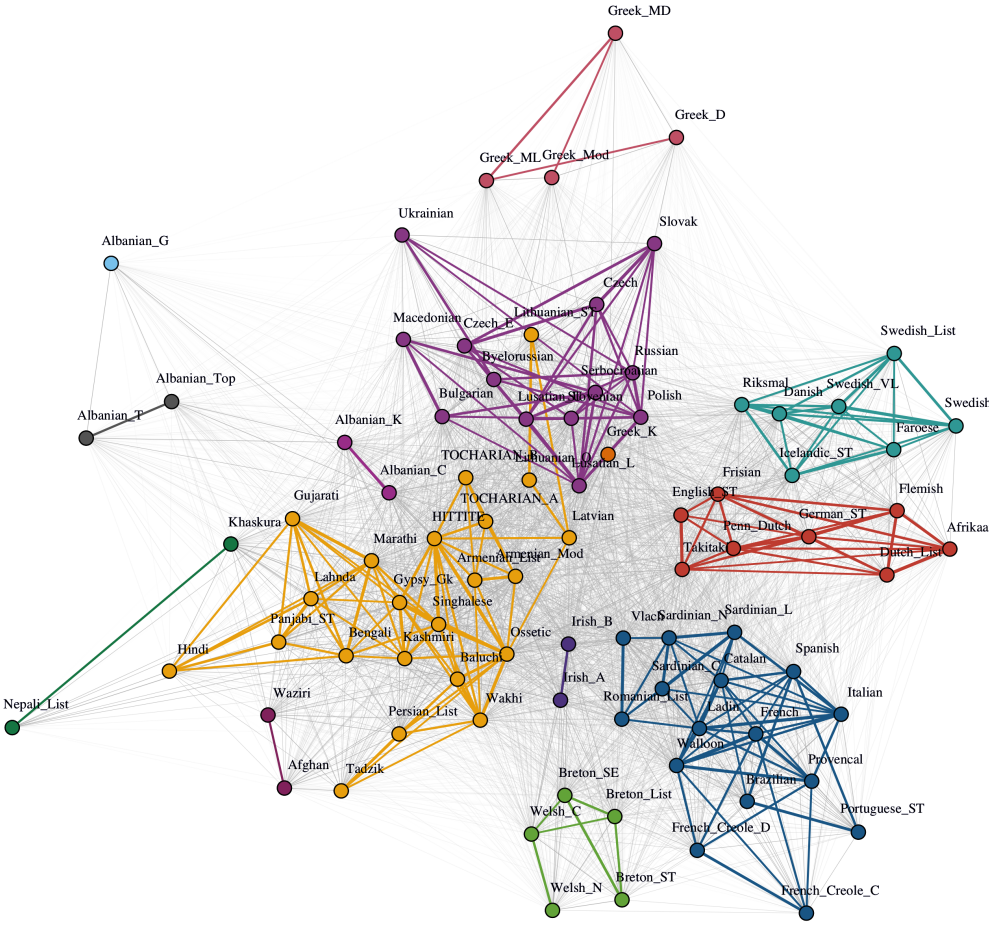

Figure 7: Community structure for 87 Indo-European languages, which employs cognate information that was coded via 2,665-dimensional binary vectors. Commonly identifiable language clusters arise along with informative inter- and intra-cluster structure. Several ancient languages are centrally located.

Community analysis for generated data

The **pald** package includes three randomly generated data frames corresponding to plots from Berenhaut et al. (2022):

- `exdata1` is a data set consisting of 8 points used to create Figure 1 in Berenhaut et al. (2022)
- `exdata2` is a data set consisting of 16 points used to create Figure 2 in Berenhaut et al. (2022)
- `exdata3` is a data set consisting of 240 points used to create Figure 4D in Berenhaut et al. (2022)

Here, we will demonstrate how to use `exdata3`. These points were generated from bivariate normal distributions with varying means and variances. There are eight "true" communities.

We can contrast resulting community clusters obtained via PaLD (i.e. connected components of the network of strong ties) with clusters as obtained through common cluster analysis techniques. Here we will consider two common clustering methods. The code below calculates the cohesion matrix (`exdata_cohesion`) as well as the community clusters obtained via PaLD (`exdata_pald`), along with $k = 8$ clusters obtained via k-means

(exdata_kmeans) and hierarchical clustering using complete linkage (exdata_hclust). Recall that there is no need to determine values for extraneous inputs in the case of PaLD (e.g. the number of clusters, as is necessary to specify for k-means and hierarchical clustering).

```
exdata_cohesion <- exdata3 |>
  dist() |>
  cohesion_matrix()

exdata_pald <- community_clusters(exdata_cohesion)$community

exdata_kmeans <- kmeans(exdata3, 8)$cluster

exdata_hclust <- exdata3 |>
  dist() |>
  hclust() |>
  cutree(k = 8)
```

The information is displayed in Figure 8.

```
par(mfrow = c(1, 3), pty = "s")
plot(
  exdata3,
  pch = 16,
  col = pald_colors[exdata_pald],
  xlab = "",
  ylab = "",
  main = "PaLD Communities",
  asp = 1
)
plot(
  exdata3,
  pch = 16,
  col = pald_colors[exdata_kmeans],
  xlab = "",
  ylab = "",
  main = "K-Means Clusters (k = 8)",
  asp = 1
)
plot(
  exdata3,
  pch = 16,
  col = pald_colors[exdata_hclust],
  xlab = "",
  ylab = "",
  main = "Hiearchical Clusters (k = 8)",
  asp = 1
)
```

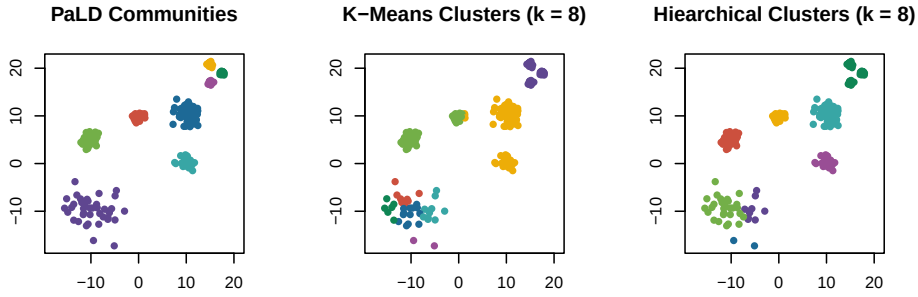

Figure 8: PaLD, k-means, and hierarchical 8-clustering of randomly generated example data (from Berenhaut et al. (2022); Figure 4D).

Cohesion is particularly useful when considering data with varying local density; see Berenhaut et al. (2022) for further examples, discussion, and theoretical results. Note that the PaLD algorithm is able to detect the eight natural groups within the data (along with inter- and intra-community structure not displayed here) without the use of any additional inputs (e.g., number of clusters) nor optimization criteria. Despite the user input of the "correct" number of clusters (i.e., $k = 8$) both k-means and hierarchical clustering do not provide the desired result.

Cultural and Psychological distance analysis

In this example we perform a PaLD analysis for cultural distances obtained in Muthukrishna et al. (2020) from two recent waves of the World Values Survey (2005 to 2009 and 2010 to 2014; see Inglehart et al. (2014)). Distances are computed using the cultural fixation index (CFST), which is a measure built on the framework of fixation indices from population biology (Bell et al. (2009); Cavalli-Sforza et al. (1994)). Recall that the foundation of PaLD in within-triplet comparisons allows for the employment of application-dependent and non-Euclidean measures of dissimilarity. The `dist` object is included in the **pald** package (`cultures`). We will first create the cohesion matrix using the `dist` object `cultures`, and proceed to plot the community graph.

```
cultures_cohesion <- cohesion_matrix(cultures)

plot_community_graphs(
  cultures_cohesion,
  edge_width_factor = 30,
  emph_strong = 3,
  vertex.label.cex = 0.7,
  vertex.size = 3,
  vertex.label.dist = 1
)
```

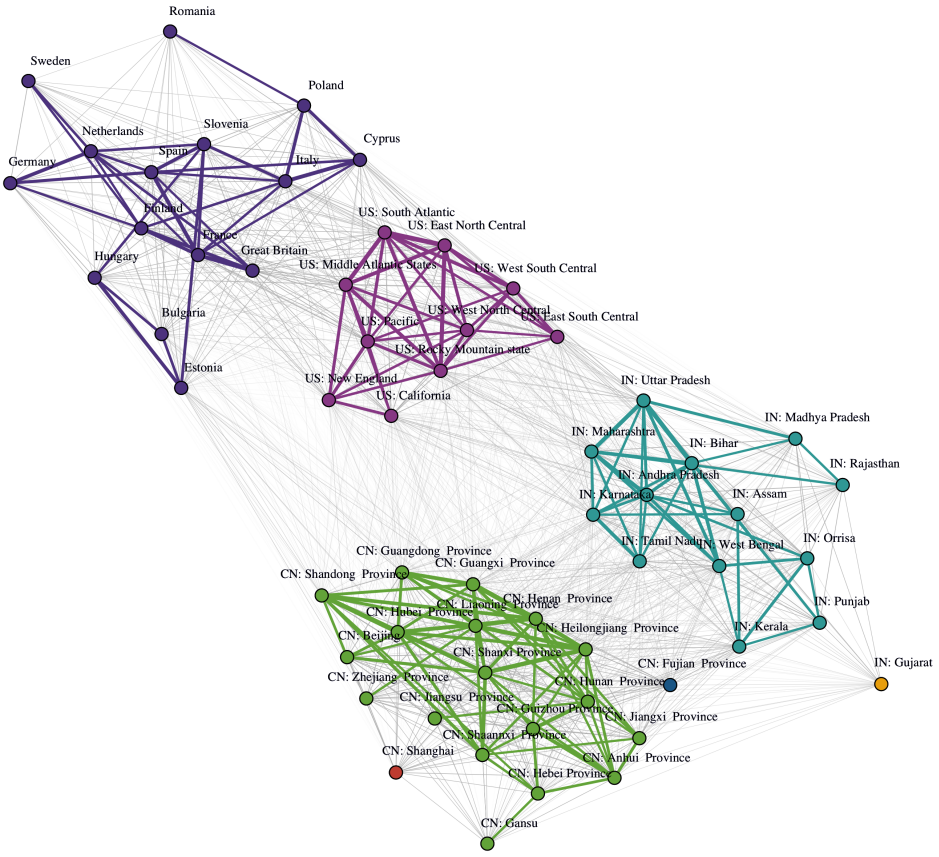

Figure 9: Community structure for cultural distance data.

In addition to viewing the local and global community structure as seen in Figure 9, the **pald** package allows for a two-dimensional display of cohesion against distance for the data, via the `dist_cohesion_plot` function, as seen below (Figure 10).

```
dist_cohesion_plot(cultures, mutual = TRUE)
```

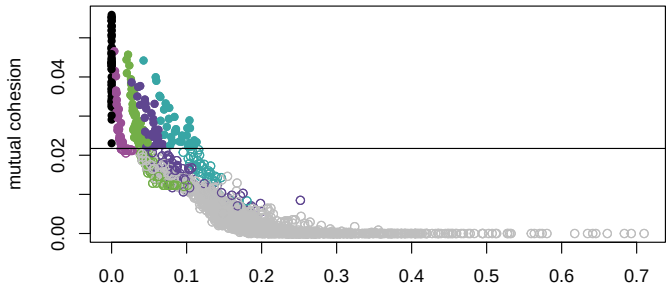

Figure 10: A plot of cohesion versus distance for the data. The identified communities are colored as in Figure 9.

Notice here that the magnitude of the distances within each of the identified communities varies substantially between regions; in fact, the most disparate two regions in the United States (at distance ≈ 0.027) are far closer than the two most similar in India (at distance ≈ 0.043). Despite this, India remains a cohesive whole, and locally disparate regions in

the United States such as East South Central and California are not strongly cohesive. For discussion of subtleties in local density (see Berenhaut et al. (2022). Additionally, currently available techniques require specification of parameters, as seen in the previous section. For further discussion of the cultural distance data in relation to community analysis see Berenhaut et al. (2022).

Given the inherent variation in density over regions, it is of interest to consider complementary results from density-based methods such as DBSCAN and its variant HDBSCAN. Employing the **dbscan** package (see Hahsler et al. (2019)), for DBSCAN with parameters `eps=0.34` and `minPts=3`, we have a result with five clusters and six regions being classified as noise. Therein, (excluding noise) the Indian region is split into two communities. Similarly, for HDBSCAN, with `minPts=3`, we have a result with five clusters and eight noise points. Here again (excluding the noise points), India has been split into two communities. The parameter choices used for both methods were determined by maximizing normalized mutual information (NMI; see Ana and Jain (2003)), when comparing with the underlying partition of the regions into Europe, United States, India and China, by regional information), giving values of 0.848 and 0.819, respectively. For comparison, the corresponding NMI value for the result in Figure 9 is 0.933. Further discussion is available in Berenhaut et al. (2022). It is important to note the required parameter choices for both DBSCAN and HDBSCAN and that results can be somewhat sensitive to changes in these. For instance, for DBSCAN, an increase in the value of `minPts` from 3 to 5, and a decrease in `eps` value from 0.34 to 0.30 leads to a large number of noise points (25), and a decrease in NMI value from 0.848 to 0.685. For HDBSCAN an increase in the value of `minPts` from 3 to 5 leads to 15 noise points, and a decrease in NMI value from 0.819 to 0.760, while a decrease in the value of `minPts` from 3 to 2 leads to 10 clusters and a decrease in NMI to 0.762.

0.7 Computational considerations

Computation of the cohesion matrix as implemented in the package is of order $O(n^3)$. The method is highly parallelizable, though, and recent work has resulted in extensive speed-up (see Devarakonda and Ballard (2024)). Approximations are also available (see Baron et al. (2021)). Note that the method is entirely deterministic, and one does not need to search a parameter space nor select initial values.

0.8 Summary

This paper introduces the **pald** package, demonstrating its utility for providing novel parameter-free community analysis which can easily be implemented for a variety of data sets, supplementing results from other methods for clustering, embedding, data depth, nearest neighbors, etc. Example code is provided along with discussion in the context of other commonly used R-based approaches.

References

C. Agostinelli and M. Romanazzi. Local depth. *Journal of Statistical Planning and Inference*, 141(2):817–830, 2011. [p39]

L. F. Ana and A. K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003. [p53]

J. D. Baron, R. Darling, J. L. Davis, and R. Pettit. Partitioned k-nearest neighbor local depth for scalable comparison-based learning. *arXiv preprint arXiv:2108.08864*, 2021. [p53]

A. V. Bell, P. J. Richerson, and R. McElreath. Culture rather than genes provides greater scope for the evolution of large-scale human prosociality. *Proceedings of the National Academy of Sciences*, 106(42):17671–17674, 2009. [p51]

K. S. Berenhaut and K. E. Moore. Communities in data. *SIAM News Blog*, 2022. [p37, 39]

K. S. Berenhaut, K. E. Moore, and R. L. Melvin. A social perspective on perceived distances reveals deep community structure. *Proceedings of the National Academy of Sciences*, 119(4), 2022. [p35, 36, 37, 38, 39, 41, 42, 49, 51, 53]

K. S. Berenhaut, J. D. Foley, and L. Lyu. Generalized partitioned local depth. *Journal of Statistical Theory and Practice*, 18(1):10, 2024. [p37, 38, 39]

R. J. Campello, P. Kröger, J. Sander, and A. Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1343, 2020. [p39]

L. L. Cavalli-Sforza, L. Cavalli-Sforza, P. Menozzi, and A. Piazza. *The history and geography of human genes*. Princeton university press, 1994. [p51]

A. Devarakonda and G. Ballard. Sequential and shared-memory parallel algorithms for partitioned local depths. In *Proceedings of the 2024 SIAM Conference on Parallel Processing for Scientific Computing*, pages 53–64, 2024. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611977967.5. [p53]

I. Dyen, J. B. Kruskal, and P. Black. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5):iii, 1992. doi: 10.2307/1006517. [p41, 47]

S. L. Feld. The focused organization of social ties. *American Journal of Sociology*, 86(5): 1015–1035, 1981. [p38]

A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1:1–30. [p41]

M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973. [p38]

M. Hahsler, M. Piekenbrock, and D. Doran. dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91:1–30, 2019. [p53]

R. Inglehart, C. Haerpfer, A. Moreno, C. Welzel, K. Kizilova, J. Díez-Medrano, M. Lagos, P. Norris, E. Ponarin, B. Puranen, et al. World values survey: All rounds–country-pooled datafile 1981-2014. 2014. [p41, 51]

M. Love and R. Irizarry. *tissuesGeneExpression. Subset of gene expression data*, 2015. URL https://github.com/genomicsclass/tissuesGeneExpression. [p41, 45]

M. N. McCall, K. Uppal, H. A. Jaffee, M. J. Zilliox, and R. A. Irizarry. The gene expression barcode: leveraging public data repositories to begin cataloging the human and murine transcriptomes. *Nucleic Acids Research*, 39(suppl_1):D1011–D1015, 2011. [p45]

M. N. McCall, H. A. Jaffee, S. J. Zelisko, N. Sinha, G. Hooiveld, R. A. Irizarry, and M. J. Zilliox. The gene expression barcode 3.0: improved data processing and mining tools. *Nucleic Acids Research*, 42(D1):D938–D943, 2014. [p45]

M. Muthukrishna, A. V. Bell, J. Henrich, C. M. Curtin, A. Gedranovich, J. McInerney, and B. Thue. Beyond Western, Educated, Industrial, Rich, and Democratic (WEIRD) psychology: Measuring and mapping scales of cultural and psychological distance. *Psychological Science*, 31(6):678–701, 2020. [p41, 51]

D. Paindaveine and G. Van Bever. From depth to local depth: a focus on centrality. *Journal of the American Statistical Association*, 108(503):1105–1119, 2013. [p39]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011. [p41]

M. J. Zilliox and R. A. Irizarry. A gene expression bar code for microarray data. *Nature Methods*, 4(11):911–913, 2007. [p45]

Lucy D'Agostino McGowan
Wake Forest University
Winston-Salem, NC
27106
mcgowald@wfu.edu

Katherine Moore
Amherst College
Amherst, MA
1002
kmoore@amherst.edu

Kenneth S. Berenhaut
Wake Forest University
Winston-Salem, NC
27106
berenhks@wfu.edu