Cluster analysis via random partition distributions

David B. Dahl 1, Jacob Andros 1, J. Brandon Carter 2

1Department of Statistics, Brigham Young University, Provo, Utah, USA
2Department of Statistics and Data Sciences, University of Texas at Austin, Austin, Texas, USA

Abstract
Hierarchical and k-medoids clustering are deterministic clustering algorithms defined on pairwise distances. We use these same pairwise distances in a novel stochastic clustering procedure based on a probability distribution. We call our proposed method CaviarPD, a portmanteau from cluster analysis via random partition distributions. CaviarPD first samples clusterings from a distribution on partitions and then finds the best cluster estimate based on these samples using algorithms to minimize an expected loss. Using eight case studies, we show that our approach produces results as close to the truth as hierarchical and k-medoids methods, and has the additional advantage of allowing for a probabilistic framework to assess clustering uncertainty. The method provides an intuitive graphical representation of clustering uncertainty through pairwise probabilities from partition samples. A software implementation of the method is available in the CaviarPD package for R.

KEYWORDS
dendrogram, Ewens–Pitman attraction distribution, hierarchical clustering, k-medoids clustering, random partition models

1 | INTRODUCTION
Cluster analysis seeks to partition data into distinct subsets, or clusters, with the aim of making observations in the same cluster more similar than observations from different clusters. Numerous clustering methods exist with applications in many fields [8, 11]. Common approaches to cluster analysis are either distance-based or model-based. Distance-based methods only require the pairwise distance between observations and are therefore relatively easy to apply in a variety of settings. However, since these methods are algorithmic in nature, a disadvantage is that they do not allow a probabilistic interpretation of clustering uncertainty. In contrast, model-based methods specify a sampling model which allows clustering uncertainty to be assessed in a probabilistic sense. While this approach allows flexibility, it can be challenging to tailor a statistical model for a particular dataset, and software may not be readily available. For example, in the context of mixed discrete and continuous variables, specifying the joint sampling model may be challenging.

In this paper, we combine benefits of both methods. We use a statistical model defined on pairwise distances to develop a novel clustering procedure that we call CaviarPD: Cluster Analysis via Random Partition Distributions. The method relies on sampling from a random partition distribution that is based on pairwise distances, that is, the Ewens–Pitman Attraction (EPA) distribution [4]. It then finds the best cluster estimate based on these samples using algorithms to minimize an expected loss. Note that CaviarPD uses, as inputs, the same pairwise distances as those in hierarchical and k-medoids clustering. Our approach has the advantage of being defined on pairwise distances (like hierarchical and k-medoids clustering), while still allowing for probabilistic assessment of uncertainty (like model-based approaches).
Using eight case studies, we show that, on average, CaviarPD performs as well as hierarchical and k-medoids methods in estimating the true partition of the data. In other words, CaviarPD’s ability to assess clustering uncertainty comes without a reduction in average estimation performance. A computationally-efficient implementation of the method in R is provided in the CaviarPD package [3] available on our GitHub repository at https://github.com/dbdahl/caviarpd. The software provides functions to obtain clustering estimates and to create graphical visualizations of uncertainty.

2 | EXISTING DISTANCE-BASED CLUSTERING METHODS

2.1 | Clustering concepts and terminology

We introduce common concepts used in both traditional clustering methods and CaviarPD. For a detailed description of current clustering practices, we suggest Hennig et al. [10]. A clustering \( c = (c_1, \ldots, c_n) \) gives labels for \( n \) items in which items \( i \) and \( j \) are in the same cluster if and only if \( c_i = c_j \). Equivalently, a partition \( \pi = \{S_1, \ldots, S_K\} \) of integers \( 1, \ldots, n \) is composed of mutually exclusive, non-empty, and exhaustive subsets such that \( i, j \in S \) implies that \( c_i = c_j \). We use the terms “clustering” and “partition” interchangeably and note that the term “cluster” is synonymous with “subset”.

In cluster analysis, we seek to cluster the \( n \) observations of a dataset \( D = \{x_1, \ldots, x_n\} \) into distinct groups so that observations within a group are more similar than observations from different groups. CaviarPD, like the other two methods we consider in this paper, relies on distance information between observations in order to partition items into subsets. The pairwise distances between observations \( x_i \) and \( x_j \) are calculated from a specified distance function \( d(x_i, x_j) \). One common distance function is Euclidean distance: \( d(x_i, x_j) = \sqrt{(x_i - x_j)^T(x_i - x_j)} \). The pairwise distances between all items can be stored in an \( n \times n \) distance matrix \( D \), where \( d_{ij} = d(x_i, x_j) \). The choice of distance metric is an important consideration in the analysis, but our primary task for this paper is to compare distance-based clustering methods given the user’s chosen distance matrix.

2.2 | Hierarchical clustering

In this section, we highlight the decisions a user must make with hierarchical clustering. For a thorough introduction to hierarchical clustering, see Rencher and Christensen [20] or Gareth et al. [9].

In agglomerative hierarchical clustering, each observation begins in its own cluster and the most similar clusters are sequentially merged until all data points are in a single cluster. The criteria used to define similarity between clusters are called the linkage and is computed from the pairwise distances between items in each cluster. Hierarchical clustering requires that a user decide which linkage to use and how to cut the dendrogram to obtain a partition estimate. For a more detailed explanation of the merging criterion for each linkage type, see Nielsen [18].

We demonstrate how the choice of linkage leads to highly varied clustering estimates. Figure 1 shows the resulting dendrograms from four common linkages applied to the wine recognition dataset using the function hclust in R [19]. The wine dataset contains many different chemical attributes for samples of wine from three different cultivars. The distance matrix was computed with Euclidean distance using all 13 chemical attributes. All four linkages produced dendrograms with very different clustering structures. Single linkage produces long chains of connected clusters. Ward linkage seeks to create compact spherical clusters such that the dendrogram shows more distinct branches [25]. For the wine dataset, the dendrograms from average and complete linkage show clustering structures between the long chains of single linkage and compact clusters of Ward linkage.

After choosing the type of linkage, the structure of the dendrogram is used to cut the tree to obtain a clustering estimate. From the dendrogram using Ward linkage, there appear to be three main clusters. Inspection of the other dendrograms leads to less definite conclusions about the number of clusters in the dataset as each tree varies drastically. Using complete linkage, one could reasonably argue for a cut of the tree that gives anywhere from 2 to 7 clusters. Langfelder et al. [13] introduce an automated solution to the tree-cutting problem with the dynamic tree-cut (DTC) algorithm. This procedure takes into account the structure of the tree to detect clusters and also allows for the tree to be cut at variable heights, providing flexibility in clustering estimation.

The lack of agreement between the linkages and tree-cutting estimates is prominent and concerning. There appeared to be three main clusters for Ward linkage, yet the DTC produced a default estimate of five clusters. The DTC applied to the complete linkage dendrogram resulted in an estimate with three clusters. Likewise, DTC applied to the average linkage dendrogram also resulted in three main clusters with a single observation in a fourth cluster by itself. Lastly, the single linkage tree resulted in a clustering estimate with all but three observations in a single cluster.
2.3 | K-medoids clustering

K-medoids clustering remains one of the most common and popular clustering approaches used today. The closely related k-medoids clustering, like k-means, seeks to minimize the sum of distances between points in a cluster and its cluster center. Unlike k-means, k-medoids use actual data points (exemplars or medoids) for the cluster centers. Thus, k-medoids only use the pairwise distance matrix as input, making it a comparable clustering approach to hierarchical clustering and CaviarPD.

Since there are \( \binom{n}{k} \) combinations of medoids that can be tested to minimize the total distance, finding the exact solution to the equation is difficult. Kaufman and Rousseeuw [12] proposed the PAM algorithm (partitioning around medoids) to conduct a nonexhaustive but greedy search through the combination space. The PAM algorithm was then modified by Schubert and Rousseeuw [22] to reduce computation time even further at the expense of some thoroughness in the search. Since the datasets in our case studies are all relatively small, we can easily use the original PAM algorithm for our k-medoids comparison.

When implementing k-medoids in our case studies, we use the silhouette method, a straightforward procedure for selecting \( k \) in k-means and k-medoids clustering [21]. The silhouette method is most reliable when the choices for \( k \) are narrowed down to a reasonable range using some prior intuition about the data. As with hierarchical clustering, the k-medoids approach offers no way to assess uncertainty in the results. We use the pam function in the \texttt{cluster} package [16] since it has a built-in calculation for silhouette width.

3 | CLUSTER ANALYSIS VIA RANDOM PARTITION DISTRIBUTIONS

We propose the CaviarPD method, a novel approach to clustering based on random partition distributions. CaviarPD is based on the EPA distribution, originally proposed as a prior distribution for random partitions in a Bayesian framework [4]. In the proceeding subsections, we explain how the EPA distribution is used as a component of the CaviarPD method. We describe how to find a clustering estimate using the samples from the EPA distribution.
Since the EPA distribution is a probability distribution over partitions, our simulation-based method provides the probabilities of pairs of items being clustered together, and we show how to use these for visualization of the clustering uncertainty. We also outline all other tuning parameters in the CaviarPD function and explain the proposed default values for these parameters.

3.1 | EPA distribution

In the EPA distribution, observations are sequentially allocated to subsets of a partition with probability proportional to the attraction of the item to that subset. We use \( \sigma = (\sigma_1, \ldots, \sigma_n) \) to denote the order of the \( n \) observations that are sequentially allocated into subsets in order to form a partition. Thus, \( \sigma_t \) is the \( t \)th observation allocated in the partition. This sequential allocation of items yields a sequence of partitions for each step of the allocation. Let \( \pi (\sigma_1, \ldots, \sigma_{t-1}) \) denote the partition of the first \( t - 1 \) observations at time \( t - 1 \). At each time \( t \), let \( K_t \) be the number of subsets in the partition \( \pi (\sigma_1, \ldots, \sigma_t) \). When \( t = 1 \), the first item is randomly assigned to an existing subset with probability \( 1 \) as shown in equation 2. For \( t > 1 \), \( \sigma_t \) can either be allocated to one of the existing \( K_{t-1} \) subsets or it can be allocated to a new subset in the partition. The probability mass function is conveniently expressed as the product of increasingly conditional probabilities:

\[
p (\pi_n | \alpha, \lambda, \sigma) = \prod_{t=1}^{n} p_t (\alpha, \lambda, \pi (\sigma_1, \ldots, \sigma_{t-1}))
\]

\[
p_t (\alpha, \lambda, \pi (\sigma_1, \ldots, \sigma_{t-1})) = \Pr (\sigma_t \in S | \alpha, \lambda, \pi (\sigma_1, \ldots, \sigma_{t-1}))
\]

\[
\begin{align*}
&= \begin{cases} 
\frac{\sum_{\sigma \in A} \lambda (x_{\sigma_t}, x_{\sigma_s}) / \sum_{i=1}^{t-1} \lambda (x_{\sigma_i}, x_{\sigma_i})}{\alpha + \tau} & \text{for } S \in \pi (\sigma_1, \ldots, \sigma_{t-1}) \\
\alpha & \text{for } S \text{ being a new subset}
\end{cases}
\end{align*}
\]

The probability that \( \sigma_t \) is allocated into a subset \( S \) is a function of the similarity function \( \lambda \) and the mass parameter \( \alpha > 0 \).\(^1\) The mass parameter \( \alpha \) plays an important role in the distribution of the number of clusters, with \( \alpha \to 0 \) yielding one cluster and \( \alpha \to \infty \) yielding \( n \) clusters. We discuss the mass parameter in more detail in Section 3.5. The similarity function \( \lambda (x_i, x_j) \) gives pairwise similarity between observations \( x_i \) and \( x_j \) for any \( x_i, x_j \in D \). Dahl et al. [4] propose a general class of similarity functions \( \lambda (x_i, x_j) = f (d_{ij}) \), where \( f \) is a non-increasing function of the pairwise distance \( d_{ij} \) between observations \( x_i \) and \( x_j \). \( d_{ij} \) is the same distance matrix \( D \) used as input for hierarchical and k-medoids clustering. Two common similarity functions are reciprocal similarity \( f(d) = d^{-\tau} \) and exponential similarity \( f(d) = \exp (-\tau d) \). The role of the similarity function \( \lambda (x_i, x_j) = f (d_{ij}) \) in (2) is to encourage items \( i \) and \( j \) with small distances \( d_{ij} \) to be in the same cluster. Finally, the parameter \( \tau \geq 0 \) is the temperature, which has the effect of accentuating or dampening the distance between items. As discussed in Section 3.5, we found results for our CaviarPD method to be robust to the choice of the similarity function and temperature parameter.

3.2 | Sampling from the EPA distribution

Sampling from the EPA distribution is a straightforward process [4]. Because items are allocated with probability proportional to the similarity to items in an existing subset, similar items are more likely to be clustered together. In order to sample a partition \( \pi_n \) from the EPA distribution, we begin with a permutation (an ordering) of the data and fixed \( \alpha \) and \( \lambda \). The first item \( \sigma_1 \) is allocated to a subset by itself with probability \( 1 \). The next item, \( \sigma_2 \) can either be assigned to the subset with \( \sigma_1 \) or to a new subset by itself. The probability of each allocation is given in (Equation 2). Let \( \sigma_2 \) be randomly assigned to the existing subset or new subset with the respective probabilities. For each subsequent item, the item \( \sigma_{t+1} \) is randomly assigned to an existing subset or new subset with probability respective to being assigned to \( S_1, S_2, \ldots, S_K, S_{K+1} \). Here \( S_1, \ldots, S_K \) represent the existing subsets of the partition and \( S_{K+1} \) is a new subset. Continue to sequentially assign the items until a partition of the data is obtained. The resulting partition is a single draw from the EPA distribution. Sampling can be parallelized over many cores to simultaneously obtain many draws from the EPA distribution.

The order by which the data are sampled affects the resulting probabilities of obtaining particular partitions. To remove this dependence on sampling order, randomize the order by which the items are allocated into subsets for each draw. This has the effect of making the probability of each partition independent of any particular permutation of the data.

3.3 | Visualizing pairwise probabilities

A key advantage of CaviarPD over traditional clustering is its ability to quantify and visualize uncertainty in clustering estimates. This can be done using a heat map based on the samples from the EPA distribution.

Each of the partitions \( \pi \) can be represented as an \( n \times n \) association matrix denoted \( \gamma (\pi) \), where the \((i, j)\) element of
the association matrix is 1 if \(x_i\) and \(x_j\) are in the same cluster and 0 otherwise. Concisely, \(\gamma_j(\xi) = I(c_i = c_j)\). For \(B\) samples from the EPA distribution, there are \(B\) association matrices \(\gamma(\xi)\). These matrices \(\gamma(\xi_1), \ldots, \gamma(\xi_B)\) can then be averaged together element-wise to create a pairwise similarity matrix, which contains the estimated pairwise probabilities that items appear in the same subset for a given \(\alpha\) and \(\lambda\). Thus, the pairwise similarity matrix is an \(n \times n\) matrix denoted \(\Psi\), where the \(i\)th, \(j\)th element is the relative frequency with which items \(i\) and \(j\) are clustered together among the samples.

\[
\Pr(c_i = c_j) \approx \Psi_{ij} = \frac{1}{B} \sum_{k=1}^{B} \gamma_{ij}(\xi_k) \quad (3)
\]

Each element of the \(\Psi\) matrix is the estimated probability that observations \(x_i\) and \(x_j\) are in the same cluster for a given \(\lambda\) and \(\alpha\). \(\Psi\) can then be conveniently visualized in the form of a heat map. Choosing the number of samples \(B\) typically depends on the size of the clustering dataset and the computational resources available to the user. In short, as \(B\) increases the variability in the estimated pairwise probabilities in \(\Psi\) decreases, though at the expense of increased computation time. We give more details on choosing this parameter in Section 3.6.

In the heat map, the color of a cell represents the probability that two items are clustered together while also highlighting the actual clustering estimate. In Figure 2 we can see clearly that, for \(\alpha = 0.90\), there appear to be three distinct clusters. In this visualization, the observations are ordered to group similar items together, creating the block diagonal structure. The heat map becomes even more useful when the observations are ordered by an estimated partition, thereby showing the probability relationships within and between clusters. For example, clusters 1 and 2 appear to be more similar than clusters 1 and 3 because there is a darker shade of yellow in the off-diagonal blocks for clusters 1 and 2, and a near-white shade of yellow in the off-diagonals for clusters 1 and 3.

### 3.4 Partition estimation from samples

The Bayesian literature provides multiple methods for obtaining a point estimate of a random partition based on samples from a posterior partition distribution \([6]\). Our approach here is not Bayesian, but we can draw upon the Bayesian literature on random partitions to obtain a representative point estimate of a partition based on samples from the EPA distribution.

In decision theory, a loss function is specified in order to pick an optimal estimate that incurs a minimal expected loss. For partitions, loss functions evaluate how distant the estimated partition is from the true partition of the data. Binder loss is a function of the number of disagreements between the estimated and true partition for all possible pairs of observations \([1]\). The function is a weighted sum of the two types of disagreements: observations in different clusters when they should be in the same cluster with weight \(a\), and observations in the same cluster when they should be in different clusters with weight \(b\). In practice, the constraint \(a + b = 2\) is used to give the default \(a = b = 1\). Wade and Ghahramani \([24]\) demonstrate that when the weights of the two errors are equal, \(a = b\), then the partition that minimizes Binder loss is given by:

\[
\hat{\xi}_{\text{binder}} = \arg\min_{\xi} \sum_{i < j} (\gamma_{ij}(\xi) - \Psi_{ij})^2.
\]

Wade and Ghahramani \([24]\) also propose using the variation of information, introduced by Meilă \([17]\) as a loss function. The variation of information is developed from information theory and is the information present in both clusters (partition entropy) minus the information shared between the two clusters. Minimization by enumeration over the whole partition space is unfeasible, so we use the SALSO method \([6]\) as implemented in the R package \texttt{salso} \([5]\). Although the SALSO method was designed for Bayesian posterior inference, it applies equally well in our non-Bayesian context here. The SALSO method seeks to find a partition estimate which minimizes the expected loss over any number of clusters. As such, the user does not have to specify the number of clusters, although our software allows one to target the search to a range or a specific number of clusters.
Most parameters in the EPA distribution have a default value for the purposes of CaviarPD. Dahl et al. [4] showed that the effect of the temperature \( \tau \) in the EPA distribution eventually asymptotes in terms of the expected number of subsets, yielding virtually identical results for \( \tau \) equal to, for example, 100 or 1000, and often even lower than that. The software default for the temperature is set at \( \tau = 100 \). There are two common similarity functions for \( \lambda \), namely, exponential and reciprocal similarity. There are many other similar functions that could be explored, however, all clustering estimates in our case studies were robust to the choice of the similarity function, so we simply use the exponential similarity throughout the paper. Our algorithm also scales the pairwise distance matrix such that the median distance is 1.

The mass parameter \( \alpha \), however, does not have a clear default value and is highly influential in determining the number of subsets in a partition estimate. In practice, \( \alpha \) should be selected such that, once a clustering is obtained, the associated heat map shows that items in the same cluster are clustered together with high probability. The best values of \( \alpha \) should ideally show a clear distinction between subsets of the partition in the heat map (i.e., low probability regions between clusters). Figure 3 shows two heat maps for the pairwise probabilities and clustering estimates of the wine dataset with the mass parameter set at 0.9 and 0.5, respectively. When the mass is 0.9, the resulting heat map shows overall higher pairwise probabilities within clusters as compared to the mass set at 0.5. Likewise, there is less variance of the pairwise probabilities within clusters when the mass is 0.9. The estimated three clusters appear more distinct because of the greater within-cluster pairwise probabilities. Selecting a single \( \alpha \) to find an optimal clustering and heatmap can be challenging. Additionally, selecting \( \alpha \) is not very intuitive in how it relates to the pairwise probabilities and number of clusters in the SALSO estimate. To address these challenges, while capitalizing on the pairwise probabilities from CaviarPD, we use an ensemble method to obtain a clustering estimate across a reasonable range of values of \( \alpha \).

The approach that we now described decreases the reliance on a single value of \( \alpha \) and also decreases the variability of our estimates. Rather than specifying a value for \( \alpha \) directly, we allow users to specify a range of the number of clusters to be considered in the estimation. The EPA distribution has a closed-form expression for the expected number of clusters given a value of \( \alpha \) [4]. CaviarPD builds a grid of expected number of clusters across the user-supplied range, and then for each grid value inverts the closed-form expression for the expected number of clusters to obtain the corresponding mass \( \alpha \). Then, for each mass \( \alpha \), \( B \) draws from the EPA distribution are obtained and then used in the SALSO algorithm to obtain a clustering estimate by minimizing the expected loss. To ensure that the clustering estimate for each mass value \( \alpha \) has a number of clusters within the user-specified range, we allow the algorithm to tune the weight \( a \). In both Binder and VI loss, \( a \) controls the cost of failing to cluster two items that should be together [5]. Tuning \( a \) allows us to adjust the number of clusters in the SALSO estimate as \( a < 1 \) tends to yield estimates with more clusters and \( a > 1 \) tends towards fewer clusters. The ensemble aspect of the algorithm comes in as we then run SALSO again with \( a = b = 1 \) using the clustering estimates from each mass value as input. This gives us a clustering estimate that is an “average” of the clustering estimates obtained across the grid of mass values.

Of course, if preferred, the user may outright fix the mass \( \alpha \) or set a grid of mass values, although we expect users will have stronger intuition about the number of clusters than about the mass \( \alpha \).
3.6 Tuning parameters

As discussed in the preceding sections, results are robust to the temperature and similarity function, and we use temperature $\tau = 100$ and exponential similarity here. The mass parameter $\alpha$ is selected as described in the previous section. Beyond the parameters of the EPA distribution, however, there are two main tuning parameters in the CaviarPD algorithm. First, the user can select the number of samples $B$ and second, the user specifies the range for the number of clusters to consider with its associated grid length. The number of samples $B$ indicates how many partitions are taken from the EPA distribution for each mass $\alpha$, and the grid length specifies the number of mass values for which sampling is performed. This means that the number of total samples drawn from the EPA distribution is equal to the product of these two values. The runtime of the algorithm is heavily dependent on the number of total samples obtained for the search, so increasing either or both of these values offers the potential for more accurate estimation at the expense of additional computation. In other words, the number of samples and the grid length can be adjusted by the user to balance the competing priorities of stable estimation and computational efficiency.

As defaults, we suggest $B = 200$ and a grid length of 5 yielding 1000 total samples for the search. For small datasets, these inputs could easily be increased; for larger datasets, these values may need to be lowered for the sake of computational time. $B = 200$ samples and a grid length of 5 were chosen as defaults for the algorithm because we found it to balance the stability and efficiency of the algorithm for our datasets. The runtime for CaviarPD also tends to increase, and so high values of $B$ or $k$ could prompt the user to lower $B$ or the grid length. Of course, $k$ is not known in practice, and $n$ is the most important factor in deciding $B$, unless the user already has prior intuition of a range that $k$ might fall within. The dimensionality of the observations does not need to be a consideration in altering these defaults because the pairwise distance matrix is only computed once, and can then be used by the algorithm as much as needed without going back to the original data. In the appendix, we give the results of cluster estimates from the wine dataset using different values for the number of samples and the grid length.

The remaining parameters for the CaviarPD algorithm are usually less pivotal and are detailed in the package documentation. In summary, when calling the `CaviarPD` function in its simplest form, the user needs only to supply two arguments: (i) a matrix of the pairwise distances and (ii) a range of the number of clusters to be considered. In our experience, the default values for the other inputs are sufficient in the majority of applications.

4 Simulation study

We now describe a simulation study to demonstrate that our CaviarPD method provides reasonable clusterings when data is generated according to known clusterings. We simulated data from the model described in Section 6.1 of Wade and Ghahramani [24]. Data $D = \{x_1, \ldots, x_n\}$ were independent and identically distributed from a four-component mixture model with equal mixture weights. The mixture components were bivariate Gaussians with means $(\pm 2, \pm 2)$, no covariances, and variances that were equal (in the first scenario) to $1/m$ or equal (in the second scenario) to $1/m$, $1/m$, $0.5^2/m$, and $1.5^2/m$ in the four quadrants. Wade and Ghahramani [24] use a fixed value $m = 1$, but we let $m$ vary here to show that, as $m$ goes to infinity, the expected value of Binder and VI loss goes to zero, indicating that CaviarPD perfectly recovers the true clusterings. For each value of $m$, we generated 5000 datasets of $n = 200$ observations and noted the true clusterings that generated the datasets. We scaled the data to have mean zero and variance one and then computed the pairwise Euclidean distances between all pairs of observations. We computed the mean Binder and mean VI for each value of $m$. For small $m$, the clusters overlapped, and as such, a perfect recovery of the true clusterings was not expected. Figure 4 shows that, as $m$ increased (i.e., as the variance of the mixture components went to zero), the losses converged to 0, meaning that CaviarPD (solid curves) perfectly recovers the true clustering. For the sake of comparison, we find that k-medoids (dashed curves) show similar behavior. These two scenarios are rather simplistic, but in the next section, we explore CaviarPD’s relative performance in real-world datasets.

5 Case studies

To compare CaviarPD with the other distance-based clustering methods, we evaluate how well each method clusters data where the true partition of the data is known. Datasets were obtained from the UC Irvine Machine Learning Repository [7]. The pairwise distance matrix was computed for all eight case studies. For each of the datasets with numeric attributes, we centered and scaled the data and computed the Euclidean pairwise distances. Though not always necessary, this helps safeguard against
attributes with large variances carrying more weight in the distance computation. For datasets with categorical attributes, we computed the Jaccard pairwise distances.

In order to evaluate the quality of the estimates, we compute the Binder and VI loss between each estimate and the true partition. Both of these measures are nonnegative, where lower values are indicative of a partition estimate that is more similar to the truth partition. We run hierarchical clustering with average, complete, and Ward linkages. Because the possibilities for tree cutting are numerous, we cut the dendrograms using the default settings for the cutreeDynamic function in the dynamicTreeCut package in R to obtain a partition estimate for each of the linkages [14]. We also choose k in k-medoids using the silhouette method.

Finally, we run each cluster method for each case study 200 times. For the deterministic algorithms (namely, hierarchical clustering and k-medoids), this gives an estimate of the mean runtime and the variability in the runtime, for the sake of comparison with CaviarPD. More importantly, the repeated simulations give us an understanding of the internal stability of CaviarPD, a stochastic algorithm, for which estimates can vary from run to run (which is discussed further in Section 6.2).

5.1 | Wine dataset

Table 1 gives the clustering results for the wine dataset, which was used for demonstration in previous sections and contains 13 chemical attributes on wines from three different cultivars. For the wine dataset, CaviarPD approaches the true partition more closely than Ward or complete linkage, but not as well as average linkage or k-medoids. However, the partition estimates from hierarchical clustering are inconsistent in their results. Without knowing the true partition of the data, we would not know that average linkage produces the best estimate. Hence, hierarchical clustering may be less reliable than CaviarPD or k-medoids for the wine dataset, due to the ambiguity in choosing the linkage.

5.2 | House dataset

The house vote’s dataset contains voting records for the 1984 House of Representatives. The class attribute is party affiliation, Republican or Democrat. Having only two clusters yet nearly 500 total observations, this data provides a unique test for all clustering methods. Results are displayed in Table 1. CaviarPD and average linkage produce estimates with nearly identical loss metrics. However, note that CaviarPD under VI loss is usually able to identify the correct number of clusters at 2 (along with k-medoids), whereas average linkage does not. As with the wine dataset, the clustering variation between linkage choices is concerning. Without knowing the true partition of the data, a hierarchical clustering implementation could be just as likely to use complete or Ward linkage instead of average, resulting in a less informative clustering.

5.3 | Flea-Beetle dataset

For some datasets, it is not uncommon to perfectly recover the true partition in the estimation process. As an example, we take the flea-beetle dataset, which contains measurements on three different species of beetles. It contains only 74 observations in total. Only Ward linkage with DTC is able to estimate the exact partition of the data. Table 2
Table 1: Clustering results for the wine and house votes datasets, where the number of clusters for a particular estimate is shown in the column labeled $\hat{k}$. Since CaviarPD is a stochastic algorithm, we report the average number of clusters $\hat{k}$ over the 200 simulations as well as the average loss, and also supply the standard deviation of the loss (SD). The reported runtime is the average wall time for each method over the 200 simulations.

| Wine ($n = 178, k = 3$) | House votes ($n = 435, k = 2$) |
|-------------------------|-------------------------------|
| $\hat{k}$ | Binder | VI | SD | Runtime | $\hat{k}$ | Binder | VI | SD | Runtime |
| CaviarPD (Binder) | 2.8 | 0.15 | 0.79 | 0.08 | 0.71 | 2.4 | 0.22 | 1.01 | 0.01 | 2.04 |
| CaviarPD (VI) | 3.1 | 0.13 | 0.74 | 0.15 | 0.70 | 2.1 | 0.22 | 1.00 | 0.03 | 1.93 |
| Average | 4.0 | 0.09 | 0.69 | —— | 0.11 | 3.0 | 0.22 | 1.03 | —— | 0.25 |
| Complete | 3.0 | 0.19 | 1.21 | —— | 0.11 | 10.0 | 0.43 | 2.82 | —— | 0.26 |
| Ward | 5.0 | 0.17 | 1.23 | —— | 0.12 | 6.0 | 0.37 | 2.20 | —— | 0.24 |
| k-medoids | 3.0 | 0.12 | 0.68 | —— | 0.14 | 2.0 | 0.23 | 1.10 | —— | 0.34 |

Table 2: Clustering results for the flea and olive datasets, applying the same interpretations from Table 1.

| Flea-Beetle ($n = 74, k = 3$) | Olive ($n = 572, k = 9$) |
|-----------------------------|-------------------------|
| $\hat{k}$ | Binder | VI | SD | Runtime | $\hat{k}$ | Binder | VI | SD | Runtime |
| CaviarPD (Binder) | 2.9 | 0.03 | 0.11 | 0.06 | 0.39 | 9.7 | 0.05 | 0.89 | 0.01 | 6.81 |
| CaviarPD (VI) | 3.0 | 0.02 | 0.06 | 0.17 | 0.40 | 8.6 | 0.05 | 0.82 | 0.02 | 6.25 |
| Average | 2.0 | 0.19 | 0.83 | —— | 0.04 | 10.0 | 0.07 | 1.05 | —— | 0.42 |
| Complete | 2.0 | 0.27 | 1.08 | —— | 0.05 | 14.0 | 0.12 | 1.69 | —— | 0.45 |
| Ward | 3.0 | 0.0 | 0.0 | —— | 0.03 | 9.0 | 0.11 | 1.25 | —— | 0.37 |
| k-medoids | 3.0 | 0.02 | 0.17 | —— | 0.01 | 7.0 | 0.05 | 0.94 | —— | 0.48 |

shows that this results in the comparative loss functions, both Binder and VI, being equal to 0. CaviarPD also frequently estimates the exact partition under both losses, but its reported losses are averaged across the 200 simulations, skewing upward due to the occasional erroneous estimate. k-medoids do not have the same linkage-choice problem that hierarchical clustering does, but it does fall just short of estimating the exact partitioning of the data.

5.4 Olive dataset

With the olive dataset, we demonstrate the use of the pairwise similarity matrix to detect clusters within subsets of the estimated partition. The olive dataset contains measurements of the levels of different oils in olives from nine different regions of Italy.

Table 2 shows the results for the olive dataset. The only method that always estimates 9 clusters are Ward linkage; however, Ward linkage misclassifies too many observations to be considered as accurate as the other estimates. The CaviarPD estimate has the lowest Binder and VI loss, but it also has some variation in the number of clusters it estimates. k-medoids have difficulty separating some pairs of regions, resulting in an estimate with only 7 clusters. On the other hand, the average linkage DTC estimate concentrates far too many observations in the first 3 clusters, while also creating a 10th cluster with only a few observations. Unlike other methods, however, the heat maps from CaviarPD’s pairwise probabilities can be used to detect subcluster structures and assess problematic points in the estimation. These subclusters are especially prevalent in the olive dataset, and so we provide a heat map from one of the CaviarPD estimates (using Binder loss) in Figure 5A. In cluster 7 of the heat map, there appear to be two higher probability regions of items being grouped together within each cluster. In other words, cluster 7 may have merged olives from two different regions. The fact that clusters 3 and 5 are extremely small (not even visible in the heat map) could indicate that some of these observations were misallocated into cluster 7. Of course, post-processing is also possible in hierarchical clustering; however, these decisions would not be based on probabilities.

Furthermore, CaviarPD allows the user to target a range for the number of clusters (or a specific value) for the estimation. Therefore, upon noting merged clusters through the pairwise probability plot, the user could then
re-run CaviarPD on the olive dataset, but with the target number of clusters being larger.

### 5.5 Additional datasets

The four additional datasets, whose results are shown in Table 3, further demonstrate the efficacy of CaviarPD for partition estimation. The yeast dataset contains 10 subsets in the true partition which represent the protein localization site in yeast cells. Ward and complete linkage rank as the best estimates in terms of Binder loss because they notably overestimate the number of clusters, for which Binder penalizes less heavily than VI. However, these estimates are then the worst when ranked by VI loss. In most applications, an estimate of 19 clusters would be less interpretable than an estimate with few clusters, as represented in the poor performance by VI loss. By that standard, CaviarPD performs better than all other methods, and it is the only method that manages (in some instances) to correctly identify 10 clusters.

The yeast case study is an example of a larger dataset (containing approximately 1500 observations) that takes CaviarPD (with default settings) around 57 s (on average) to produce its clustering estimate. Though the algorithm is still feasible for a dataset of this size, it may not be...
practical for significantly larger sets without overriding the default values. In such cases, users may consider lowering the number of samples $B$ below the default of 200. Or, they could build heat maps for 3–5 different values of $a$ and select the mass for the plot with the most concentrated pairwise probabilities. It is also worth noting that, as evidenced by the yeast case study, neither CaviarPD nor k-medoids scale to large datasets (in terms of their computation time) as well as hierarchical clustering does.

The lymphography dataset contains information on four different classifications of lymph nodes and serves as an example in which the cluster distinctions are not well-represented by the pairwise distances. This leads to poor estimates by all methods, and a less interpretable heat map as given in Figure 5B. Regardless, CaviarPD still ranks at the top of all methods in terms of both losses. The E. coli dataset contains attributes for eight different localization sites of proteins in E. coli bacteria. In this example, CaviarPD is the only method to obtain an 8-cluster estimate multiple times throughout the simulations. It is also closer to the truth than all other estimates terms of both in Binder and VI loss. Finally, the glass dataset contains chemical composition measurements taken on six different glass products. The glass example once again demonstrates the problems with linkage selection in hierarchical clustering. Without knowing the true partition, the choice of linkage is ambiguous, and the user would only get a better clustering than CaviarPD (in terms of Binder loss) by selecting Ward linkage. Similarly, the user would only get a better clustering in terms of VI loss by selecting average or complete linkage. CaviarPD and k-medoids are less ambiguous on this matter and have very similar loss metrics for both Binder and VI.

## 6. DISCUSSION

### 6.1. Ranking the methods

We earlier made the assertion that CaviarPD tends to produce estimates as close to the truth as hierarchical and k-medoids methods. Having examined the performance of CaviarPD under each of these eight case studies, we return to this assertion and find, in fact, that CaviarPD even is somewhat better on average. In some case studies, CaviarPD was notably outperformed by one of the linkages or by k-medoids, and in other cases, it was the best method. In many examples, CaviarPD produced the most accurate clusterings under one type of loss but not both. To summarize the overall performance of each method, we give the average ranking of each method in Table 4. Under both losses, CaviarPD was the top-ranked method in terms of having the lowest loss to the truth. This does not mean that CaviarPD was the best method in every case study, but rather that averaged across all our case studies, it came the closest to recovering the true clustering.

### Table 4

| Method               | Rank | Variation of information |
|----------------------|------|--------------------------|
| CaviarPD (Binder)    | 2.250| CaviarPD (VI)            |
| Ward linkage         | 2.625| k-medoids                |
| k-medoids            | 2.625| Average linkage          |
| Average linkage      | 3.500| Complete linkage         |
| Complete linkage     | 4.000|                          |

### 6.2. Clustering stability

Clustering stability is the notion of how similar clustering estimates are when repeatedly sampling from the same population. Evaluating the stability of our CaviarPD method is a twofold problem. Note that our algorithm is stochastic in nature and we first must assess how similar its estimates are in the repeated application of CaviarPD to the same dataset. Second, as a more traditional measure of stability, we want to estimate the expected distance between clustering estimates from two samples from the same data-generating process [23].

To quantify the uncertainty associated with the stochasticity of the algorithm, that is, its internal stability, we calculated standard deviations of the loss function values for each dataset. The standard deviations of the loss (a measure of internal stability) are given in Tables 1–3 in the “SD” column.

We also evaluate the stability of CaviarPD and the competing clustering algorithms in the traditional sense; that is, the expected difference between clustering estimates from two samples in the same data-generating process. To estimate this expectation, we follow the subsampling method of Levine and Domany [15]. For each run in the simulation, we took a subsample of 50% of the observations from the dataset. Each clustering algorithm was applied to the subsample to produce a clustering estimate. We then calculated the distance (Binder and VI loss) between the subsampled clustering estimate and the estimate obtained from the full dataset, where only observations and cluster labels from the subsample were part of the loss calculation.
To account for the stochasticity of CaviarPD, we additionally reran the CaviarPD estimate on the full dataset for each run of the simulation. Thus our stability estimates for CaviarPD under Binder and VI loss have marginalized the stochasticity of CaviarPD. Table 5 contains the results of the stability study.

Out of our eight case study datasets when measuring stability under Binder loss, CaviarPD with Binder and VI loss produced the first and second most stable estimates six times (CaviarPD using VI loss coming in first in four out of those six case studies). For both the wine and glass products datasets, CaviarPD using Binder and VI loss came in second and third, respectively. For five of the eight case study datasets with stability measured under VI loss, CaviarPD using VI loss produced the most stable estimates, and CaviarPD using Binder loss the second most stable estimates. For the lymphography dataset, CaviarPD Binder produced the most stable and CaviarPD VI the second most stable estimates. CaviarPD VI came in third and fourth for the wine and glass products datasets respectively, while CaviarPD Binder came in second and third. No other clustering method consistently produces stability results better than CaviarPD (using either VI or Binder loss) for these eight case studies. While we are not using stability as a means to select a clustering estimate, stability is a desirable quality of a clustering algorithm. We want clustering estimates to be similar across datasets from the same data-generating mechanism. CaviarPD produces stable estimates in part due to the ensemble built into the algorithm. As described in Section 3.5, the final clustering estimate from CaviarPD is an “average” of all the clustering estimates generated that fall within the user-specified range of number of clusters. Ensembles often perform better, as they are less susceptible to chance variation that occurs from a single run.

### 6.3 Other distributions

The EPA distribution is suited for the CaviarPD method since it is a random partition distribution taking pairwise distances as input. Most other random partition distributions do not take pairwise distances as input and would therefore not be suitable for CaviarPD. The distance-dependent Chinese Restaurant Process (ddCRP), however, is one other partition distribution defined by pairwise distances [2]. We explored the possibility of using the ddCRP distribution as a basis for CaviarPD, but the clustering estimates from the ddCRP distribution were not any more accurate than those from the EPA and were highly sensitive to the tuning parameters. Further, the pairwise probability plots showed less distinct cluster separation. As such, we do not recommend using the ddCRP distribution for the CaviarPD method.
7 CONCLUSION

CaviarPD does not always produce the most accurate clustering from any given dataset. However, each case study demonstrated various areas in which CaviarPD performed well compared to the other methods. In the wine and house datasets, the estimates from hierarchical clustering had a concerning amount of variation between linkage choices, and there was typically only one linkage that outperformed CaviarPD and k-medoids. In the yeast and lymphography datasets, CaviarPD sometimes underestimated the true number of clusters and had higher Binder loss, but also had the best VI loss. Additionally, CaviarPD provided clusterings that, in many cases, might be more informative than the alternatives which severely overestimated the number of clusters. The olive and E. coli datasets saw CaviarPD strictly outperform all other methods on both losses, although k-medoids and hierarchical clustering still have more consistency in estimation since they are deterministic algorithms. Ranking each competing method by its loss to the true partition showed that CaviarPD was the best-ranking method when averaged over the eight case studies. Perhaps most importantly, in all eight examples, CaviarPD has the ability to assess clustering uncertainty through pairwise probability plots.

In summary, CaviarPD provides a novel alternative to existing distance-based clustering methods. Using a variety of case studies, we find that CaviarPD performs as well as k-medoids and hierarchical clustering on average. However, the advantage of our method is that it provides a probabilistic interpretation of the results. In other words, CaviarPD’s ability to assess clustering uncertainty comes without a reduction in estimation performance.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

David B. Dahl © https://orcid.org/0000-0002-8173-1547
J. Brandon Carter © https://orcid.org/0000-0003-1687-0564

REFERENCES

1. D. A. Binder, Bayesian cluster analysis, Biometrika 65 (1978), no. 1, 31–38. https://doi.org/10.1093/biomet/65.1.31.
2. D. M. Blei and P. I. Frazier, Distance dependent chinese restaurant processes, J. Mach. Learn. Res. 12 (2011), no. 8, 2461–2488.
3. D. B. Dahl, B. Carter, and J. Andros, 2021: Caviarpd: cluster analysis via random partition distributions. R Package Version 0.2.9.
4. D. B. Dahl, R. Day, and J. W. Tsai, Random partition distribution indexed by pairwise information, J. Am. Stat. Assoc. 112 (2017), no. 518, 721–732. https://doi.org/10.1080/01621459.2016.1165103.
5. D. B. Dahl, D. J. Johnson, and P. Müller, 2021: salso: Search Algorithms and Loss Functions for Bayesian Clustering. R package version 0.2.22. URL https://CRAN.R-project.org/package=salso
6. D. B. Dahl, D. J. Johnson, and P. Müller, 2021: Search algorithms and loss functions for bayesian clustering.
7. D. Dheeru and E. Karra Taniskidou, 2017: UCI machine learning repository. URL http://archive.ics.uci.edu/ml
8. R. Dubes and A. K. Jain, Clustering techniques: the user's dilemma, Pattern Recogn. 8 (1976), no. 4, 247–260. https://doi.org/10.1016/0031-3203(76)90045-5. URL, http://www.sciencedirect.com/science/article/pii/0031320376900455.
9. J. Gareth, D. Witten, T. Hastie, and R. Tibshirani, An introduction to statistical learning. Springer texts in statistics, Springer, New York, 2013.
10. C. Hennig, M. Meila, F. Murtagh, and R. Rocci, Handbook of cluster analysis. Chapman & Hall/CRC handbooks for modern statistical methods, Taylor & Francis, New York, 2016.
11. A. K. Jain, M. N. Murty, and P. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999), 264–323.
12. L. Kaufman and P. J. Rousseeuw, Finding groups in data: an introduction to cluster analysis, John Wiley & Sons Inc., Hoboken, New Jersey, 1990.
13. P. Langfelder, B. Zhang, and S. Horvath, Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for R, Bioinformatics 24 (2008), no. 5, 719–720. https://doi.org/10.1093/bioinformatics/btm563.
14. P. Langfelder, B. Zhang, and S. Horvath 2016: dynamicTreeCut: Methods for Detection of Clusters in Hierarchical Clustering Dendrograms. R package version 1.63–1. URL https://CRAN.R-project.org/package=dynamicTreeCut
15. E. Levine and E. Domany, Method for unsupervised estimation of cluster validity, Neural Comput. 13 (2001), 2573–2593. https://doi.org/10.1162/089976601753196030.
16. M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik, 2021: cluster: Cluster Analysis Basics and Extensions. R package version 2.1.2 — For new features, see the “Changelog” file (in the package source). URL https://CRAN.R-project.org/package=cluster
17. M. Meilă, Comparing clusterings: an information based distance, J. Multivar. Anal. 98 (2007), no. 5, 873–895. https://doi.org/10.1016/j.jmva.2006.11.013. URL, http://www.sciencedirect.com/science/article/pii/S0047259X06002016.
18. F. Nielsen, Introduction to HPC with MPI for data science. Undergraduate topics in computer science, Springer, New York, 2016.
19. R Core Team, R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria. URL. 2021. https://www.R-project.org/.
20. A. C. Rencher and W. F. Christensen, Methods of multivariate analysis. Wiley series in probability and statistics, 3rd ed., John Wiley & Sons, Hoboken, New Jersey, 2012.
21. P. J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987), 53–65.
22. E. Schubert and P. J. Rousseeuw, Faster k-medoids clustering: improving the pam, clara, and clarans algorithms, Springer, Cham, Switzerland, 2019, 171–187.
APPENDIX A. CAVIARPD WITH ALTERNATIVE TUNING PARAMETERS

To illustrate the effect that each of the turning parameters on the estimation performance and runtime, we use the wine and house datasets as examples in Table A1. In the wine dataset, increasing the number of samples or the grid length results in lower average loss across our simulations. This comes at the expense of increased runtime, but all the runtimes are still under 1 s. In contrast, the clustering for the housing dataset does not improve in estimating the true partition when these parameters are increased. The increase in runtime for the house example is also much more noticeable than it was in the wine example. (Recall that the housing dataset is larger than the wine dataset.) In summary, increasing the grid length and/or number of samples may be beneficial for some datasets, but not others.

### Table A1

| B   | Binder Loss | Binder Time | VI Loss | VI Time |
|-----|-------------|-------------|---------|---------|
| 50  | 0.18        | 0.08        | 0.74    | 0.08    |
| 100 | 0.16        | 0.13        | 0.74    | 0.14    |
| 200 | 0.15        | 0.25        | 0.75    | 0.26    |
| 400 | 0.14        | 0.37        | 0.74    | 0.39    |
| Grid length | Loss | Time | Loss | Time |
| 5   | 0.15        | 0.25        | 0.75    | 0.26    |
| 10  | 0.14        | 0.44        | 0.75    | 0.44    |
| 20  | 0.13        | 0.85        | 0.71    | 0.95    |

**Wine dataset** (n = 178, k = 3)

**House dataset** (n = 435, k = 2)