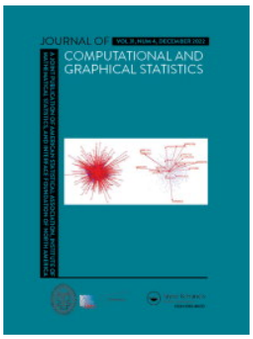

Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ucgs20

Search Algorithms and Loss Functions for Bayesian Clustering

David B. Dahl, Devin J. Johnson & Peter Müller

To cite this article: David B. Dahl, Devin J. Johnson & Peter Müller (2022) Search Algorithms and Loss Functions for Bayesian Clustering, Journal of Computational and Graphical Statistics, 31:4, 1189-1201, DOI: 10.1080/10618600.2022.2069779

To link to this article: https://doi.org/10.1080/10618600.2022.2069779

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2022, VOL. 31, NO. 4, 1189–1201
https://doi.org/10.1080/10618600.2022.2069779

Taylor & Francis
Taylor & Francis Group

Search Algorithms and Loss Functions for Bayesian Clustering

David B. Dahl[a], Devin J. Johnson[a], and Peter Müller[b]

[a]Department of Statistics, Brigham Young University, Provo, UT; [b]Department of Statistics and Data Sciences, University of Texas at Austin, Austin, TX

ABSTRACT

We propose a randomized greedy search algorithm to find a point estimate for a random partition based on a loss function and posterior Monte Carlo samples. Given the large size and awkward discrete nature of the search space, the minimization of the posterior expected loss is challenging. Our approach is a stochastic search based on a series of greedy optimizations performed in a random order and is embarrassingly parallel. We consider several loss functions, including Binder loss and variation of information. We note that criticisms of Binder loss are the result of using equal penalties of misclassification and we show an efficient means to compute Binder loss with potentially unequal penalties. Furthermore, we extend the original variation of information to allow for unequal penalties and show no increased computational costs. We provide a reference implementation of our algorithm. Using a variety of examples, we show that our method produces clustering estimates that better minimize the expected loss and are obtained faster than existing methods. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received May 2021
Accepted April 2022

KEYWORDS
Bayesian nonparametrics; Binder loss; Cluster estimation; Random partition models; Stochastic optimization; Variation of information

1. Introduction

In a typical Bayesian analysis, a great deal of computational effort is spent on "fitting the model," such as sampling from the posterior distribution or finding a tractable approximation to the posterior distribution. This, however, is only part of the inference problem. It is also necessary to summarize the posterior distribution in order to convey meaningful results. In many problems, parameters of interest often lie in a subset of $\mathbb{R}^n$ and, depending on the loss function, the Bayes rule might be the mean or median, which can easily be derived from posterior samples. Increasingly, parameters with a more complicated structure are being considered. It is often less clear how to summarize the posterior distribution of these more complicated structures. In this article, we focus on partitions and address the problem of point estimation from a partition distribution based on samples.

A partition $\rho = \{S_1, \ldots, S_p\}$ of integers $\{1, \ldots, n\}$ is a collection of subsets (i.e., clusters) such that the subsets are mutually exclusive, nonempty, and exhaustive. In model construction, partitions are often used to arrange data $y_1, \ldots, y_n$ such that data within a cluster are homogeneous. Items i and j are clustered together if $i \in S$ and $j \in S$ for some subset $S \in \rho$. A partition can alternatively be represented by cluster labels. We say that items i and j belong to the same cluster if and only if their cluster labels c_i and c_j are equal. We use the terms clustering and partition interchangeably. Likewise, the terms cluster and subset are used interchangeably. As a notational convention, we use cluster labels $1, \ldots, p$ for the elements of $\boldsymbol{c} = (c_1, \ldots, c_n)$ when a partition $\rho = \{S_1, \ldots, S_p\}$ has p subsets.

Under the Bayesian paradigm, the canonical approach to choosing an estimator is to introduce a loss function and then report the Bayes rule that minimizes the posterior expectation of the chosen loss function. We have:

$$\hat{\rho}^* = \underset{\hat{\rho}}{\operatorname{argmin}} \quad \mathbb{E}(L(\rho, \hat{\rho}) \mid \mathcal{D}) \qquad \text{or}$$
$$\hat{\mathbf{c}}^* = \underset{\hat{\mathbf{c}}}{\operatorname{argmin}} \quad \mathbb{E}(L(\mathbf{c}, \hat{\mathbf{c}}) \mid \mathcal{D}), \tag{1}$$

where $\mathcal{D}$ represents data and $L(\rho, \hat{\rho})$ and $L(\mathbf{c}, \hat{\mathbf{c}})$ are the same loss function represented in the partition and cluster label notations, respectively. Without loss of generality, we assume that if the estimator ($\hat{\rho}$ or $\hat{\mathbf{c}}$) is equal to the true parameter (ρ or $\boldsymbol{c}$), then the loss function evaluates to zero. Otherwise, the loss is some positive number representing the economic cost or regret associated with the decision ($\hat{\rho}$ or $\hat{\mathbf{c}}$) in light of the truth (ρ or $\boldsymbol{c}$).

Except in a situation with trivially small sample size n, the posterior expectation in (1) must be approximated, usually using posterior samples:

$$\mathbb{E}(L(\rho, \hat{\rho}) \mid \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} L(\rho^{(m)}, \hat{\rho}) \qquad \text{or}$$
$$\mathbb{E}(L(\mathbf{c}, \hat{\mathbf{c}}) \mid \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} L(\boldsymbol{c}^{(m)}, \hat{\mathbf{c}}), \tag{2}$$

where $\rho^{(1)}, \ldots, \rho^{(M)}$ or $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(M)}$ are M samples from a posterior distribution $p(\rho \mid \mathcal{D})$ or $p(\boldsymbol{c} \mid \mathcal{D})$. These are often obtained from several Markov chain Monte Carlo

CONTACT David B. Dahl dahl@stat.byu.edu Department of Statistics, Brigham Young University, Provo, UT.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JCGS.

(MCMC) chains and may require considerable effort. Here we merely assume these are available, and our task is to use these samples to obtain an estimate that summarizes the partition distribution.

In Section 2, we review the most common partition loss functions and related criteria, including Binder (1978) loss and the variation of information (Meilă 2007; Vinh, Epps, and Bailey 2010; Wade and Ghahramani 2018). Wade and Ghahramani (2018) and Rastelli and Friel (2018) conclude that Binder loss *overestimates* the number of clusters, but we note that the applied literature has almost exclusively used equal costs of misclassification. We offer an efficient technique in Section 3 to compute Binder loss with unequal costs of misclassification and show in Section 6 that this addresses the concerns about Binder loss finding too many clusters. Further in Section 3, we introduce a novel extension to variation of information (VI) that maintains theoretical properties of this original loss function, yet allows unequal costs of misclassification (analogous to the flexibility in Binder loss when using our efficient technique) and addresses the issue that VI may *underestimate* the number of clusters.

Selecting a loss function and generating posterior samples allows us to compute a Monte Carlo estimate of the posterior expected loss, but a far more challenging practical problem is searching the vast space of partitions for the minimizer of the Monte Carlo estimate of the posterior expected loss. We review the literature of existing search algorithms in Section 4, and in Section 5 present SALSO, a new search algorithm for any partition loss function. We show in Section 6 that SALSO is substantially faster and leads to demonstrably better estimates than existing algorithms, yet has tractable complexity such that it readily scales in the sample size n. The new and existing loss functions and the novel search algorithm are implemented in our `salso` package on the Comprehensive R Archive Network (CRAN).

2. Existing Partition Loss Functions and Other Criteria

Here we describe existing loss functions to estimate a random partition, as well as related criteria. Perhaps the simplest loss function is the 0–1 loss function, defined as $L_{0\text{-}1}(\rho, \hat{\rho}) = \mathbb{I}\{\rho = \hat{\rho}\}$ or $L_{0\text{-}1}(\mathbf{c}, \hat{\mathbf{c}}) = \mathbb{I}\{\mathbf{c} = \hat{\mathbf{c}}\}$, where $\mathbb{I}\{A\}$ is the indicator function equaling 1 if A is true and 0 otherwise. The 0–1 loss function yields the maximum a posteriori (MAP) partition, the mode of the posterior partition distribution. As in many other contexts in Bayesian analysis, the mode may not well represent the "center" of the partition distribution and the loss functions presented below are generally preferred over the 0–1 loss function. A partition loss function is computed from a contingency table as shown in Table 1, that is, a cross-tabulation of counts among all pairs of subsets from two partitions $\rho = \{S_1, \ldots, S_p\}$ and $\hat{\rho} = \{E_1, \ldots, E_q\}$. Conceptually, we think of ρ as the true partition and $\hat{\rho}$ as being its estimate. At times it is convenient to use the equivalent representation with cluster labels, where we use $\boldsymbol{c} = (c_1, \ldots, c_n)$ to denote the population clustering and $\hat{\boldsymbol{c}} = (\hat{c}_1, \ldots, \hat{c}_n)$ to be its estimate.

2.1. Binder Loss and Related Criteria

Binder (1978) loss has historically been the most widely used loss function to estimate a random partition. He suggested the following:

$$L_{\text{Binder}}(\mathbf{c}, \hat{\mathbf{c}}) = \sum_{i<j} \big(a \cdot \mathbb{I}\{c_i = c_j\}\mathbb{I}\{\hat{c}_i \neq \hat{c}_j\} + b \cdot \mathbb{I}\{c_i \neq c_j\}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\big), \tag{3}$$

where $a > 0$ and $b > 0$ give the unit costs for pairwise misclassification. Specifically, a represents the cost of failing to cluster together two items which should in fact be clustered together, whereas b represents the cost of clustering two items which should in fact be separate. Lau and Green (2007) noted that minimizing the posterior expectation of Binder loss is equivalent to maximizing $f(\hat{\mathbf{c}}) = \sum_{i<j} \mathbb{I}\{\hat{c}_i = \hat{c}_j\}(\pi_{ij} - b/(a+b))$, where the posterior similarity matrix $\boldsymbol{\pi}$ is an n-by-n matrix with elements $\pi_{ij} = \Pr(c_i = c_j \mid \mathcal{D})$. Its Monte Carlo estimate $\hat{\boldsymbol{\pi}}$ is obtained using $\hat{\pi}_{ij} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{I}\{c_i^{(m)} = c_j^{(m)}\}$, which can be computed before optimization. The task then becomes to maximize $f(\hat{\mathbf{c}}) = \sum_{i<j} \mathbb{I}\{\hat{c}_i = \hat{c}_j\}\left(\hat{\pi}_{ij} - b/(a+b)\right)$.

Without a reference to loss functions, Dahl (2006) suggested the "least squares clustering" criterion which seeks the clustering that minimizes $f(\hat{\mathbf{c}}) = \sum_{i=1}^{n}\sum_{j=1}^{n}(A(\hat{\mathbf{c}})_{ij} - \hat{\pi}_{ij})^2$, where $A(\hat{\mathbf{c}})$ is an n-by-n adjacency matrix whose ijth element is $\mathbb{I}\{\hat{c}_i = \hat{c}_j\}$. Dahl and Newton (2007) noted that minimizing this criterion is equivalent to minimizing the Monte Carlo estimate of the posterior expectation of Binder loss when $a = b$.

Wade and Ghahramani (2018) introduced an "n-invariant version" of Binder loss, which is interpretable across sample size n, under the assumption that $a = b = 1$:

$$L_{\text{nBinder}}(\rho, \hat{\rho}) = \frac{2}{n^2} L_{\text{Binder}}(\rho, \hat{\rho}) = \sum_{S\in\rho}\left(\frac{|S|}{n}\right)^2 + \sum_{E\in\hat{\rho}}\left(\frac{|E|}{n}\right)^2 - 2\sum_{S\in\rho}\sum_{E\in\hat{\rho}}\left(\frac{|S\cap E|}{n}\right)^2. \tag{4}$$

Table 1. Contingency table of counts used to compute various loss functions when estimating a population partition $\rho = \{S_1, \ldots, S_p\}$ with an estimated partition $\hat{\rho} = \{E_1, \ldots, E_q\}$.

		Estimated partition $\hat{\rho}$				
		E_1	E_2	$\cdots$	E_q	
True partition ρ	S_1	$n_{11} = \lvert S_1 \cap E_1\rvert$	$n_{12} = \lvert S_1 \cap E_2\rvert$	$\cdots$	$n_{1q} = \lvert S_1 \cap E_q\rvert$	$n_{1\cdot} = \lvert S_1\rvert$
	S_2	$n_{21} = \lvert S_2 \cap E_1\rvert$	$n_{22} = \lvert S_2 \cap E_2\rvert$	$\cdots$	$n_{1q} = \lvert S_2 \cap E_q\rvert$	$n_{2\cdot} = \lvert S_2\rvert$
	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
	S_p	$n_{p1} = \lvert S_p \cap E_1\rvert$	$n_{p2} = \lvert S_p \cap E_2\rvert$	$\cdots$	$n_{pq} = \lvert S_p \cap E_q\rvert$	$n_{p\cdot} = \lvert S_p\rvert$
		$n_{\cdot 1} = \lvert E_1\rvert$	$n_{\cdot 2} = \lvert E_2\rvert$	$\cdots$	$n_{\cdot q} = \lvert E_q\rvert$	n

Although Binder loss was first introduced with general costs a and b of pairwise misclassification, to our knowledge, every software implementation forces $a = b$ and applications seem to invariably assume that $a = b$. Whereas the n-invariant Binder loss of Wade and Ghahramani (2018) assumes $a = b = 1$, we provide in Section 3 an n-invariant version of Binder loss that restores the flexibility of potentially different costs a and b in (3) and is computationally no harder to evaluate. We demonstrate the advantage of this flexibility in Section 6, noting that the often criticized property of Binder loss overestimating the number of clusters is merely a result of assuming $a = b$, which is easily remedied by using $a > b$.

Rand (1971) introduced a measure of similarity between two partitions, which can be expressed in terms of Binder loss with $a = b = 1$ as $\text{RI}(\rho, \hat{\rho}) = 1 - L_{\text{Binder}}(\rho, \hat{\rho})/\binom{n}{2}$. Maximizing the posterior expectation of the Rand index (RI) is equivalent to minimizing the posterior expectation of Binder loss with $a = b = 1$. Hubert and Arabie (1985) proposed the adjusted Rand index which accounts for chance agreement, yielding:

$$\text{AR}(\rho, \hat{\rho}) = \frac{\sum_{S\in\rho}\sum_{E\in\hat{\rho}}\binom{|S\cap E|}{2} - \sum_{S\in\rho}\binom{|S|}{2}\sum_{E\in\hat{\rho}}\binom{|E|}{2}/\binom{n}{2}}{\frac{1}{2}\left[\sum_{S\in\rho}\binom{|S|}{2} + \sum_{E\in\hat{\rho}}\binom{|E|}{2}\right] - \sum_{S\in\rho}\binom{|S|}{2}\sum_{E\in\hat{\rho}}\binom{|E|}{2}/\binom{n}{2}}. \tag{5}$$

As large values of the adjusted Rand index indicate more similarity between clusterings, Fritsch and Ickstadt (2009) entertained the idea of seeking the clustering that maximizes the posterior expectation of the adjusted Rand index. However, they found it computationally expedient to instead maximize an approximation of that expectation:

$$f(\hat{\mathbf{c}}) = \frac{\sum_{i<j}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\hat{\pi}_{ij} - \sum_{i<j}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\sum_{i<j}\hat{\pi}_{ij}/\binom{n}{2}}{\frac{1}{2}\left[\sum_{i<j}\mathbb{I}\{\hat{c}_i = \hat{c}_j\} + \sum_{i<j}\hat{\pi}_{ij}\right] - \sum_{i<j}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\sum_{i<j}\hat{\pi}_{ij}/\binom{n}{2}}. \tag{6}$$

Using our computational techniques that we detail in Section 5, we find that resorting to the approximation is not necessary and, for large n, may be detrimental to computations. Instead, we suggest maximizing the posterior expectation of the adjusted Rand index, or equivalently minimizing the posterior expectation of "one minus the adjusted Rand index" (omARI) loss, that is, $L_{\text{omARI}}(\rho, \hat{\rho}) = 1 - AR(\rho, \hat{\rho})$.

2.2. Variation of Information and Other Information-Based Losses

An alternative, more recently introduced class of partition loss functions based on information theory has been proposed and studied by Meilă (2007), Vinh, Epps, and Bailey (2010), Wade and Ghahramani (2018), and Rastelli and Friel (2018). These are calculated using the totals from the contingency table (see Table 1) and are functions of individual entropies $H(\rho)$ and $H(\hat{\rho})$, joint entropy $H(\rho, \hat{\rho})$, and mutual information $I(\rho, \hat{\rho})$, as defined below:

$$\begin{aligned} H(\rho) &= -\sum_{S\in\rho}\frac{|S|}{n}\log_2\left(\frac{|S|}{n}\right) \\ H(\hat{\rho}) &= -\sum_{E\in\hat{\rho}}\frac{|E|}{n}\log_2\left(\frac{|E|}{n}\right) \\ H(\rho, \hat{\rho}) &= -\sum_{S\in\rho}\sum_{E\in\hat{\rho}}\frac{|S\cap E|}{n}\log_2\left(\frac{|S\cap E|}{n}\right) \\ I(\rho, \hat{\rho}) &= H(\rho) + H(\hat{\rho}) - H(\rho, \hat{\rho}). \end{aligned} \tag{7}$$

Note the use of the binary logarithm $\log_2(\cdot)$. The conditional entropy $H(\rho \mid \hat{\rho})$ can be written in several forms, including $H(\rho \mid \hat{\rho}) = H(\rho, \hat{\rho}) - H(\hat{\rho}) = H(\rho) - I(\rho, \hat{\rho})$.

The variation of information (VI) was introduced by Meilă (2007) as a measure of distance between two partitions. Wade and Ghahramani (2018) were the first to consider using the VI as a loss function to estimate a random partition. The VI is expressed as

$$\begin{aligned} L_{VI}(\rho, \hat{\rho}) &= H(\rho) + H(\hat{\rho}) - 2I(\rho, \hat{\rho}) \\ &= -H(\rho) - H(\hat{\rho}) + 2H(\rho, \hat{\rho}) \\ &= \sum_{S\in\rho}\frac{|S|}{n}\log_2\left(\frac{|S|}{n}\right) + \sum_{E\in\hat{\rho}}\frac{|E|}{n}\log_2\left(\frac{|E|}{n}\right) \\ &\quad - 2\sum_{S\in\rho}\sum_{E\in\hat{\rho}}\frac{|S\cap E|}{n}\log_2\left(\frac{|S\cap E|}{n}\right). \end{aligned} \tag{8}$$

Since $H(\rho)$ is constant when minimizing with respect to $\hat{\rho}$, Wade and Ghahramani (2018) note that minimizing the posterior expectation of VI loss is equivalent to

$$\begin{aligned} \hat{\mathbf{c}}^* &= \underset{\hat{\mathbf{c}}}{\text{argmin}} \quad \mathbb{E}(L_{VI}(\mathbf{c}, \hat{\mathbf{c}}) \mid \mathcal{D}) \\ &= \underset{\hat{\mathbf{c}}}{\text{argmin}} \sum_{i=1}^{n}\log_2\left(\sum_{j=1}^{n}\mathbb{I}(\hat{c}_j = \hat{c}_i)\right) \\ &\quad - 2\sum_{i=1}^{n}\mathbb{E}\left(\log_2\left(\sum_{j=1}^{n}\mathbb{I}(c_j = c_i)\mathbb{I}(\hat{c}_j = \hat{c}_i)\right) \mid \mathcal{D}\right). \end{aligned} \tag{9}$$

The expectation at the end of (9) can be approximated using Monte Carlo integration based on posterior samples $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(M)}$ obtained from MCMC. Wade and Ghahramani (2018) note, however, that evaluating this Monte Carlo approximation is $O(Hn^2)$, which may be costly when considering many candidate $\hat{\mathbf{c}}$'s in an optimization procedure. As such, they suggest applying Jensen's inequality to swap the logarithm and expectation, yielding a lower bound on (9). Specifically, Wade and Ghahramani (2018) suggest

$$\begin{aligned} \hat{\mathbf{c}}^* &= \underset{\hat{\mathbf{c}}}{\text{argmin}} \sum_{i=1}^{n}\log_2\left(\sum_{j=1}^{n}\mathbb{I}(\hat{c}_j = \hat{c}_i)\right) \\ &\quad - 2\sum_{i=1}^{n}\left(\log_2\left(\sum_{j=1}^{n}\hat{\pi}_{ij}\mathbb{I}(\hat{c}_j = \hat{c}_i)\right)\right), \end{aligned} \tag{10}$$

where $\hat{\pi}_{ij} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{I}\{c_i^{(m)} = c_j^{(m)}\}$ can be cached before optimization, making the computational complexity $O(n^2)$ for a

given $\hat{\boldsymbol{c}}$. In seeking to minimize the *lower bound* of the posterior expectation of VI, Wade and Ghahramani (2018) approach VI in an analogous manner to how Fritsch and Ickstadt (2009) approach omARI; they both seek to optimize an *approximation* to the expectation of the target quantity. The effect of applying Jensen's inequality has not been formally investigated, although our experience is that minimizing the two criteria almost never leads to the same estimate in typical applications. More to the point, although Meilă (2007) and Vinh, Epps, and Bailey (2010) find many desirable properties of VI, the extent to which they still hold when applying Jensen's inequality is not well understood.

In Section 3 we provide a generalization of the original variation of information with weights a and b to influence the tradeoffs between (i) failing to cluster two items which should be together and (ii) clustering two items which should be separate. These weights are analogous to those in Binder loss. We show that our generalization maintains a desirable property of the original VI, specifically that our generalization is still a quasi-metric. Moreover, whereas Wade and Ghahramani (2018) resort to an approximation of the posterior expectation of VI, we show in Section 5 a computationally-cheap way to evaluate the actual expectation, whether for the original VI or for our generalization with unequal a and b.

Vinh, Epps, and Bailey (2010) consider two dozen variants of the information-based distances and their respective properties. As they explain, the normalized variation of information (NVI) takes the VI as defined in (8), which has a range of $[0, \log_2 n]$, and scales it to have a range of $[0, 1]$. The NVI is defined as $L_{\text{NVI}}(\rho, \hat{\rho}) = 1 - \frac{I(\rho,\hat{\rho})}{H(\rho,\hat{\rho})}$. Instead of the NVI, however, Vinh, Epps, and Bailey (2010) advocate for the normalized information distance (NID) as a general purpose loss function with useful and important properties, defined as $L_{\text{NID}}(\rho, \hat{\rho}) = 1 - \frac{I(\rho,\hat{\rho})}{\max\{H(\rho),H(\hat{\rho})\}}$. The NID is normalized in that it takes values in $[0, 1]$; there is a corresponding unnormalized version taking values in $[0, \log_2 n]$, which we call the information distance (ID), defined as $L_{\text{ID}}(\rho, \hat{\rho}) = \max\{H(\rho), H(\hat{\rho})\} - I(\rho, \hat{\rho})$.

All of the loss functions described in this section can be used with our SALSO algorithm that we present in Section 5. We compare these loss functions in Section 6.

3. Generalizations of Binder and VI Loss

3.1. Generalization of Binder Loss

A major criticism of the Binder loss function in (3) is that it tends to produce estimates with too many clusters (Wade and Ghahramani 2018; Rastelli and Friel 2018), although these results were obtained with the implicit assumption that $a = b$. Recall that the Binder loss function has weights a and b which are positive constants, where a represents the cost of failing to cluster two items which should be together and b represents the cost of clustering two items which should be separate. With the notable exception being Lau and Green (2007), the original statement of Binder loss with potentially unequal weights has been lost in the literature. Available software seems to only implement Binder loss with equal weights.

Recall that Wade and Ghahramani (2018) provide (4), an expression for the Binder loss in terms of the contingency table when $a = b$. Expressing the Binder loss in terms of the contingency table facilitates fast computations in exploring the partition space, as we describe in Section 5. Extending the work of Wade and Ghahramani (2018), below we provide an expression for the n-invariant Binder loss in terms of general weights a and b:

Definition 1. For weights $a, b > 0$, the generalized Binder loss is

$$L_{\text{GBinder}}(\rho, \hat{\rho}) = a \sum_{S\in\rho} \left(\frac{|S|}{n}\right)^2 + b \sum_{E\in\hat{\rho}} \left(\frac{|E|}{n}\right)^2 - (a+b) \sum_{S\in\rho} \sum_{E\in\hat{\rho}} \left(\frac{|S\cap E|}{n}\right)^2. \tag{11}$$

We refer to (11) as the generalized Binder loss to contrast it with (4) and to differentiate it from the typical practice in the literature. A proof of the equivalence between the Binder loss as it was originally stated in (3) and our expression in (11) is found in the Appendix.

3.2. Generalization of the Variation of Information

While Binder loss may yield too many clusters, it has been suggested that the variation of information may produce too few clusters in some cases. In order to counteract this, we propose the generalized variation of information loss (GVI):

Definition 2. For weights $a, b > 0$, the generalized variation of information (GVI) is

$$\begin{aligned} L_{\text{GVI}}(\rho, \hat{\rho}) &= bH(\rho) + aH(\hat{\rho}) - (a+b)I(\rho, \hat{\rho}) \\ &= -aH(\rho) - bH(\hat{\rho}) + (a+b)H(\rho, \hat{\rho}) \\ &= a \sum_{S\in\rho} \frac{|S|}{n} \log_2\left(\frac{|S|}{n}\right) + b \sum_{E\in\hat{\rho}} \frac{|E|}{n} \log_2\left(\frac{|E|}{n}\right) \\ &\quad - (a+b) \sum_{S\in\rho} \sum_{E\in\hat{\rho}} \frac{|S\cap E|}{n} \log_2\left(\frac{|S\cap E|}{n}\right). \end{aligned} \tag{12}$$

Note the similarity between this proposed measure and Binder loss as defined in (11).

3.3. Quasi-Metric Properties

Let $\mathcal{X}_n$ be the set of all possible partitions of n items and let $L : \mathcal{X}_n \times \mathcal{X}_n \to [0, \infty)$ be a partition loss function. For all $\rho_1, \rho_2, \rho_3 \in \mathcal{X}_n$, L is a metric (i.e., a distance function) if it satisfies the following three properties: (i) $L(\rho_1, \rho_2) = 0 \Leftrightarrow \rho_1 = \rho_2$ ["identity of indiscernibles"], (ii) $L(\rho_1, \rho_2) = L(\rho_2, \rho_1)$ ["symmetry"], and (iii) $L(\rho_1, \rho_2) + L(\rho_2, \rho_3) \geq L(\rho_1, \rho_3)$ ["the triangle inequality"]. Meilă (2007) showed that a partition loss function which satisfies the metric properties aligns with human intuition. It is instinctual, for example, that the distance between two partitions can only be zero if they are, in fact, the same partition (as provided by the identity of indiscernibles). The triangle inequality is particularly appealing, ensuring that, if two

partitions are both close to a third partition, they must not be too far apart from each other. Meilă (2007) proved that the variation of information is a metric and showed that it behaves in "an intuitive and desirable way on clusterings that are related by elementary operations like splitting/merging of clusters."

Wade and Ghahramani (2018) noted that Binder loss (under the implicit assumption $a = b$) is a metric. The properties of the generalized Binder loss ($a \neq b$) are not discussed in the literature, but a trivial proof-by-contradiction reveals that it violates the symmetry property and is, therefore, not a metric. This is, however, intuitive; when $a \neq b$ and thinking of $\mathcal{X}_n$ as a river, the effort to go downstream (e.g., $L(\rho_1, \rho_2)$) is not the same as going upstream (e.g., $L(\rho_2, \rho_1)$). Setting $a \neq b$ induces a "current" in the river by penalizing misclassification errors unequally. When $a = b$ there is no current in the river so symmetry does hold. A quasi-metric is a function with the identity of indiscernibles and the triangle inequality (but not symmetry), thus, preserving all the intuitive properties of a metric except the one property that is explicitly rejected when choosing $a \neq b$.

Theorem 1. The generalized Binder loss is a quasi-metric.

Likewise, when $a \neq b$, we explicitly reject symmetry, but our generalized variation of information (GVI) preserves the other two desirable properties.

Theorem 2. The generalized variation of information is a quasi-metric.

We provide proofs for both theorems in the Appendix.

3.4. Controlling the Number of Clusters

The criticism of the Binder loss function yielding too many clusters may well be an artifact that practitioners are using equal costs of misclassification. Intuitively, one would expect that using $a > b$ would address the overestimation of the number of clusters, because this defines a greater cost to incorrectly splitting clusters than there is to incorrectly clustering more items together. Indeed, taken to the extremes, for fixed $b > 0$, we prove that $a \to 0$ yields a clustering estimate with every item in a singleton cluster. Likewise, for fixed $a > 0$, we prove that $b \to 0$ yields a clustering estimate with every item in the same cluster.

Theorem 3. For fixed $b > 0$ and $\hat{\rho} = \{\{1\}, \{2\}, \ldots, \{n\}\}$, $\lim_{a \to 0} L_{\text{GBinder}}(\rho, \hat{\rho}) = 0$. Conversely, for fixed $a > 0$ and $\hat{\rho} = \{\{1, 2, \ldots, n\}\}$, $\lim_{b \to 0} L_{\text{GBinder}}(\rho, \hat{\rho}) = 0$.

The proof is trivial when applying the limit to the right hand side of (11) and noting that $\hat{\rho} = \{E_1\} = \{\{1, 2, \ldots, n\}\}$ for the $a > 0$ case, and $\hat{\rho} = \{E_1, \ldots, E_n\} = \{\{1\}, \{2\}, \ldots, \{n\}\}$ for the $b > 0$ case. Thus, the costs a and b control the number of clusters. Without loss of generality, suppose that $a \in [0, 2]$ and $b = 2 - a$. Recall that the generalized Binder loss is a quasi-metric which implies that, for $a, b > 0$, $L_{\text{GBinder}}(\rho, \hat{\rho}) > 0$ when $\rho \neq \hat{\rho}$. Hence, for a nondegenerate partition distribution, the expected loss will be greater than zero when $a, b > 0$. Thus, as a goes from 0 to 2, the expected value of the generalized Binder loss goes from 0 to values greater than 0 and then back down to 0. Therefore, the minimizer of the posterior expected loss will range from $\{\{1\}, \{2\}, \ldots, \{n\}\}$ to $\{\{1, 2, \ldots, n\}\}$ as a ranges from 0 to 2.

We demonstrate how the minimizer of the posterior expected loss is controlled by a with the following example. Consider data $\boldsymbol{x} = (x_1, \ldots, x_9) = (-0.81, -0.80, -0.43, 0.18, 0.22, 0.28, 0.30, 0.43, 0.54)$. Suppose $x_i \mid \theta_i \sim \text{N}(\theta_i, 0.2^2)$, $\theta_i \mid G \sim G$, and G has a Dirichlet process prior with concentration parameter 1 and a standard normal centering distribution. Marginalizing over G and the model parameters $\theta_1, \ldots, \theta_9$ yields a posterior distribution for a partition $p(\rho \mid \boldsymbol{x})$ which can be obtained exactly by enumerating all 21,147 partitions of $n = 9$ items. Again through enumeration, the minimizer of the posterior expected loss, that is, (1) can be obtained for the generalized Binder loss in (11) as a function of a (with $b = 2 - a$). This is summarized in Table 2. Note that small values of a yield partition estimates with many clusters, whereas large values of a yield partition estimates with many clusters. Likewise, we show empirically in Section 6 that the generalized Binder loss with $a > b$ (e.g., $a = 4/3, b = 2/3$) yields estimates with fewer clusters than when $a = b$. We believe that some of the rise in popularity of the variation of information loss, which tends to produce fewer clusters, is due to the use of Binder loss with equal weights. We hope that noting the utility of unequal weights (together with the fast software implementation in `salso`) has the potential to reignite interest in the Binder loss function.

Since the GVI is a quasi-metric and because it has the same limit properties as the generalized Binder loss in Theorem 3, the GVI also allows control over the number of clusters in estimates. While it is not possible to write the GVI in terms of the sum of per-unit costs—as can be done for the Binder loss in (3)—it is nonetheless the case that GVI behaves similarly to the Binder loss in that, for fixed $b > 0$, $a \to 0$ yields a clustering estimate with every item in a singleton cluster and, for fixed $a > 0, b \to 0$ yields a clustering estimate with every item in the same cluster. This is summarized in Table 3 using the previous example. Note that small values of a yield partition estimates with many clusters, whereas large values of a yield partition estimates with many clusters. In contrast to generalized Binder loss, however, changes to the partition estimate can be dramatic around $a = 0.532$. We show in Section 6 that GVI with $a = 2/3, b = 4/3$ yields more clusters than when $a = b$.

Table 2. Minimizer of the posterior expectation of Binder loss as a function of a (and $b = 2 - a$).

a in Binder loss		Expected	Number	Minimizer of the
Range	Value	loss	of clusters	posterior expected loss
(0.000, 0.679)	0.339	0.147	9	{{1},{2},{3},{4},{5},{6},{7},{8},{9}}
(0.679, 0.696)	0.687	0.297	8	{{1},{2},{3},{4},{5},{6},{7},{8,9}}
(0.696, 0.698)	0.697	0.301	7	{{1},{2},{3},{4},{5},{6,7},{8,9}}
(0.698, 0.700)	0.699	0.301	7	{{1},{2},{3},{4},{5},{6},{7,8,9}}
(0.700, 0.712)	0.706	0.303	6	{{1},{2},{3},{4},{5},{6,7,8,9}}
(0.712, 0.721)	0.716	0.306	5	{{1},{2},{3},{4},{5,6,7,8,9}}
(0.721, 0.809)	0.765	0.309	4	{{1},{2},{3},{4,5,6,7,8,9}}
(0.810, 0.973)	0.891	0.315	3	{{1,2},{3},{4,5,6,7,8,9}}
(0.973, 1.167)	1.070	0.317	2	{{1,2,3},{4,5,6,7,8,9}}
(1.167, 1.355)	1.261	0.305	2	{{1,2},{3,4,5,6,7,8,9}}
(1.355, 2.000)	1.678	0.147	1	{{1,2,3,4,5,6,7,8,9}}

Table 3. Minimizer of the posterior expectation of generalized variation of information (GVI) loss in (12) as a function of a (and $b = 2 - a$).

a in GVI Loss		Expected	Number	Minimizer of the
Range	Value	loss	of clusters	posterior expected loss
(0.000, 0.532)	0.266	0.543	9	{{1},{2},{3},{4},{5},{6},{7},{8},{9}}
(0.532, 0.809)	0.671	1.131	4	{{1},{2},{3},{4,5,6,7,8,9}}
(0.810, 0.846)	0.828	1.177	3	{{1,2},{3},{4,5,6,7,8,9}}
(0.847, 0.901)	0.874	1.173	2	{{1,2,3},{4,5,6,7,8,9}}
(0.901, 0.997)	0.949	1.149	2	{{1,2},{3,4,5,6,7,8,9}}
(0.997, 2.000)	1.499	0.566	1	{{1,2,3,4,5,6,7,8,9}}

4. Existing Algorithms for Partition Summarization

Computing a Monte Carlo estimate of the posterior expected loss requires both a loss function—whether an existing one from Section 2 or one of our extensions from Section 3—and samples from the posterior partition distribution. A far more challenging practical problem, however, is searching the vast space of partitions for the minimizer of the chosen criterion, that is, searching for the minimizer of the Monte Carlo estimate of the expectation of the chosen loss function. We review the literature of existing search algorithms in this section and then, in Section 5, propose SALSO, a new search algorithm for any loss function that is substantially faster and leads to demonstrably better estimates than existing algorithms, yet has tractable complexity such that it readily scales in n items.

Obviously an exhaustive search of all possible clusterings of n items, evaluating the chosen criterion for each clustering, will yield the absolute minimizer. Exhaustive enumeration is only feasible for very small n because the nth Bell number $B(n)$, that is, the number of possible clusterings of n items, grows exponentially. For example, $B(50)$ is more than 10^{47}. Dahl (2006) suggested the draws method which simply selects, among all those in the MCMC output, the clustering that minimizes the chosen criterion. This method is practical, fast, and applicable to any loss function, but the clustering estimate is obviously limited to the clusterings visited by the Markov chain. We will now review existing methods that can produce clustering estimates beyond those visited by the Markov chain.

4.1. Medvedovic and Sivaganesan (2002)—Hierarchical Clustering

Medvedovic and Sivaganesan (2002) proposed using hierarchical agglomerative clustering based on MCMC output. Medvedovic and Sivaganesan (2002) proposed using as the distance matrix one minus the estimated posterior similarity matrix $\hat{\boldsymbol{\pi}}$ defined in Section 2.1. Medvedovic and Sivaganesan (2002) used complete linkage, but other linkage methods could be used. This method has been viewed as ad hoc (see, Dahl 2006; Fritsch and Ickstadt 2009) because it is not based on a loss function and it builds a full tree of possible clusterings, leaving the problem of where to cut the tree. This method does, however, quickly give a reasonable answer. Fritsch and Ickstadt (2009) took this idea further where they cut the tree (i.e., select the clustering among those implied by the tree) to minimize the Monte Carlo estimate of the posterior expected loss, yielding a clearly defined implementation based on a loss function. Nevertheless, Rastelli and Friel (2018) found in a simulation study comparing several procedures that "it is clear that the (Medvedovic and Sivaganesan (2002) method with cuts from Fritsch and Ickstadt (2009)) performs quite poorly" in terms of the quality of the clustering estimate produced.

4.2. Lau and Green (2007)—Binary Integer Programming

Lau and Green (2007) proposed two search procedures for an optimal clustering. The first involves formulating and solving a binary integer programming problem, but is impractical as n increases. Recognizing the intractability, Lau and Green (2007) propose another procedure, a heuristic item-swapping algorithm as a fast approximation of their first procedure. The second procedure, however, still suffers from scalability problems. Fritsch and Ickstadt (2009) were able to apply the algorithm for $n = 200$ after several hours of computation, but found "it was not possible to apply the algorithm to all 400 observations, as the optimization problems required at each iteration got too large to be handled by the software." In addition to this lack of scalability and slow computation, the methods proposed by Lau and Green (2007) are only detailed in terms of Binder loss and the estimated posterior similarity matrix. It is not clear how broadly the algorithm could be extended since, for example, the variation of information cannot be expressed as a function of the posterior similarity matrix.

4.3. Wade and Ghahramani (2018)—Greedy Algorithm

The method of Wade and Ghahramani (2018) is one of three "greedy algorithms" described in this article. A greedy algorithm is a procedure that takes small, locally-optimal updates at each step of the algorithm along the way to finding its final solution. The method of Rastelli and Friel (2018) and our proposed SALSO algorithm are also greedy algorithms.

The greedy search algorithm of Wade and Ghahramani (2018) takes locally-optimal moves in a neighborhood of partitions defined in terms of the chosen loss function and the Hasse diagram, a lattice in which the nodes are all possible partitions and edges are those partitions that are one change away from each other. See Wade and Ghahramani (2018) for details and examples of the Hasse diagram. Their algorithm is implemented for Binder loss and the *lower bound* of VI loss in the `mcclust.ext` package for R, available on Wade's website. It is worth noting that their implementation relies on the estimated posterior similarity matrix, and thus cannot be easily applied to loss functions such as VI, NID, and NVI.

One downside to this method is its dependence on the initial partition, especially since the algorithm can get stuck in a local minimum as it traverses the Hasse diagram. Even though it scales better in n than the method proposed by Lau and Green (2007), Rastelli and Friel (2018) conclude from their simulation study that "the method of Wade and Ghahramani does not scale particularly well."

4.4. Rastelli and Friel (2018)—Greedy Algorithm

The greedy search algorithm of Rastelli and Friel (2018) starts at a randomly selected partition with many small clusters and

iteratively reassigns one item at a time to existing clusters or a new singleton cluster, where reassignment decisions are made to minimize the Monte Carlo estimate of the expected loss. A scan is completed once each item, in a random order, has been considered for reassignment and the algorithm stops once a scan yields no change.

The search algorithm proposed by Rastelli and Friel (2018) is stochastic in nature, as the starting partition is assigned randomly and the one-at-a-time optimizations are done in a random order each time. Although Rastelli and Friel (2018) recommend against multiple runs of their algorithm, we show in Section 6 that multiple runs can greatly increase the chance of a better answer and that, for difficult problems, it is very unlikely that the optimal partition will be obtained on only one run. The algorithm is implemented by the authors in the `GreedyEPL` package on CRAN for the following losses: Binder, VI, NVI, and NID. We note that the user can specify the maximum number of clusters K_{up}, but the software will actually use the maximum of K_{up} and the largest number of clusters found in any iteration of the MCMC output. Thus, while the method as described in the paper can control the maximum number of clusters, in practice their software can be unwieldy in this regard.

5. SALSO Algorithm

In this section, we describe our SALSO algorithm, investigate its complexity, and discuss computational shortcuts such that (2) need not be fully evaluated for each partition that is considered in the SALSO algorithm.

5.1. Description of the SALSO Algorithm

The SALSO algorithm, like most algorithms discussed in Section 4, tries to perform the optimization in (1) using Monte Carlo estimates in (2). Like the Rastelli and Friel (2018) method (R&F), the SALSO algorithm is a greedy, stochastic search and—with particular choices for its parameters—can mimic their behavior. There are four phases of the SALSO algorithm: initialization, sweetening, zealous updates, and recording. The algorithm is embarrassingly parallel and we advocate for performing multiple runs. The algorithm is implemented for several loss functions in the `salso` package available on CRAN.

The SALSO algorithm provides two methods to initialize a partition. The first method is sequential allocation, in which each item is allocated one at a time—in the order determined by a permutation sampled uniformly among all possible permutations—to an existing cluster or a new cluster, based on the allocation which minimizes the Monte Carlo estimate of the posterior expected loss, ignoring any yet-to-be allocated items. For example, the first randomly-selected item is placed in a cluster by itself. The second randomly-selected item is placed in either the cluster with the first item, or in a new cluster by itself, depending on which minimizes the posterior expected loss, computed as if there were only those two items in the system. The process continues until all the items are allocated, each time allocating to one of the existing clusters or to a new cluster, depending on which allocation minimizes the posterior expected loss with respect to the allocated items. The algorithm, however, does not consider allocation to a new cluster if doing so would lead to a partition having more than the desired maximum number of clusters, denoted k_d. In the second initialization method, cluster labels are obtained by uniformly sampling the labels $1, \ldots, k_d$. The initialization method for a particular run is randomly chosen and the user can specify the probability of sequential allocation, denoted p_{SA}, with 0.5 being its default value.

Once the partition is initialized, the next step is the sweetening phase, in which random one-at-a-time reallocations of individual items are performed in a random order. This is essentially the same idea as in the sequential allocation method, except now every item is allocated and each item—one at a time and in the order determined by a permutation sampled uniformly among all possible permutations—is removed from its cluster and reallocated to existing clusters or a new cluster, according to the choice that minimizes the Monte Carlo estimate of the posterior expected loss. This process is repeated until there is no change after a complete pass through all n items.

In the third phase, "zealous" updates attempt to break out of a local minimum. As these zealous updates can be computationally expensive, we suggest setting an upper bound on the number of such updates, denoted $n_{\text{maxZealous}}$. (The default in our software is 10.) For a random ordering of up to $n_{\text{maxZealous}}$ clusters, the current state is recorded and the cluster is destroyed by removing all of its items. These deallocated items are then sequentially reallocated—one at a time and in a random order—conditioning on the already allocated items, in the same way as sequential allocation in the initialization phase. Once everything is reallocated, the Monte Carlo estimate of the posterior expected loss of the current partition is compared to what it was before destroying the cluster. If no improvement was found by this zealous update, it is abandoned and the state reverts to the previous state.

Finally, in the fourth phase, the Monte Carlo estimate of the posterior expected loss is recorded for the current state. This algorithm is "embarrassingly parallel"—since each run of the algorithm does not rely on any other run—such that n_{runs} runs of the algorithm can easily be conducted using all available CPU cores. Among all n_{runs} candidates, the partition with the smallest Monte Carlo estimate is then reported as the partition estimate. The entire SALSO algorithm is shown in pseudocode in Algorithm 1.

We note that our SALSO algorithm almost reduces to that of Rastelli and Friel (2018) when: (a). the probability of sequential allocation p_{SA} is set to 0, (ii). the number of zealous updates $n_{\text{maxZealous}}$ is set to 0, and (iii) the number of runs n_{runs} is set to 1. Even with those specific choices, however, significant practical differences remain between these two algorithms. First, the default (and recommended) value for the maximum number of clusters k_d in Rastelli and Friel (2018) is n, but this can worsen complexity, greatly slow down computations, increase RAM requirements, and lead to uninterpretable estimates. (See Sections 5.2 and 6.) Further, their implementation forces k_d to be no smaller than the maximum number of clusters observed among the posterior samples $\rho^{(1)}, \ldots, \rho^{(M)}$, which still may result in too many clusters for interpretable estimates. In our implementation, the user has full control over k_d and we set its default value to be the maximum number of clusters observed instead of n. Another practical difference is that R&F "hard-coded" epsilon

Algorithm 1 Pseudocode for the SALSO Algorithm. Let n_{runs} be the number of runs, p_{SA} be the probability of sequential allocation, k_d be the maximum number of clusters, and $n_{\text{maxZealous}}$ be the maximum number of zealous updates. Note that, when considering "all existing clusters and a new cluster" below, the algorithm does *not* consider a new cluster if that would lead to a state in which the number of clusters exceeds the maximum number of clusters k_d.

1: **for** $i = 1, \ldots, n_{\text{runs}}$ **do**
2: ▷ Initialization phase
3: **if** Uniform[0,1] $< p_{\text{SA}}$ **then** ▷ Do sequential initialization
4: Uniformly sample a permutation $(\sigma_1, \ldots, \sigma_n)$ of $\{1, \ldots, n\}$.
5: **for** $\sigma = \sigma_1, \ldots, \sigma_n$ **do**
6: **for** c being all existing clusters and a new cluster **do**
7: Calculate the posterior expected loss if σ were added to cluster c.
8: Allocate σ to the cluster that minimizes the posterior expected loss.
9: **else** ▷ Do random initialization
10: Sample a random partition with at most k_d clusters,
11: e.g., `sample(1:`k_d`, n, replace=TRUE)` in `R`.
12: **while** `TRUE` **do** ▷ Sweetening phase
13: Uniformly sample a permutation $(\sigma_1, ..., \sigma_n)$ of $\{1, ..., n\}$.
14: **for** $\sigma = \sigma_1, \ldots, \sigma_n$ **do**
15: Remove σ from its cluster.
16: **for** c being all existing clusters and a new cluster **do**
17: Calculate the posterior expected loss if σ were added to cluster c.
18: Allocate σ to the cluster that minimizes the posterior expected loss.
19: **if** partition is unchanged **then break**
20: **for** c' in a random ordering of up to $n_{\text{maxZealous}}$ clusters **do** ▷ Zealous updates phase
21: Record the current partition as $\hat{\rho}$.
22: Destroy cluster c' by removing (deallocating) all of its items.
23: **for** σ in a random ordering of deallocated items **do**
24: **for** c being all existing clusters and a new cluster **do**
25: Calculate the posterior expected loss if σ were added to cluster c.
26: Allocate σ to the cluster that minimizes the posterior expected loss.
27: **if** $\hat{\rho}$ has a smaller posterior expected loss than the current partition **then**
28: Revert the current partition to $\hat{\rho}$.
29: Record the current partition as $\hat{\rho}_i$ and note its posterior expected loss. ▷ Recording phase
30: **return** the $\hat{\rho}_i$ with the smallest posterior expected loss.

in their stopping rule in the sweetening phase, which does not account for differences in scales among various loss functions. Finally, we show in Section 6 that our implementation is faster, in part because of the computational shortcuts that we detail in Section 5.3.

5.2. Complexity Comparison

We now compare the complexity of the SALSO algorithm with those of Wade and Ghahramani (2018) and Rastelli and Friel (2018) in greater detail. The complexity of the SALSO algorithm is $\mathcal{O}(M \cdot k_d \cdot k_M \cdot n)$, where M is the number of MCMC samples, k_d is the maximum number of clusters desired by the user, k_M is the maximum number of clusters observed among the MCMC samples, and n is the number of items. We recommend setting k_d to a relatively small number for the sake of interpretation of the clustering estimate. It defaults to k_M, which is typically much smaller than n. So, the default complexity for SALSO is $\mathcal{O}(M \cdot k_M^2 \cdot n)$, but it will be less if the user specifies $k_d < k_M$. The complexity of the R&F algorithm is $\mathcal{O}(M \cdot k_d^2 \cdot n)$, and the implementation defaults to k_d being n and requires that k_d be at least k_M. So, in the default case, the R&F algorithm has complexity $\mathcal{O}(M \cdot n^3)$ and, in the best case, has complexity $\mathcal{O}(M \cdot k_M^2 \cdot n)$. Finally, the complexity of the Wade and Ghahramani (2018) algorithm is $\mathcal{O}(\ell \cdot n^2)$, where 2ℓ defines the number of partitions to consider at each iteration and defaults to n, meaning the default complexity is $\mathcal{O}(n^3)$.

5.3. Computational Speedups

Notice that on lines 6–8, 17–19, and 26–28 in Algorithm 1, the SALSO algorithm needs to allocate the current item, denoted σ, to one of the existing clusters or to a new cluster and that this choice is made such that the Monte Carlo estimate of the posterior expected loss is minimized. Let $\hat{\boldsymbol{c}}_1, \ldots, \hat{\boldsymbol{c}}_q, \hat{\boldsymbol{c}}_{q+1}$ denote the clusterings obtained by allocating the current item σ to the q existing clusters or to a new cluster. To make this allocation, then, it would seem that the SALSO algorithm must compute:

$$\mathbb{E}(L(\boldsymbol{c}, \hat{\boldsymbol{c}}_j \mid \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} L(\boldsymbol{c}^{(m)}, \hat{\boldsymbol{c}}_j), \quad \text{for } j = 1, \ldots, q, q+1, \tag{13}$$

where $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(M)}$ are M samples from a posterior distribution $p(\boldsymbol{c} \mid \mathcal{D})$ and then allocate the current item, σ, to the cluster among all $q+1$ explored that minimizes the Monte Carlo estimate of the posterior expected loss, that is, to cluster s where

$$\frac{1}{M}\sum_{m=1}^{M} L(\boldsymbol{c}^{(m)}, \hat{\boldsymbol{c}}_s) \leq \frac{1}{M}\sum_{m=1}^{M} L(\boldsymbol{c}^{(m)}, \hat{\boldsymbol{c}}_j), \quad \forall \; j \in \{1, \ldots, q, q+1\}. \tag{14}$$

We have recognized and implemented certain computational speedups that have allowed for the efficient repeated calculation of the Monte Carlo approximation of the posterior expected loss. This is possible when the loss is written in terms of counts, for example, as in (11) and (12). Both (i) allocating a new item to a cluster and (ii) moving an item from one cluster to another require updating just four counts in a contingency matrix (see Table 1). Because only four counts are updated, software can easily cache these contingency tables—one for each $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(M)}$—and make only the four required updates per MCMC sample after each change to the estimated partition. This allows for efficient incremental calculation and storage of the contingency matrices necessary for calculation of the loss functions.

For some loss functions, further shortcuts are possible. For the generalized Binder loss in (11), picking the $\hat{\boldsymbol{c}}_j$ with the smallest value in (13) is equivalent to choosing among:

$$bMn_{\cdot j} - (a+b)\sum_{m=1}^{M} n_{c^{(m)}j}, \quad \text{for } j = 1, \ldots, q, q+1, \tag{15}$$

where $n_{\cdot j}$ and n_{ij} are defined in the contingency table (Table 1) and $c^{(m)}$ is the cluster label of the current item σ in the mth posterior sample. Computing (15) is very fast given cached counts in a contingency table. We implement a similar shortcut for our generalized variation of information (GVI) in (12). That is, we choose among:

$$bMf(n_{\cdot j}) - (a+b)\sum_{m=1}^{M} f(n_{c^{(m)}j}), \quad \text{for } j = 1, \ldots, q, q+1, \tag{16}$$

where $f(n) = n\log_2(n) - (n-1)\log_2(n-1)$ is a function of an integer than can be cached rather than repeatedly computed.

6. Verifications

In this section we demonstrate the ability of the SALSO algorithm to correctly identify (in a small n problem) the global minimizer and to find (in large n problems) a partition whose Monte Carlo estimate of the posterior expectation is generally less than that of other search algorithms. All calculations were performed using R 4.1.2, `salso` 0.3.0, and `GreedyEPL` 1.2 on a computer with two AMD EPYC 7502 processors and 384G of RAM. Timing benchmarks did not exhaust the RAM and used no more than 20% of the available CPU threads on the otherwise-idle machine.

We first consider a small $n = 9$ problem, where all 21,147 possible partitions can be enumerated and the global minimizer of the Monte Carlo estimate of the posterior expected loss can be found through exhaustive search. Continuing the example in Section 3.4, we obtained 1000 sets of Monte Carlo samples of size 100 partitions from the distribution and applied the software of Rastelli and Friel (2018) and our SALSO algorithm with 1 and 16 runs. For SALSO, in addition to $a = b = 1$, we also considered a values from Tables 2 and 3. For Binder loss, the Rastelli and Friel (2018) software obtained the global minimizer in 62.5% of the 1000 sets, whereas SALSO with 1 and 16 runs did so in 78.5% and 96.6%, respectively. For the other values of a, the SALSO algorithm's success rate was at least 86.9%. For VI loss, the Rastelli and Friel (2018) software had a success rate of 96.5% and SALSOs rate was 96.7% and 97.0%. For other values of a in GVI, SALSOs success rate was at least 95.7%. These results suggest that SALSO tends to find the global minimizer with higher probability than the Rastelli and Friel (2018) algorithm, yet both methods can fail to find the global minimizer even in a small problem. This should serve as a caution against claiming that these methods will necessarily find the global minimum in much larger problems where the search space grows exponentially in size.

We now describe a study for large n in which the SALSO algorithm tends to find a partition whose Monte Carlo estimate of the expectation is less than that of other search algorithms. We focus on Binder and VI loss functions, although similar trends were found for omARI, NVI, ID, and NID loss functions. We consider using three sets of samples from posterior partition distributions. Each of these three sets provide multiple model fits for different data or model specifications. In our study, each procedure was replicated 10 times for each set of MCMC output for a particular model. We report results as averages across the replications. Our first set is labeled "PM10." Page, Quintana, and Dahl (2021) analyze averaged monthly PM 10 data from 60 stations in the European air quality database using 8 different models for each of 12 months of data. The PM10 set of posterior samples consists of a collection of $8 \times 12 = 96$ models, each providing 1000 samples for the clustering of 60 stations. Our second set is labeled "Gaskins" and comes from Gaskins, Fuentes, and la Cruz (2017), who perform a simulation study of many methods to summarize samples from posterior clustering distributions using two sets of posterior samples. Here we consider the more challenging set, which consists of 200 distinct model fits, each providing 2000 posterior samples, for the clustering of 200 items. Our third set is labeled "SIMCE," which again comes from Page, Quintana, and Dahl (2021). They consider eight models for 1072 schools at each of seven time periods, yielding a collection of $8 \times 7 = 56$ models clustering 1072 items with each model having 1000 posterior samples.

Using these three sets, we first compare our implementation of the SALSO algorithm to the algorithm of Rastelli and Friel (2018) as implemented in the `MinimiseEPL` function of their `GreedyEPL` package, which is archived on CRAN. Rastelli and Friel (2018) recommend using a single run of their stochastic algorithm so, for the sake of comparison, we also limit SALSO to a single run. For SALSO, we try at most 10 zealous updates, use a 50% probability of sequential allocation, and do not constrain the number of clusters. We use the default settings for the `MinimiseEPL` function in the `GreedyEPL` package. The results are shown in Table 4. The "A < B" column shows the proportion of times that the Monte Carlo estimate of the posterior expected loss for the estimate reported by the

Table 4. The quality and run time of: (A) SALSO algorithm and (B) the default Rastelli and Friel (2018) algorithm from their software.

		Quality			Run Time		
Dataset	Loss	A < B	B < A	Diff.	A	B	Ratio
PM10	Binder	**0.14**	0.00	0.14	**0.01**	0.02	0.54
	VI	**0.32**	0.00	0.31	**0.01**	0.09	0.11
Gaskins	Binder	**0.09**	0.02	0.08	**0.20**	0.47	0.42
	VI	**0.72**	0.01	0.71	**0.09**	1.95	0.05
SIMCE	Binder	**0.18**	0.07	0.11	12.30	**11.33**	1.09
	VI	**0.45**	0.01	0.44	**0.65**	19.99	0.03

NOTE: Here both methods are allowed to produce results with any number of clusters. **A**: SALSO (10, 0.5), 1 run, **B**: R&F 2018, 1 run.

SALSO algorithm was lower than that obtained by the R&F algorithm, while the "B < A" column shows the proportion of times that R&F is better than SALSO. To aid comparison, the difference between these two proportions is also noted. Note that the proportion of ties is 1 − ("A < B" + "B < A"). The mean run time for each method is also recorded, as well as the ratio of the times. From Table 4, it is clear that the SALSO algorithm is outperforming the algorithm proposed by Rastelli and Friel (2018) both in terms of quality of the answer and in terms of the run time. The SALSO algorithm yields an answer better than the R&F algorithm much more than the R&F algorithm yields an answer better than the SALSO algorithm. The SALSO algorithm also obtains an answer much more quickly on average.

The next part of the study is meant to show the ability of the SALSO algorithm and the chosen loss function to control the number of clusters. In the SALSO algorithm, the maximum number of clusters can easily be set, which has important implications for the interpretability of the resulting clustering and can also influence the RAM and CPU time needed for the optimization algorithm. The default in the `salso` package is to constrain the optimization by the maximum number of clusters observed among the supplied clusterings. We considered the SALSO algorithm with default settings (10 zealous updates, 0.5 probability of sequential allocation) having both constrained (by the maximum observed) and unconstrained number of clusters. The results are shown for Binder and VI losses in Table 5. The constraint is clearly successful in limiting the number of clusters. Notice that especially for the SIMCE set, the number of clusters is huge, leading to solutions that are hard to interpret and take substantially more CPU time and, although not shown here, more RAM. Because the constrained algorithm yields more practical results, we will use the constrained algorithm for the remainder of the study. Another important point from Table 5 is the success of the generalized Binder and the generalized VI in controlling the number of clusters. Table 5 also considers the Binder loss with $a = 4/3, b = 2/3$ (labeled "Binder(4/3)"), and our generalized variation of information with $a = 2/3, b = 4/3$ (labeled "VI(2/3)"). The "Binder(4/3)" sets the penalty for incorrectly separating items that should be clustered together to be twice as much as the penalty for incorrectly clustering items that should be separate. Similarly, the "VI(2/3)" loss represents the generalized VI loss in which the penalty for incorrectly separating items is half that of incorrectly clustering items. The loss functions with unequal weights are discussed in Section 3. Note that "Binder(4/3)" yields estimates

Table 5. Mean number of clusters and the run time of: (A) the SALSO algorithm with any number of clusters and (B) the SALSO algorithm constrained to yield an estimate with no more clusters than the maximum number of clusters among the samples.

		# of Clusters		Run time		
Dataset	Loss	A	B	A	B	Ratio
PM10	Binder	6.35	**6.01**	0.01	**0.01**	1.05
	Binder(4/3)	3.75	**3.75**	0.01	**0.01**	1.12
	VI	3.90	**3.84**	0.01	**0.01**	1.10
	VI(2/3)	10.19	**6.37**	0.01	**0.01**	1.34
Gaskins	Binder	30.59	**16.34**	0.20	**0.09**	2.34
	Binder(4/3)	7.16	**7.16**	0.09	**0.07**	1.34
	VI	4.46	**4.44**	0.09	**0.07**	1.31
	VI(2/3)	25.65	**15.74**	0.27	**0.12**	2.33
SIMCE	Binder	262.29	**10.98**	12.30	**0.26**	46.86
	Binder(4/3)	38.43	**7.79**	1.94	**0.29**	6.60
	VI	2.93	**2.93**	0.65	**0.16**	4.05
	VI(2/3)	3.76	**3.55**	1.13	**0.21**	5.41

NOTE: "Binder(4/3)" and "VI(2/3)" are losses with unequal weights. The constrained optimization and the unequal weights are both able to successfully control the number of clusters and lead to faster computations. **A**: SALSO (10, 0.5), 1 run, unconstrained, **B**: SALSO (10, 0.5), 1 run, constrained

Table 6. The quality of: (A) the default SALSO algorithm with four runs, (B) the default SALSO algorithm with 1 run, and (C) our implementation of the Rastelli and Friel (2018) algorithm with multiple runs executing as long as (A).

		Quality			Quality		
Dataset	Loss	A < B	B < A	Diff.	A < C	C < A	Diff.
PM10	Binder	**0.05**	0.00	0.05	**0.01**	0.01	0.00
	Binder(4/3)	**0.01**	0.00	0.01	**0.01**	0.00	0.01
	VI	**0.07**	0.00	0.07	**0.09**	0.01	0.08
	VI(2/3)	**0.22**	0.03	0.19	**0.08**	0.06	0.02
Gaskins	Binder	**0.76**	0.18	0.58	0.44	**0.47**	−0.02
	Binder(4/3)	**0.01**	0.00	0.01	**0.00**	0.00	0.00
	VI	**0.12**	0.01	0.12	**0.14**	0.01	0.13
	VI(2/3)	**0.62**	0.13	0.49	0.33	**0.35**	−0.03
SIMCE	Binder	**0.42**	0.09	0.33	**0.26**	0.24	0.02
	Binder(4/3)	**0.21**	0.05	0.16	**0.16**	0.10	0.06
	VI	**0.06**	0.01	0.06	**0.22**	0.02	0.21
	VI(2/3)	**0.14**	0.01	0.14	**0.11**	0.03	0.08

NOTE: (C) is the SALSO algorithm without zealous updates or sequential allocation. **A**: SALSO (10, 0.5), 4 runs, **B**: SALSO (10, 0.5), 1 run, **C**: SALSO (0, 0.0), timed to 4 runs of SALSO (10, 0.5).

with fewer mean number of clusters than the regular "Binder" estimates for both the unconstrained and constrained settings. The "VI(2/3)" clustering estimates have more mean number of clusters than the regular "VI" estimates for both settings. This shows that the modifications proposed in Section 3 influence the number of clusters.

The next part of the study compares four runs of the SALSO algorithm with default settings to a single run of the SALSO algorithm with default settings. Table 6 shows these results in the comparison between (A) and (B). Four runs of the SALSO algorithm obtains a better answer on average than the single run in every case. In contrast to the recommendation of Rastelli and Friel (2018), we find that multiple runs are indeed beneficial. Since the SALSO algorithm is embarrassingly parallel, these four runs can be performed in essentially the same amount of time as a single run when using a computer with four or more cores.

When comparing with the implementation of Rastelli and Friel (2018) as shown in Table 4, the SALSO algorithm obtained

Table 7. The mean execution speed (in seconds) among 10 replicates at various number of items n for: (a) the default SALSO algorithm with 1 run, and (b) the Rastelli and Friel (2018) software using default settings.

n	100	200	400	800	1600	3200	6400	12,800	25,600
SALSO	0.02	0.06	0.13	0.26	0.75	2.16	6.49	26.27	54.25
R&F 2018	0.43	1.89	5.09	13.71	48.78	186.10	–	–	–

NOTE: The Rastelli and Friel (2018) software encountered memory errors beyond $n = 3200$.

a better answer in a faster time than the R&F algorithm on average. However, as noted in Section 4, the Rastelli and Friel (2018) algorithm as implemented in `GreedyEPL` does not control well the number of clusters and, further, there could be differences in the efficiency of the implementations. We therefore compare SALSO against the our implementation of the Rastelli and Friel (2018) algorithm, using the same constraint on the number of clusters for both methods. Specifically, we compare the SALSO algorithm with the default settings (10 zealous updates, 0.5 probability of sequential allocation) to the SALSO algorithm with 0 zealous updates and 0 probability of sequential allocation, where the number of clusters is no more than that observed among the supplied clusterings. Of course, for a fixed number of iterations, SALSO with zealous updates can do no worse on average than SALSO without zealous updates on average, but zealous updates add CPU cost. It is interesting to compare these two algorithms for a fixed time budget. To this end, both algorithms run for a fixed amount of time, which is roughly the time that it takes for four runs of the SALSO algorithm with default settings. These results are shown in Table 6 in the comparison between (A) and (C). In the vast majority of cases, it appears that the SALSO algorithm with default settings is obtaining a better answer on average than our implementation of the R&F algorithm, even though our implementation of the R&F algorithm can perform more runs in the fixed CPU budget.

Finally, we considered a very large dataset containing 1000 partitions of 30,104 items from the model in Denti et al. (2021). Table 7 shows the mean executive speed (in seconds) for 1 run among 10 replicates at various sizes of random subsets of the full 30,104 items when using VI loss for the default SALSO algorithm and `GreedyEPL` of Rastelli and Friel (2018) using default settings. Note that `GreedyEPL` encountered a memory allocation error beyond $n = 3200$. We found that `GreedyEPL` would work for larger datasets and is considerably faster when overriding the default value $k_d = n$ (i.e., `Kup` in their software) such that k_d is no more than the maximum number of clusters observed among the sampled partitions. With this restriction, we compared the execution speed and the quality of the estimates for both methods on the full dataset of 30,104 items. Among 50 replications, SALSO with 1 run executed in 98 sec on average whereas `GreedyEPL` took 126 sec, yet SALSO provided an estimate with lower expected VI loss in 62% of the cases (whereas `GreedyEPL` did so in only 38% of the cases). The improvement by SALSO seems to be caused by the zealous updates, where 22% of the proposed zealous updates were accepted.

7. Conclusions

This article addresses the problem of point estimation of a partition based on samples from a partition distribution. We view the major contributions of this article as 4-fold. First, we provide a comprehensive review of decades of work on partition estimation based on samples. Second, we propose the generalized variation of information (GVI) loss which allows for differential weights on the two classification mistakes. We prove that the GVI is a quasi-metric and show that it can control the number of clusters. Likewise, the Binder (1978) loss function, as originally proposed, allows for differential weights and we note that the literature seems to have forgotten the advantage of differential weights in controlling the number of clusters. We suspect this realization may revitalize Binder loss in the community. The third contribution is the SALSO algorithm, which is a novel greedy search algorithm over the space of partitions to minimize the Monte Carlo estimate of the expected loss. One key aspect of the algorithm is the so-called zealous updates. The algorithm is amenable for any loss function and benefits from caching. In the case of Binder loss and the GVI, additional computational shortcuts allow for even more computational efficiency, allowing for optimization for large values of n. The last contribution is a parallel, computationally-efficient implementation of the SALSO algorithm, along with the new and many existing loss functions, in the form of the `salso` package on CRAN.

As a in the Binder loss and GVI loss encodes a practitioners judgment as to relative trade-offs of competing mistakes, it seems impossible to declare what the "correct" value for a would be. In practice, we recommend starting with $a = b$, but adjusting a as desired. For example, if $a = b$ yields more clusters than can easily be interpreted for the application, the practitioner can use a greater than b to find a more parsimonious clustering. Or, a might be adjusted such that the partition estimate has about the same number of clusters as the partition samples. Finally, in general, we feel more comfortable with partition estimates which are robust to a wide range of values of a.

Appendix A.

A.1. Proof of General Form of Binder Loss

Here we prove the equivalence between the Binder loss in (3)—as it was originally stated by Binder (1978)—and our expression in terms of the contingency table in (11). As described in Section 5, this lends itself to computationally efficient optimization of the posterior expected loss. Note that $a, b > 0$ are the costs of the misclassification mistakes discussed in Section 3. We use the notation in the contingency table of Table 1.

$$
\begin{aligned}
L_{\text{Binder}}(\mathbf{c}, \hat{\mathbf{c}}) &= \sum_{i<j} \big(a \cdot \mathbb{I}\{c_i = c_j\}\mathbb{I}\{\hat{c}_i \neq \hat{c}_j\} + b \cdot \mathbb{I}\{c_i \neq c_j\}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\big) \\
&= \sum_{i<j} \big(a \cdot \mathbb{I}\{c_i = c_j\}(1 - \mathbb{I}\{\hat{c}_i = \hat{c}_j\}) + b \cdot (1 - \mathbb{I}\{c_i = c_j\})\mathbb{I}\{\hat{c}_i = \hat{c}_j\}\big) \\
&= a\sum_{i<j} \mathbb{I}\{c_i = c_j\} + b\sum_{i<j} \mathbb{I}\{\hat{c}_i = \hat{c}_j\} - (a+b)\sum_{i<j} \mathbb{I}\{c_i = c_j\}\mathbb{I}\{\hat{c}_i = \hat{c}_j\}
\end{aligned}
$$

$$= a\sum_{i=1}^{k}\binom{n_{i\cdot}}{2} + b\sum_{j=1}^{\hat{k}}\binom{n_{\cdot j}}{2} - (a+b)\sum_{i=1}^{k}\sum_{j=1}^{\hat{k}}\binom{n_{ij}}{2}$$

$$= \frac{1}{2}\left(a\sum_{i=1}^{k} n_{i\cdot}^2 + b\sum_{j=1}^{\hat{k}} n_{\cdot j}^2 - (a+b)\sum_{i=1}^{k}\sum_{j=1}^{\hat{k}} n_{ij}^2 - a\sum_{i=1}^{k} n_{i\cdot} - b\sum_{j=1}^{\hat{k}} n_{\cdot j} + (a+b)\sum_{i=1}^{k}\sum_{j=1}^{\hat{k}} n_{ij}\right)$$

$$= \frac{1}{2}\left(a\sum_{S\in\rho}|S|^2 + b\sum_{E\in\hat{\rho}}|E|^2 - (a+b)\sum_{S\in\rho}\sum_{E\in\hat{\rho}}|S\cap E|^2\right).$$

Thus, the generalized Binder loss is

$$L_{\text{GBinder}}(\mathbf{c},\hat{\mathbf{c}}) = \frac{2}{n^2}L_{\text{Binder}}(\mathbf{c},\hat{\mathbf{c}}) = a\sum_{S\in\rho}\left(\frac{|S|}{n}\right)^2 + b\sum_{E\in\hat{\rho}}\left(\frac{|E|}{n}\right)^2 - (a+b)\sum_{S\in\rho}\sum_{E\in\hat{\rho}}\left(\frac{|S\cap E|}{n}\right)^2.$$

A.2. Proofs of Quasi-metric Properties

Here we prove Theorems 1 and 2, which state that the generalized Binder loss and our generalized variation of information (GVI) are quasi-metrics. For Binder loss, it is helpful to recall the definition in (3). Likewise, for the GVI, recall the identities in Section 2.2 and the definition in (12). In order to satisfy the quasi-metric property, a distance measure must have: i. the identity of indiscernibles and ii. the triangle inequality.

A loss function L satisfies the identity of indiscernibles when $L(\rho,\hat{\rho}) = 0 \iff \rho = \hat{\rho}$. (Meilă 2007, see, p. 879) states that $\rho = \hat{\rho} \iff I(\rho,\hat{\rho}) = H(\rho) = H(\hat{\rho})$. Therefore, the identity of indiscernibles holds, since the GVI can be written in the following form:

$$\begin{aligned} L_{\text{GVI}}(\rho,\hat{\rho}) &= bH(\rho) + aH(\hat{\rho}) - (a+b)I(\rho,\hat{\rho}) \\ &= b\left\{H(\rho) - I(\rho,\hat{\rho})\right\} + a\left\{H(\hat{\rho}) - I(\rho,\hat{\rho})\right\}. \end{aligned}$$

The identity of indiscernibles is straightforward for the generalized Binder loss as defined in (3). It is clear that if two clusterings are equivalent, there will be no disagreements for any i,j pair, and thus 0 loss will be incurred. Conversely, if the two clusterings are not equivalent, there will be a disagreement for at least one i,j pair, and a strictly positive loss of either a or b will be incurred. The triangle inequality requires that, for any three partitions ρ_1,ρ_2,ρ_3 in the space of all possible partitions, $L(\rho_1,\rho_2) + L(\rho_2,\rho_3) \geq L(\rho_1,\rho_3)$. Note that this is equivalent to proving that $L(\rho_1,\rho_2) + L(\rho_2,\rho_3) - L(\rho_1,\rho_3) \geq 0$, as shown below.

$$\begin{aligned} &L_{\text{GVI}}(\rho_1,\rho_2) + L_{\text{GVI}}(\rho_2,\rho_3) - L_{\text{GVI}}(\rho_1,\rho_3) \\ &\quad= bH(\rho_1) + aH(\rho_2) - (a+b)I(\rho_1,\rho_2) + bH(\rho_2) \\ &\qquad + aH(\rho_3) - (a+b)I(\rho_2,\rho_3) \\ &\qquad - bH(\rho_1) - aH(\rho_3) + (a+b)I(\rho_1,\rho_3) \\ &\quad= (a+b)[H(\rho_2) - I(\rho_1,\rho_2) - I(\rho_2,\rho_3) + I(\rho_1,\rho_3)] \\ &\quad= (a+b)[H(\rho_2) - (H(\rho_1) + H(\rho_2) - H(\rho_1,\rho_2)) \\ &\qquad - (H(\rho_2) + H(\rho_3) - H(\rho_2,\rho_3)) \\ &\qquad + (H(\rho_1) + H(\rho_3) - H(\rho_1,\rho_3))] \\ &\quad= (a+b)[(H(\rho_1,\rho_2) - H(\rho_2)) + (H(\rho_2,\rho_3) - H(\rho_3)) \\ &\qquad - (H(\rho_1,\rho_3) - H(\rho_3))] \\ &\quad= (a+b)[H(\rho_1 \mid \rho_2) + H(\rho_2 \mid \rho_3) - H(\rho_1 \mid \rho_3)] \geq 0 \end{aligned}$$

The final line follows because $a, b > 0$ by definition and $[H(\rho_1 \mid \rho_2) + H(\rho_2 \mid \rho_3) - H(\rho_1 \mid \rho_3)] \geq 0$ is established by (Meilă 2007, see proof of Property 1, (35)–(37)). Therefore, the product is greater than or equal to 0, so the triangle inequality holds for the GVI.

Now, consider the following proof for the generalized Binder loss. Let the clusterings $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ correspond to the equivalent partitions ρ_1, ρ_2, and ρ_3. For convenience, the notation $\mathbb{X}_{i=j}$ is used in place of the indicator function $\mathbb{I}\{x_i = x_j\}$.

$$\begin{aligned} &L_{\text{Binder}}(\mathbf{x},\mathbf{y}) + L_{\text{Binder}}(\mathbf{y},\mathbf{z}) - L_{\text{Binder}}(\mathbf{x},\mathbf{z}) \\ &\quad= \sum_{i<j}\left(\left(a\mathbb{X}_{i=j}\mathbb{Y}_{i\neq j} + b\mathbb{X}_{i\neq j}\mathbb{Y}_{i=j}\right) + \left(a\mathbb{Y}_{i=j}\mathbb{Z}_{i\neq j} + b\mathbb{Y}_{i\neq j}\mathbb{Z}_{i=j}\right)\right. \\ &\qquad \left. - \left(a\mathbb{X}_{i=j}\mathbb{Z}_{i\neq j} + b\mathbb{X}_{i\neq j}\mathbb{Z}_{i=j}\right)\right) \\ &\quad= a\sum_{i<j}\left(\mathbb{X}_{i=j}\mathbb{Y}_{i\neq j} + \mathbb{Y}_{i=j}\mathbb{Z}_{i\neq j} - \mathbb{X}_{i=j}\mathbb{Z}_{i\neq j}\right) \\ &\qquad + b\sum_{i<j}\left(\mathbb{X}_{i\neq j}\mathbb{Y}_{i=j} + \mathbb{Y}_{i\neq j}\mathbb{Z}_{i=j} - \mathbb{X}_{i\neq j}\mathbb{Z}_{i=j}\right) \geq 0 \end{aligned}$$

The final line holds by the following logic. Since $a > 0$ and $b > 0$ by definition, we simply need to show that both sums are nonnegative, since this makes their positively weighted sum also nonnegative. The only way that the first sum could be negative is if there exists some pair i,j such that $\mathbb{X}_{i=j}\mathbb{Z}_{i\neq j}$ can evaluate to one while $\mathbb{X}_{i=j}\mathbb{Y}_{i\neq j}$ and $\mathbb{Y}_{i=j}\mathbb{Z}_{i\neq j}$ both evaluate to zero. If, for a specific i,j pair, $x_i = x_j$ and $z_i \neq z_j$, then $\mathbb{X}_{i=j}\mathbb{Z}_{i\neq j} = 1$. However, for both of the other terms to evaluate to 0 (and thus yielding -1 for that summand), we must have that $x_i = x_j$, $y_i \neq y_j$, $y_i = y_j$, and $z_i \neq z_j$. It is clear that $y_i = y_j$ and $y_i \neq y_j$ are mutually exclusive, and therefore the first sum can never be negative. Similar logic can be used to show that the second sum can never be negative. Therefore, the triangle inequality holds for the generalized Binder loss.

Supplementary Materials

Scripts to reproduce the empirical results in the article are provided.

ORCID

David B. Dahl https://orcid.org/0000-0002-8173-1547
Devin J. Johnson http://orcid.org/0000-0003-2619-6649

References

Binder, D. A. (1978), "Bayesian Cluster Analysis," *Biometrika*, 65, 31–38. [1190,1199]

Dahl, D. B. (2006), "Model-Based Clustering for Expression Data via a Dirichlet Process Mixture Model," in *Bayesian Inference for Gene Expression and Proteomics*, eds. K.-A. Do, P. Müller, and M. Vannucci, pp. 201–218, Cambridge: Cambridge University Press. [1190,1194]

Dahl, D. B., and Newton, M. A. (2007), "Multiple Hypothesis Testing by Clustering Treatment Effects," *Journal of the American Statistical Association*, 102, 517–526. [1190]

Denti, F., Camerlenghi, F., Guindani, M., and Mira, A. (2021), "A Common Atoms Model for the Bayesian Nonparametric Analysis of Nested Data," *Journal of the American Statistical Association*, 1–12. doi:10.1080/01621459.2021.1933499 [1199]

Fritsch, A., and Ickstadt, K. (2009), "Improved Criteria for Clustering Based on the Posterior Similarity Matrix," *Bayesian Analysis*, 4, 367–391. [1191,1192,1194]

Gaskins, J. T., Fuentes, C., and la Cruz, R. D. (2017), "A Bayesian Nonparametric Model for Predicting Pregnancy Outcomes Using Longitudinal Profiles," available at *https://arxiv.org/abs/1711.01512*. [1197]

Hubert, L., and Arabie, P. (1985), "Comparing Partitions," *Journal of Classification*, 2, 193–218. [1191]

Lau, J. W., and Green, P. J. (2007), "Bayesian Model-Based Clustering Procedures," *Journal of Computational and Graphical Statistics*, 16, 526–558. [1190,1192,1194]

Medvedovic, M., and Sivaganesan, S. (2002), "Bayesian Infinite Mixture Model Based Clustering of Gene Expression Profiles," *Bioinformatics*, 18, 1194–1206. [1194]

Meilă, M. (2007), "Comparing Clusterings—An Information Based Distance," *Journal of Multivariate Analysis*, 98, 873–895. [1190,1191,1192,1193,1200]

Page, G. L., Quintana, F. A., and Dahl, D. B. (2021), "Dependent Modeling of Temporal Sequences of Random Partitions," *Journal of Computational and Graphical Statistics*, 1–14, accepted, DOI: 10.1080/10618600.2021.1987255. [1197]

Rand, W. M. (1971), "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, 66, 846–850. [1191]

Rastelli, R., and Friel, N. (2018), "Optimal Bayesian Estimators for Latent Variable Cluster Models," *Statistics and Computing*, 28, 1169–1186. [1190,1191,1192,1194,1195,1196,1197,1198,1199]

Vinh, N. X., Epps, J., and Bailey, J. (2010), "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *The Journal of Machine Learning Research*, 11, 2837–2854. [1190,1191,1192]

Wade, S., and Ghahramani, Z. (2018), "Bayesian Cluster Analysis: Point Estimation and Credible Balls," (with discussion), *Bayesian Anal.*, 13, 559–626. [1190,1191,1192,1193,1194,1196]