Federated Variational Inference for Bayesian Mixture Models

Jackie Rao JACKIE.RAO@MRC-BSU.CAM.AC.UK
MRC Biostatistics Unit, University of Cambridge

Francesca L. Crowe F.CROWE@BHAM.AC.UK
Institute of Applied Health Research, University of Birmingham

Tom Marshall T.P.MARSHALL@BHAM.AC.UK
Institute of Applied Health Research, University of Birmingham

Sylvia Richardson SYLVIA.RICHARDSON@MRC-BSU.CAM.AC.UK
MRC Biostatistics Unit, University of Cambridge

Paul D. W. Kirk PAUL.KIRK@MRC-BSU.CAM.AC.UK
MRC Biostatistics Unit, University of Cambridge

Abstract

We present a one-shot, unsupervised federated learning approach for Bayesian model-based clustering of large-scale binary and categorical datasets, motivated by the need to identify patient clusters in privacy-sensitive electronic health record (EHR) data. We introduce a principled 'divide-and-conquer' inference procedure using variational inference with local merge and delete moves within batches of the data in parallel, followed by 'global' merge moves across batches to find global clustering structures. We show that these merge moves require only summaries of the data in each batch, enabling federated learning across local nodes without requiring the full dataset to be shared. Empirical results on simulated and benchmark datasets demonstrate that our method performs well relative to comparator clustering algorithms. We validate the practical utility of the method by applying it to a large-scale British primary care EHR dataset to identify clusters of individuals with common patterns of co-occurring conditions (multimorbidity).

Keywords: Federated learning, Bayesian inference, cluster analysis

Data and Code Availability Code is available here, which includes code for generation of all simulated datasets and random seeds used in this paper.

The MNIST dataset used is publicly available (http://yann.lecun.com/exdb/mnist/), and details and code for preprocessing are available in the above repository. The THIN EHR dataset used is not publicly available, but we provide details regarding preprocessing in the Appendix.

Institutional Review Board (IRB) Collection of data in THIN was approved by the NHS South East Multi-Centre Research Ethics Committee (MREC). Approval to conduct this analysis was obtained from the Scientific Review Committee (reference number: 21SRC055).

1. Introduction

Federated learning (FL) refers to the setting in which a network of clients collaboratively train a model while keeping training data local (Kairouz et al., 2021), introduced for privacy-sensitive datasets (McMahan et al., 2017) common in healthcare. While supervised FL has seen many developments, unsupervised FL methods remain limited (Tian et al., 2024). With the primary motivation of clustering electronic health record (EHR) data across multiple hospitals, we address: (i) preserving data privacy while learning global clustering structure, (ii) enabling scalability to large patient record datasets, and (iii) maintaining clustering accuracy in distributed environments. While various federated clustering algorithms and Bayesian FL methods exist (Cao et al., 2023), there is limited work on FL for unsupervised Bayesian model-based clustering. Our method implements a one-shot FL scheme where clients perform local computation and share summary statistics with a central server in a single round, enabling asynchronous participation ideal for healthcare institutions. By combining clustering information across sites, we identify

population-level disease clusters transcending institutional boundaries, enabling characterization of multimorbidity patterns across diverse populations impossible with site-specific clustering alone.

Mixture models provide a probabilistic approach for clustering by assuming data are generated from a mixture of underlying probability distributions. Bayesian mixture models enable the uncertainty in cluster allocations and, potentially, the number of clusters to be modelled. Markov Chain Monte Carlo (MCMC), particularly Gibbs sampling (Neal, 2000), is often used to draw samples from the intractable posterior (e.g. Robert and Casella, 2004; Walker, 2007). Despite efforts to improve scalability, MCMC algorithms are often too computationally intensive for even moderately large datasets, and/or may experience mixing issues, preventing the sampler from reaching convergence (Celeux et al., 2000).

To address the growth of 'massive' datasets, a range of methods have been proposed to scale algorithms to large sample sizes (e.g Huang and Gelman, 2005). These include distributed computing approaches in which the computational and memory requirements of an algorithm are split, with the algorithm running in parallel on multiple nodes (Zhu et al., 2017). However, models with unidentifiable latent variables present additional complexity for such approaches. Furthermore, few distributed computing approaches are in the spirit of FL, where data heterogeneity, data privacy, asynchronous client participation and node dropout are challenges to address.

While variational inference (VI) methods tend to have better run-time performance than MCMC algorithms (e.g. Blei et al., 2017), they can also be sensitive to initialisation due to local optima in the objective function. To address these issues when fitting mixture models, 'merge' and 'delete' moves may be incorporated into the variational algorithm to dynamically combine or remove unnecessary clusters. Such moves, in addition to 'split' moves to create new clusters, have improved inference in expectation-maximisation (EM) algorithms (Ueda et al., 2000) and MCMC samplers (Jain and Neal, 2004; Dahl and Newcomb, 2021) for mixture models, and have also been employed less commonly in VI algorithms for Dirichlet process (DP) mixtures, including DP mixtures of Gaussians (Hughes et al., 2015; Lin, 2013).

Here we propose a 'one-shot' distributed algorithm that performs inference for Bayesian mixture models using VI with local merge and delete moves within batches of the data across a network of nodes in parallel. This is followed by 'global' merge moves to aggregate clusters across batches, utilising the variational objective function to combine results from data partitions in a statistically principled manner.

We focus on applications to categorical, particularly binary (2-category), datasets, as is common in EHR datasets such as the one studied later in the manuscript. Binary data analysis has a long history in the context of characterising multimorbidity across populations (see e.g. Cornell et al., 2009; Ng et al., 2018; Nichols et al., 2022) but there has been limited research into clustering methodology for binary and categorical data. By applying our method to large-scale EHR data, we demonstrate how global clustering structures (in this case, population-level multimorbidity patterns) can be identified without compromising data privacy. Our approach can be adapted to other members of the exponential family, and we consider in the Appendix how our model can be extended to tackle variable selection.

2. Related Work

A central challenge in scaling and distributing Bayesian inference is maintaining statistical accuracy while distributing computation. One could consider parallelising steps within VI to enhance scalability, and we look at this in Section 4.1. In the MCMC setting, an important example of distributed inference is *consensus Monte Carlo* (Scott et al., 2016), where Monte Carlo sampling is performed independently on partitions of the data, and the results are then aggregated to approximate a global posterior. While effective for certain models, these approaches typically struggle with models involving unidentifiable latent variables. Informally, the challenge for clustering models is how to match clusters/cluster labels across data partitions. Ge et al. (2015) considers a distributed inference algorithm for the DP and HDP mixture models with a variant of the DP slice sampler, but requires frequent communication between servers. The SNOB (Zuanetti et al., 2018) and SIGN (Ni et al., 2019) algorithms also perform inference for distributed mixture models, where MCMC samplers are run locally on partitions of the data, before local clusters are combined into global clusters with another sampler, allowing for asynchronous operation. These models are conceptually closest to our proposed model, although they do not consider the federated learning (FL) setting.

Distributed VI algorithms have also been proposed to obtain an approximation of the posterior distribution for mixture models, including variational consensus Monte Carlo (VCMC) (Rabinovich et al., 2015), where the authors suggested a heuristic approach using the Hungarian algorithm (Kuhn, 1955) to align cluster centres across data partitions. Our proposed method instead utilises the variational objective function to combine results from data partitions in a statistically principled manner. Campbell et al. (2015) proposes a streaming, distributed VI method for the DP mixture model, but does not address data privacy. Zhang et al. (2019) extends stochastic VI (Hoffman et al., 2013) to a distributed setting for the inference of mixture models, but relies on frequent parameter exchange between processors, despite efforts to make the algorithm asynchronous.

The majority of work on Bayesian inference in FL targets supervised tasks (see Cao et al., 2023, for a survey). These approaches have been extended to tackle e.g. Bayesian neural networks for local classification or regression tasks (Bhatt et al., 2024), and Bayesian nonparametric methods for dynamic learning of local models (Achituve et al., 2021). Some works explore clustering clients in FL to tackle data heterogeneity in supervised settings (Sattler et al., 2021; Briggs et al., 2020). This is popular in personalised FL, concerning the learning of individual local distributions for clients in a federated network. For example, in Zhang et al. (2023), clients in the same cluster share a prior, and Wu et al. (2023) represents clients with mixtures of shared-parameter distributions where clients have different mixture weights. These methods aim to improve performance on downstream prediction tasks, but do not cluster data itself, differing fundamentally from our objective.

Far fewer works have explored *unsupervised* FL. Most adopt heuristic approaches like federated k-means (Kumar et al., 2020; Liu et al., 2020; Garst and Reinders, 2025; Dennis et al., 2021) or fuzzy clustering (Stallmann and Wilbik, 2022; Pedrycz, 2022) lacking probabilistic foundations, while FedEM (Dieuleveut et al., 2021) provides probabilistic modeling but requires frequent communication rounds.

Overall, our work fills a gap at the intersection of unsupervised FL, Bayesian clustering, and scalable distributed inference. To the best of our knowledge, we are the first to propose a communication-efficient, one-shot federated algorithm for Bayesian model-based clustering of categorical and binary data.

3. Model Overview

3.1. Bayesian Finite Mixture Models

Let $X = \{x_1, ..., x_N\}$ denote the observed data, where x_n is one observation of P categorical variables. We model the generating distribution as a finite mixture of K components. Each observation is generated by one component. The general equation for a K component finite mixture model is:

$$p(X|\boldsymbol{\pi}, \boldsymbol{\phi}) = \prod_{n=1}^{N} p(x_n|\boldsymbol{\pi}, \boldsymbol{\phi}), \tag{1}$$

$$\text{where} \qquad p(x_n|\boldsymbol{\pi}, \boldsymbol{\phi}) = \sum_{k=1}^{K} \pi_k f(x_n|\phi_k), \tag{2}$$

where component densities $f(x_n|\phi_k)$ are usually from the same parametric family but with different parameters associated with each component. Under certain assumptions, the Dirichlet process (DP) mixture model may be derived by considering the limit as K goes to infinity (Maceachern, 1994; Escobar and West, 1995; Neal, 2000). We consider the so-called 'overfitted mixtures' setting in which K is set to be larger than the number of clusters we expect, which enables the number of clusters to be inferred (Rousseau and Mengersen, 2011). Each component is modelled as a categorical distribution across P covariates with parameters $\phi_{kj} = [\phi_{kj1}, ..., \phi_{kjL_j}]$, where ϕ_{kjl} represents the probability that variable j takes value l in component k. L_j is the number of categories for variable j. The mixture weight for the k-th component is denoted by π_k, satisfying $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$.

We introduce latent variables z_n associated with each x_n, such that each z_n is a one-hot-encoded binary vector with $z_{nk} = 1$ if and only if x_n is in the k-th cluster. We then rewrite Equation (2) as:

$$p(X|Z, \boldsymbol{\pi}, \boldsymbol{\phi}) = \prod_{n=1}^{N} \prod_{k=1}^{K} f(x_n|\phi_k)^{z_{nk}}. \tag{3}$$

To complete the model specification for the Bayesian model, we place priors on the parameters:

$$\boldsymbol{\pi} = (\pi_1, ..., \pi_K) \sim \text{Dirichlet}(\alpha_0), \tag{4}$$

$$\phi_{kj} = (\phi_{kj1}, ..., \phi_{kjL_j}) \sim \text{Dirichlet}(\epsilon_j), \tag{5}$$

where all Dirichlet priors are symmetric. It has been shown theoretically that if K exceeds the true number of clusters, then, under certain assumptions, setting $\alpha_0 < 1$ in Equation (4) allows the posterior to asymptotically converge to the correct number of

clusters as the number of observations goes to infinity (Rousseau and Mengersen, 2011; van Havre et al., 2015; Malsiner-Walli et al., 2014). Further details of hyperparameter settings are provided in the Appendix. The model can be extended to simultaneously tackle variable selection, as in Law et al. (2004) and Tadesse et al. (2005), which is explored in the Appendix. This accounts for noisy features not contributing to the clustering structure, which can help address data corruption in real-world data.

3.2. Mean-Field Variational Inference (VI)

We employ a variational inference (VI) approach to seek an approximate distribution q close to the true posterior. We optimise the Evidence Lower Bound (ELBO) below, $\mathcal{L}(q)$, with respect to $q(\theta)$, where θ is a collection of all parameters in the model.

$$\mathcal{L}(q) = \int q(\theta) \ln \left(\frac{p(X, \theta)}{q(\theta)} \right) d\theta \tag{6}$$

This is equivalent to minimising the Kullback-Leibler (KL) divergence between $q(\theta)$ and the true posterior, $p(\theta|X)$. We constrain q to be a mean-field approximation which can be fully factorised as: $q(\theta) = q_Z(Z)q_\pi(\pi)q_\Phi(\Phi)$. We optimise $\mathcal{L}(q)$ using a standard variational EM algorithm, also known as Coordinate Ascent Variational Inference (see, for example, Bishop, 2006), as described below.

'E' Step We update the latent variables Z with the current variational distributions of all other parameters.

$$q^*(Z) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}}, \qquad r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}} \tag{7}$$

$$\ln \rho_{nk} = \mathbb{E}_\pi[\ln \pi_k] + \sum_{i=1}^{P} \mathbb{E}_\Phi[\ln \phi_{kix_{ni}}] \tag{8}$$

r_{nk} is the responsibility of the k-th component for the n-th observation, and $\mathbb{E}[z_{nk}] = r_{nk}$.

'M' Step Given updated responsibilities r_{nk} in the 'E' step, we then update cluster-specific parameters:

$$q^*(\pi) = \text{Dirichlet}(\alpha_1^*, ..., \alpha_K^*), \tag{9}$$

$$q^*(\phi) = \prod_{k=1}^{K} \prod_{j=1}^{P} \text{Dirichlet}(\epsilon_{kj1}^*, ..., \epsilon_{kjL_j}^*) \tag{10}$$

where for $k = 1, ..., K, j = 1, ..., P, l = 1, ..., L_j$:

$$\alpha_k^* = \alpha_0 + \sum_{n=1}^{N} r_{nk}, \qquad \epsilon_{kjl}^* = \epsilon_{jl} + \sum_{n=1}^{N} \mathbb{I}(x_{nj} = l) r_{nk} \tag{11}$$

Expectations throughout are taken with respect to variational distributions, with ELBO monitored for convergence. All values are initialised using k-modes (Chaturvedi et al., 2001). The algorithm terminates when the ELBO is within a certain tolerance for 3 iterations in a row.

4. Variational Merge/Delete Moves

As in e.g. Hughes et al. (2015), we introduce merge and delete moves which can be integrated into the VI algorithm. Merge moves combine observations from two clusters into one unified cluster; delete moves remove unnecessary extra clusters. Both moves effectively eliminate redundant clusters to help the model to escape local optima, and are designed in a principled manner to align with the ELBO variational objective function. When moves are proposed, parameters in the VI framework are updated through initially updating responsibilities, r_{nk} which are directly associated with cluster assignment. For example, to merge clusters k_1 and k_2, we set $r_{nk^*} = r_{nk_1} + r_{nk_2}$ for a new unified cluster k^*, and perform a variational M step to update π, ϕ. We perform additional E and M steps to allow observation reassignment, and accept moves only if they improve the ELBO, otherwise we maintain the original configuration.

Variational implementations of these moves in Dirichlet process mixture models, in addition to split moves, have been shown to outperform standard variational Bayes and related algorithms (Hughes et al., 2015; Yang et al., 2019). However, little work has been published regarding variational categorical mixture models. Clear, practical strategies for the implementation of these moves in the categorical setting are provided in Appendix C. One of our key contributions lies in developing efficient strategies specifically for categorical mixture models to select candidate clusters to merge/delete. Comparing ELBOs between all possible merge/delete proposals would result in a high computational cost. We demonstrate in Appendix H.4 that using fast-to-calculate heuristics such as correlation between cluster parameters or divergence between variational distributions when proposing to merge allows for enhanced efficiency. This aspect has rarely been explored.

We name the variational algorithm including merge/delete moves 'MerDel' and introduce a parameter, 'laps', representing how many variational EM cycles we go through before performing a merge and a delete move. Since merge and delete moves include

additional E and M steps (see Appendix C), they are costly compared to standard variational steps which guarantee ELBO improvement. Subsequently, there is a trade-off between proposing more merge/delete moves to allow for bigger, more effective moves earlier on in MerDel versus reducing wasted computation as the model approaches convergence and moves are rejected, which we examine in simulations. One method to mitigate this in future could be to dynamically change the 'laps' parameter.

4.1. Parallelising MerDel

VI requires fewer sweeps over the dataset to perform inference compared to MCMC samplers, and is feasible for datasets with thousands of observations. This allows variational mixture models to be more computationally efficient, especially with merge/delete moves. However, VI still reaches computational bottlenecks when N is very large; we see in simulations later that MerDel is infeasible as we reach datasets with N of order 10^5. Repeatedly updating $N \times K$ responsibilities every E step, and subsequently summing these over N every M step is costly.

Individual components of the original algorithm can be parallelised using standard methods e.g., in the E step, calculating r_{nk} can be parallelised over n. We provide an implementation of such a parallelised VI as part of this manuscript. However, the overall algorithm is not embarrassingly parallel; for example, r_{nk} values for all N observations are required to update cluster-specific parameters for a given cluster k. Prior literature has consistently highlighted the limited gains achieved by such partial parallelisation for iterative methods such as VI. This is due to the significant computational overhead incurred when feeding intermediate results to and from one 'master' core - a globally shared computation unit - at frequent intervals (Nallapati et al., 2007; Neiswanger et al., 2015).

5. Federated Variational Inference

In this section, we propose a scalable clustering algorithm for large categorical datasets, FedMerDel (Federated MerDel), which addresses the challenge of large N and enables federated learning (FL).

In FedMerDel, we use a 'divide and conquer' inference procedure. We assume our dataset has been split into B 'batches' - these batches will already be pre-determined if subsets of the data are held by separate parties, or may be artificially created by random partitioning of the data. The first 'local' step of the algorithm involves carrying out clustering via MerDel in each data batch separately. The size and number of these batches should be chosen so inference is computationally feasible; pre-existing subsets of the data can be divided further if necessary. Each batch is processed in parallel, and the clusters produced are referred to as 'local clusters'.

The second 'global' step then combines the local clustering structures. Local clusters are frozen, and we merge similar local clusters into 'global' clusters. This is conceptually similar to approaches used in SNOB (Zuanetti et al., 2018) and SIGN (Ni et al., 2019), but both use MCMC methods which are computationally costly; local clustering in SNOB takes over 4 hours for a batch of size $N = 1000$ in the authors' simulations. We utilise a novel version of the merge move in MerDel to implement a 'global merge'. By freezing local clusters in the global merge, the proposed clustering structure and 'global ELBO' across all observations can be efficiently calculated by utilising stored parameter values. The variational framework allows us to accept or reject global merges in a principled manner. Importantly, the full dataset is not required in the global merge.

In contrast to the approach considered in Section 4.1, instead of optimising individual components of the existing classical variational algorithm, our approach is designed to present a scalable clustering model which avoids these bottlenecks entirely by adopting a distributed computing approach in which each data batch is analysed independently and in parallel before 'global merge' steps are used to obtain a (global) mixture model for the full dataset. This FL approach eliminates the need for frequent communication and synchronization during the main variational updates, while data remains decentralised.

5.1. Estimating Global Clusters

Assume our data has been split into B batches and MerDel has been run on each batch, with K_b local clusters in Batch b. Let $K = \sum_{b=1}^{B} K_b$ be the total number of local clusters. We begin with α^* and ϵ^* parameters fixed from the completed local clustering steps. In the global step, we first 'combine' all our existing variational parameters from each batch. For $q(\pi)$, we concatenate all our K_b-length α^* vectors to one K-length α^* vector. Each $q(\phi_{kj})$ is already independent for each (k, j) pair. We create a new matrix of responsibilities r_{nk} as an $N \times K$ block diagonal

matrix with blocks of $N_b \times K_b$ responsibility matrices from each batch. $\sum_{k=1}^{K} r_{nk} = 1$ still holds for all n.

Let clusters k_1 and k_2 be considered for a merge. As in the local merge, we reassign responsibilities to cluster k_1 via $r_{nk_1}^{new} = r_{nk_1} + r_{nk_2}$ for all $n = 1, ..., N$. This can be viewed as summing columns k_1 and k_2 in our responsibility matrix. We then use the definitions of α^*, ϵ^* (Equation (11)) as follows to update cluster-specific parameters:

$$\alpha_{k_1}^{*new} = \alpha_0 + \sum_{n=1}^{N}(r_{nk_1} + r_{nk_2}) = \alpha_{k_1}^* + \alpha_{k_2}^* - \alpha_0 \tag{12}$$

$$\epsilon_{k_1jl}^{*new} = \epsilon_{jl} + \sum_{n=1}^{N}\mathbb{I}(x_{nj} = l)(r_{nk_1} + r_{nk_2}) = \epsilon_{k_1jl}^* + \epsilon_{k_2jl}^* - \epsilon_j \tag{13}$$

This avoids calculating sums over N. To complete the merge, we remove all parameters associated with cluster k_2 - $\alpha_{k_2}^*$, $\epsilon_{k_2j}^*$ and r_{nk_2} for all $n = 1, ..., N$, $j = 1, ..., P$. Unlike local merge moves, we do not perform additional E/M steps to allow observations to move in and out of the new cluster. This would require a whole-dataset update of responsibilities r_{nk}. The efficiency of the global merge hinges on local clusters being frozen. We propose global merge candidates using two approaches: greedy search and random search (see Appendix D.1 for details).

5.2. Efficient Global ELBO Calculation

The ELBO variational objective function allows us to accept or reject the global merge under a rigorous probabilistic framework. For the global merge to be accepted, the ELBO must be higher. As we now consider the full dataset, many terms in the overall ELBO function involve all observations and clusters. 'Freezing' local clusters allows us to efficiently calculate the majority of the ELBO terms and enable the model to handle large-scale data. Two sums involving N in the full ELBO function can be written as a sum over smaller K, P, L_j of sufficient statistics which can in turn be calculated as a linear function of previously calculated α^* and ϵ^*: $T_k = \sum_{n=1}^{N} r_{nk} = \alpha_k^* - \alpha_0$ and $S_{kjl} = \sum_{n=1}^{N} r_{nk}\mathbb{I}(x_{nj} = l) = \epsilon_{kjl}^* - \epsilon_{jl}$. This vastly reduces the number of operations required.

The final term involving N is an assignment entropy term, $\sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \ln r_{nk}$, which is not linear in any previously calculated parameters. This is additive over N and K, so we can calculate this value separately for each batch (in parallel if required) before summing results. However, in our greedy search, we never combine two clusters from the same batch, so r_{nk_1} and/or r_{nk_2} will *always* be zero for any merge, for all $n = 1, ..., N$. Observation n is only explained by clusters originating from some batches of data and a merge is only proposed from a new batch where n is not explained by any clusters associated with that batch. This allows greedy search to be faster, although merges could be missed. See Appendix D.2 for full details of the global ELBO calculation.

Crucially, the global merge move allows us to seek a clustering structure for the full dataset without any scan over the full data matrix X. Data from separate entities remains decentralised, where only summary statistics from each batch are required to be shared, allowing for increased data security and privacy. These summary statistics represent cluster-specific weighted averages over batch-level populations, and do not directly expose individual-level data. We note that privacy risks could occur with clusters where these weights are almost all estimated to be close to 0 with a small number close to 1, but such cases can be easily detected and handled (eg. by withholding the cluster). Implementation details are in Appendix F. Parallelisation in local batches (Section 4.1) could be applied to FedMerDel for additional speed gains.

6. Experiments and Results

An additional study looking at unsupervised clustering of the MNIST dataset (LeCun et al., 2011) can be found in Appendix I.

6.1. Simulation Study

In this section, simulated data are binary, with the data generating mechanism detailed in the Appendix. Additional simulation studies comparing merge criteria, looking at categorical data with 3+ categories, and examining variable selection are in the Appendix.

6.1.1. Frequency of Merge/Delete Moves

In Section 4, we described the trade-off between more or fewer merge/delete moves in MerDel. We ran simulation studies comparing different values of the parameter 'laps' representing the frequency of merge/delete moves in a run of MerDel, with laps $\in \{1, 2, 5, 10\}$. We also compared to a model with just merge/delete moves after performing one 'E' and 'M' step after initialisation ('laps = 0'), and the variational model with no merge/delete moves ('laps =

10000'). We compared the wall-clock time taken for convergence, number of non-empty clusters found, and the Adjusted Rand Index (ARI) with the simulated 'true' labels. Appendix G.3 gives further details.

We saw that models incorporating merge/delete moves (laps $\in \{2, 5, 10\}$) consistently outperformed standard variational inference across multiple simulated datasets, achieving higher ARIs, more accurate cluster number estimation, and faster convergence (full results in Appendix H.1). Based on these results, we used laps $= 5$ for subsequent experiments.

6.1.2. Global Merge Simulations

We evaluated the clustering performance when splitting data into batches and performing a global merge as detailed in Section 5. In our simulations, we compared to the "gold standard" setting in which the full dataset may be analysed without splitting of the data using MerDel with or without parallelisation (denoted by 'par' and 'full' in our results tables). We looked at binary data with sizes varying between $N =$ between 10,000 to 1,000,000 with $P = 100$ covariates.

For our simulated datasets, we generated data with known ground truth cluster structure, then initially randomly partitioned this into batches. We also looked at scenarios with heterogeneous data across partitions, where clusters are deliberately generated such that they may only appear in certain batches and not in others. We compared this to parallelised VI on the full data and compared to existing unsupervised federated and distributed algorithms such as k-FED (Dennis et al., 2021), FedEM (Dieuleveut et al., 2021) and SNOB (Zuanetti et al., 2018). These represent the current state-of-the-art for unsupervised federated clustering, particularly for categorical data, in this nascent field. We also compared to some centralised learning methods as well as simulations varying P and looking at categorical data in the Appendix. Full details are in Appendix G.4.

A full results table can be found in Appendix H.2, with results for heterogeneously distributed data shown in Table 1.

When the data were divided into random batches, FedMerDel achieved clustering performance nearly equivalent to the "best possible" centralised baselines, with median ARIs only marginally below those of the full and parallel models (all above 0.9). While local batch-level ARIs were also high, the global merge step provides additional value by producing a unified model for the entire population. In heterogeneous scenarios, FedMerDel occasionally surpassed the full-data baseline (ARI ≈ 0.99). This occurs when batches contain fewer, well-separated clusters, making local clustering more accurate and improving the global merge outcome. Thus, FedMerDel accommodates statistical heterogeneity without forcing clusters to merge across batches, and can offer clustering-quality advantages beyond computational efficiency.

The time taken was substantially faster for both parallelised MerDel and FedMerDel as expected via distributed computing, where MerDel on the full dataset became infeasible as our data sizes became larger due to the computational requirements of manipulating large matrices. However, as hypothesised in Section 4.1, we saw that FedMerDel was faster than parallelised MerDel, and became substantially more so as N increased (Figure 1).

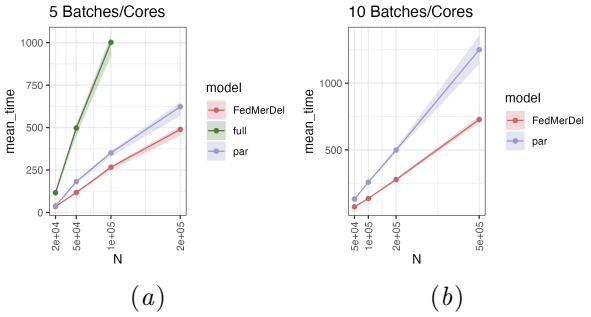


Figure 1: Plot comparing the mean time taken by MerDel with and without parallelisation ('par' and 'full'), and FedMerDel as we vary N with 5 or 10 batches/cores in 'Global Merge Simulations', with an approximate 95% confidence interval (mean $\pm$ 1.96 $\times \frac{\text{s.d}}{\sqrt{n}}$). FedMerDel results use random search.

Global merge times ranged from a mean of 0.29s(greedy)/0.39s(random) for $N = 20,000$ to a mean of 14.8s(greedy)/18.2s(random) for $N = 500,000$, showing that the time taken for the global merge was not computationally prohibitive. Different batches of data for a dataset also provided consistent results, e.g. the standard deviation of ARIs across 10 different shuffles of data in an $N = 50,000$ simulation ranged from 0.0003 to 0.005 for 20 datasets.

FedMerDel outperformed all comparison methods in terms of both accuracy and time efficiency. k-FED performed poorly (ARI between 0-0.2). FedEM involved frequent communication between clients and a central server, and took 1.5-4 hours per model. Local Gibbs sampling in parallel took 20-40 mins; FedMerDel took <3 mins (Table 2). We also compared

Table 1: A subset of results for 'Global Merge Simulations', where data is distributed heterogeneously. For (a), (b), (c), the heterogeneous data scenarios are described in Appendix G.4. We report the median and lower/upper quantiles across all 10 'shuffles'/initialisations and all 10 synthetic datasets. FedMerDel results use random search.

Scenario	Model	Batches/Cores	ARI	Clusters	Time (s)
(a)	par	10	0.945 [0.944, 0.951]	12 [12, 12]	119 [110, 131]
	FedMerDel	10	0.942 [0.936, 0.946]	13 [12, 13]	64.5 [61.5, 70.1]
(b)	par	5	0.955 [0.949, 0.956]	10 [10, 10]	169 [161, 192]
	FedMerDel	5	0.993 [0.992, 0.994]	10 [10, 10]	131 [127, 165]
(c)	par	5	0.950 [0.944, 0.950]	12 [12, 12.8]	69.5 [65.8, 77.3]
	FedMerDel	5	0.988 [0.985, 0.990]	13 [12, 13]	53.2 [48.3, 58.3]

FedMerDel to non-federated clustering approaches including k-modes, hierarchical clustering, DBSCAN, and latent class analysis (LCA; Appendix H.2.1). While the heuristic algorithms were computationally efficient, they produced near-zero ARIs on our simulated datasets, indicating poor recovery of true clustering structures. LCA performed better (mean ARI ≈ 0.87 at $N = 50,000$) but required multiple model fits and longer runtimes ($\sim$10 min per model).

6.1.3. Number of Batches

With datasets of size $N = 100,000$ and $N = 200,000$, we compared the clustering performance when considering different numbers of batches (further details in Appendix G.5). As we increased the number of batches used, although there was no statistically significant difference at the 5% level in the clustering accuracy as measured by ARI, we did see more clusters (Appendix H.3). ARI values were all above 0.93.

6.2. Electronic Health Record (EHR) data

We analysed an anonymised EHR dataset derived from The Health Improvement Network (THIN) database that comprised primary-care records from 289,821 individuals in the United Kingdom over the age of 80. For 94 long-term health conditions, we recorded which conditions were present in each individual to create a $289,821 \times 93$ binary matrix whose (i, j)-entry indicates whether individual i has condition j (encoded as a 1) or not (encoded as 0). Further details on data preprocessing are provided in Appendix J. The aim of the analysis is to identify clusters of individuals with common patterns of co-occurring conditions (multimorbidity) across a population, allowing us to discover clinically meaningful subgroups that support population health research.

We divided the data into 14 batches of size 20,000 and 1 batch of size 9,821 using random partitioning, and ran FedMerDel with $K = 20$ maximum clusters within each batch. After global merging, we obtained 12 clusters, which are summarised in Figure 2. Each cluster can be characterised by particular conditions. From top to bottom in Figure 2, we have clusters of individuals with: (i) cancer; (ii) Atrial fibrilation (AF) & arrhythmia; (iii) peripheral vascular disease & aortic aneurysm; (iv) stroke; (v) AF & arrhythmia and stroke; (vi) stroke and blindness; (vii) blindness; (viii) dementia; (ix) asthma and chronic obstructive pulmonary disease; (x)-(xii) "generally multimorbid" individuals.

To characterise these clusters and quantitatively validate their clinical relevance, we examined the mean number of conditions per person in clusters and cluster-level mortality statistics, which serve as an objective clinical outcome measure (see Appendix J.1). We found that our multimorbidity clusters were associated with different mortality levels, providing quantitative evidence that FedMerDel identified clinically relevant patient subgroups. For example, the stroke + AF/arrhythmia cluster has higher mortality than the stroke-only cluster, highlighting potential differences in clinical risk profiles. Of the "generally multimorbid" clusters, (x) sees a lower mean number of conditions per person and a lower mortality rate compared to the population average, (xi) is associated with psychiatric disorders and lower mortality, while cluster (xii) represents a high-mortality multimorbid group. While there may be further subclusters in cluster (x), we believe these do not emerge due to weak signals compared to core clusters. Future work using recursive clustering approaches could enable the discovery of such finer-grained structure.

To evaluate performance under realistic heterogeneity expected when combining data from different healthcare sites, we simulated heterogeneity in the THIN dataset (details and full results in Appendix J.2). When dementia patients were sepa-

Table 2: ARI results comparing unsupervised FL methods. k-FED is omitted due to poor performance. Where N is stated, observations were assigned to partitions randomly; 'heterogeneous (c)' is described in Appendix G.4. 'Cl'=clients, 'ba'=batches.

Sim	FedEM	Approx SNOB	FedMerDel
N=20000	10 cl: 0.847 [0.798, 0.865] 20 cl: 0.906 [0.849, 0.913]	5 ba: 0.699 [0.695, 0.700]	5 ba: 0.920 [0.912, 0.933]
N=50000	25 cl: 0.839 [0.793, 0.879]	10 ba: 0.836 [0.814, 0.861]	10 ba: 0.951 [0.946, 0.954]
Heterogeneous (c)	10 cl: 0.907 [0.902, 0.913]	5 ba: 0.502 [0.439, 0.632]	5 ba: 0.988 [0.985, 0.990]

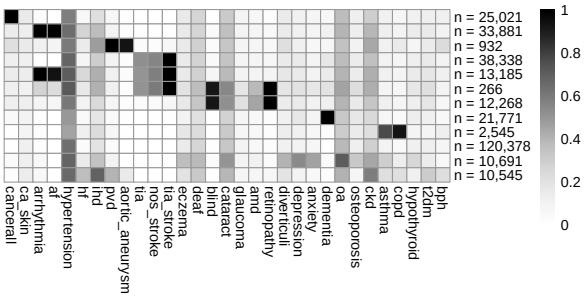


Figure 2: EHR clustering results with random search. Columns are health conditions, and rows correspond to clusters. Shading indicates proportion of individuals in each cluster with each condition; row labels show the number of individuals per cluster. To improve visualisation, only the most prevalent health conditions are shown. See Appendix J for further details, coding of health conditions and greedy search results.

rated into distinct batches, the method revealed subclusters within the dementia group, supporting future work into recursive clustering. Using a Dirichlet distribution to simulate condition-based skews across batches, core clusters were retained, while smaller subclusters emerged, likely driven by batch-specific over-representation. Beyond computational efficiency, this federated approach offers valuable insights into cluster prevalence in batches - in real EHR settings, such patterns could identify region-specific multimorbidity clusters based on different sociodemographic characteristics of local populations. FedMerDel recovers meaningful population-level patterns even in the presence of inter-node variability.

7. Discussion

In this manuscript, we introduced FedMerDel, a Bayesian mixture modelling algorithm using variational inference and 'global' merge moves to enable federated learning (FL) without sharing the full dataset across nodes. This model showed good performance on both simulated and real-world data, with only slight reductions in performance compared to centralised analysis, and excelled when data was not identically distributed across batches, a key challenge in FL. FedMerDel outperformed other existing unsupervised FL methods, where it could directly address both the challenge of scaling model-based clustering, and the challenge of clustering decentralised data. This is valuable for scenarios requiring both data privacy and scalability e.g. patient health records.

There is clearly scope for improvement in some aspects of the model. While we push scalability beyond other probabilistic approaches (eg. MCMC, EM), physical hardware limits remain, with memory use from large parameter sets creating computational bottlenecks. To scale further, we could add extra steps in the global merge as in the SIGN model (Ni et al., 2019), where local clusters are combined over multiple steps over multiple computing nodes or cores. Efficiency could be improved by adapting the 'laps' parameter controlling the frequency of merge/delete moves as MerDel approaches convergence, similar to learning rate scheduling. Future work could explore allowing local cluster splits during global merging (Song et al. (2020) represents local clusters as a mixture of distributions) and theoretical convergence guarantees for global merge moves. Further VI techniques including Stochastic Variational Inference (SVI) (Hoffman et al., 2013) and variational tempering (Mandt et al., 2016) could also be implemented within each data batch to further improve scalability and accuracy of the local clustering.

While we presented our approach in the context of modelling large categorical datasets, we note that it can be straightforwardly adapted to mixtures of Gaussians and other members of the exponential family for other applications. The use of sufficient statistics to allow for efficient ELBO calculation could extend to any exponential family distribution; Hughes and Sudderth (2013) provides further details.

Acknowledgments

The project was part of the Bringing Innovative Research Methods to Clustering Analysis of Multimorbidity (BIRM-CAM) project that was funded by the MRC (MR/S027602/1). We also acknowledge MC_UU_00040/05.

References

Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized Federated Learning With Gaussian Processes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8392–8406. Curran Associates, Inc., 2021.

Shrey Bhatt, Aishwarya Gupta, and Piyush Rai. Federated Learning with Uncertainty via Distilled Predictive Distributions. In Berrin Yanıkoğlu and Wray Buntine, editors, *Proceedings of the 15th Asian Conference on Machine Learning*, volume 222 of *Proceedings of Machine Learning Research*, pages 153–168. PMLR, 11–14 Nov 2024.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, 1 edition, August 2006.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017. ISSN 1537-274X. doi: 10.1080/01621459.2017.1285773.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-IID data. *arXiv*, 2020. doi: 10.48550/ARXIV.2004.11791.

Trevor Campbell, Julian Straub, John W Fisher III, and Jonathan P How. Streaming, distributed variational inference for bayesian nonparametrics. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Longbing Cao, Hui Chen, Xuhui Fan, Joao Gama, Yew-Soon Ong, and Vipin Kumar. Bayesian Federated Learning: A Survey. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 7233–7242. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/851. Survey Track.

Gilles Celeux, Merrilee Hurn, and Christian P. Robert. Computational and Inferential Difficulties with Mixture Posterior Distributions. *Journal of the American Statistical Association*, 95(451): 957–970, September 2000. ISSN 1537-274X. doi: 10.1080/01621459.2000.10474285.

Anil Chaturvedi, Paul E. Green, and J. Douglas Caroll. K-modes Clustering. *Journal of Classification*, 18(1):35–55, January 2001. ISSN 1432-1343. doi: 10.1007/s00357-001-0004-3.

John E Cornell, Jacqueline A Pugh, John W Williams, Jr, Lewis Kazis, Austin F S Lee, Michael L Parchman, John Zeber, Thomas Pederson, Kelly A Montgomery, and Polly Hitchcock Noël. Multimorbidity clusters: Clustering binary data from a large administrative medical database. *Appl. Multivar. Res.*, 12(3):163, January 2009.

David B. Dahl and Spencer Newcomb. Sequentially allocated merge-split samplers for conjugate Bayesian nonparametric models. *Journal of Statistical Computation and Simulation*, 92(7): 1487–1511, November 2021. ISSN 1563-5163. doi: 10.1080/00949655.2021.1998502.

Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 18–24 Jul 2021.

Aymeric Dieuleveut, Gersende Fort, Eric Moulines, and Geneviève Robin. Federated-EM with heterogeneity mitigation and variance reduction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29553–29566. Curran Associates, Inc., 2021.

Michael D. Escobar and Mike West. Bayesian Density Estimation and Inference Using Mixtures. *Journal*

of the American Statistical Association, 90(430): 577–588, June 1995. ISSN 1537-274X. doi: 10.1080/01621459.1995.10476550.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

Swier Garst and Marcel Reinders. Federated K-Means Clustering. In Apostolos Antonacopoulos, Subhasis Chaudhuri, Rama Chellappa, Cheng-Lin Liu, Saumik Bhattacharya, and Umapada Pal, editors, *Pattern Recognition*, pages 107–122, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-78166-7.

Hong Ge, Yutian Chen, Moquan Wan, and Zoubin Ghahramani. Distributed Inference for Dirichlet Process Mixture Models. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2276–2284, Lille, France, 07–09 Jul 2015. PMLR.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 14 (40):1303–1347, 2013.

Zaijing Huang and Andrew Gelman. Sampling for Bayesian Computation with Large Datasets. *SSRN Electronic Journal*, 2005. ISSN 1556-5068. doi: 10.2139/ssrn.1010107.

Michael Hughes, Dae Il Kim, and Erik Sudderth. Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 370–378. PMLR, 09–12 May 2015.

Michael C Hughes and Erik Sudderth. Memoized Online Variational Inference for Dirichlet Process Mixture Models. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Sonia Jain and Radford M. Neal. A Split-Merge Markov Chain Monte Carlo Procedure for the Dirichlet Process Mixture Model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004. ISSN 1061-8600.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawit, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. *Advances and Open Problems in Federated Learning*. Now Foundations and Trends, 2021.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109.

Hemant H Kumar, Karthik V R, and Mydhili K Nair. Federated K-Means Clustering: A Novel Edge AI Based Approach for Privacy Preservation. In *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 52–56, 2020. doi: 10.1109/CCEM50674.2020.00021.

M.H.C. Law, M.A.T. Figueiredo, and A.K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9): 1154–1166, September 2004. ISSN 0162-8828. doi: 10.1109/tpami.2004.71.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST Database. https://yann.lecun.com/exdb/mnist/, 2011. Accessed 2024-09-25.

Friedrich Leisch. FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R. *Journal of Statistical Software*, 11(8), 2004. ISSN 1548-7660. doi: 10.18637/jss.v011.i08.

Dahua Lin. Online Learning of Nonparametric Mixture Models via Sequential Variational Approximation. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Drew A. Linzer and Jeffrey B. Lewis. poLCA: An R Package for Polytomous Variable Latent Class Analysis. *Journal of Statistical Software*, 42(10), 2011. ISSN 1548-7660. doi: 10.18637/jss.v042.i10.

Yang Liu, Zhuo Ma, Zheng Yan, Zhuzhu Wang, Ximeng Liu, and Jianfeng Ma. Privacy-preserving federated k-means for proactive caching in next generation cellular networks. *Information Sciences*, 521:14–31, 2020. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2020.02.042.

Silvia Liverani, David I. Hastie, Lamiae Azizi, Michail Papathomas, and Sylvia Richardson. PReMiuM: An R Package for Profile Regression Mixture Models Using Dirichlet Processes. *Journal of Statistical Software*, 64(7), 2015. ISSN 1548-7660. doi: 10.18637/jss.v064.i07.

Steven N. Maceachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics - Simulation and Computation*, 23(3):727–741, January 1994. ISSN 1532-4141. doi: 10.1080/03610919408813196.

Gertraud Malsiner-Walli, Sylvia Frühwirth-Schnatter, and Bettina Grün. Model-based clustering based on sparse finite Gaussian mixtures. *Statistics and Computing*, 26(1–2): 303–324, August 2014. ISSN 1573-1375. doi: 10.1007/s11222-014-9500-2.

Stephan Mandt, James McInerney, Farhan Abrol, Rajesh Ranganath, and David Blei. Variational tempering. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 704–712, Cadiz, Spain, 09–11 May 2016. PMLR.

Vikash Mansinghka, Patrick Shafto, Eric Jonas, Cap Petschulat, Max Gasner, and Joshua B. Tenenbaum. CrossCat: A Fully Bayesian Nonparametric Method for Analyzing Heterogeneous, High Dimensional Data. *Journal of Machine Learning Research*, 17(138):1–49, 2016.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, December 2011. ISSN 1942-4795. doi: 10.1002/widm.53.

Ramesh Nallapati, William Cohen, and John Lafferty. Parallelized Variational EM for Latent Dirichlet Allocation: An Experimental Evaluation of Speed and Scalability. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, page 349–354. IEEE, October 2007. doi: 10.1109/icdmw.2007.33.

Radford M. Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000. ISSN 10618600.

Willie Neiswanger, Chong Wang, and Eric Xing. Embarrassingly Parallel Variational Inference in Nonconjugate Models. *arXiv preprint arXiv:1510.04163*, 2015.

Shu Kay Ng, Richard Tawiah, Michael Sawyer, and Paul Scuffham. Patterns of multimorbid health conditions: a systematic review of analytical methods and comparison analysis. *Int. J. Epidemiol.*, 47 (5):1687–1704, October 2018.

Yang Ni, Peter Müller, Maurice Diesendruck, Sinead Williamson, Yitan Zhu, and Yuan Ji. Scalable Bayesian Nonparametric Clustering and Classification. *Journal of Computational and Graphical Statistics*, 29(1):53–65, July 2019. ISSN 1537-2715. doi: 10.1080/10618600.2019.1624366.

Yang Ni, Yuan Ji, and Peter Müller. Consensus Monte Carlo for Random Subsets Using Shared Anchors. *Journal of Computational and Graphical Statistics*, 29(4):703–714, April 2020. ISSN 1537-2715. doi: 10.1080/10618600.2020.1737085.

Linda Nichols, Tom Taverner, Francesca Crowe, Sylvia Richardson, Christopher Yau, Steven Kiddle, Paul Kirk, Jessica Barrett, Krishnarajah Nirantharakumar, Simon Griffin, Duncan Edwards, and Tom Marshall. In simulated data and health records, latent class analysis was the optimum multimorbidity clustering algorithm. *J. Clin. Epidemiol.*, 152:164–175, December 2022.

Witold Pedrycz. Federated FCM: Clustering Under Privacy Requirements. *IEEE Transactions on Fuzzy Systems*, 30(8):3384–3388, August 2022. ISSN 1941-0034. doi: 10.1109/tfuzz.2021.3105193.

Maxim Rabinovich, Elaine Angelino, and Michael I Jordan. Variational Consensus Monte Carlo. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Jackie Rao and Paul D. W. Kirk. VICatMix: variational Bayesian clustering and variable selection for discrete biomedical data. *Bioinformatics Advances*, 5(1), December 2024. ISSN 2635-0041. doi: 10.1093/bioadv/vbaf055.

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer New York, 2004. ISBN 9781475741452. doi: 10.1007/978-1-4757-4145-2.

Judith Rousseau and Kerrie Mengersen. Asymptotic Behaviour of the Posterior Distribution in Overfitted Mixture Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73 (5):689–710, August 2011. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2011.00781.x.

Felix Sattler, Klaus-Robert Muller, and Wojciech Samek. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization Under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8): 3710–3722, August 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.3015958.

Richard S Savage, Zoubin Ghahramani, Jim E Griffin, Paul D. W. Kirk, and David L Wild. Identifying cancer subtypes in glioblastoma by combining genomic, transcriptomic and epigenomic data. *arXiv preprint arXiv:1304.3577*, 2013.

Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and big data: the consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, February 2016. ISSN 1750-9661. doi: 10.1080/17509653.2016.1142191.

Hanyu Song, Yingjian Wang, and David B Dunson. Distributed Bayesian clustering using finite mixture of mixtures. *arXiv*, March 2020.

Morris Stallmann and Anna Wilbik. Towards Federated Clustering: A Federated Fuzzy c-Means Algorithm (FFCM). *arXiv*, 2022. doi: 10.48550/ARXIV.2201.07316.

Mahlet G Tadesse, Naijun Sha, and Marina Vannucci. Bayesian Variable Selection in Clustering High-Dimensional Data. *Journal of the American Statistical Association*, 100(470):602–617, June 2005. ISSN 1537-274X. doi: 10.1198/016214504000001565.

Ye Tian, Haolei Weng, and Yang Feng. Towards the Theory of Unsupervised Federated Learning: Non-asymptotic Analysis of Federated EM algorithms. In *Forty-first International Conference on Machine Learning*, 2024.

Naonori Ueda and Zoubin Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15(10): 1223–1241, December 2002. ISSN 0893-6080. doi: 10.1016/s0893-6080(02)00040-0.

Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E. Hinton. SMEM Algorithm for Mixture Models. *Neural Computation*, 12(9):2109–2128, September 2000. ISSN 0899-7667, 1530-888X. doi: 10.1162/089976600300015088.

Zoé van Havre, Nicole White, Judith Rousseau, and Kerrie Mengersen. Overfitting Bayesian Mixture Models with an Unknown Number of Components. *PLOS ONE*, 10(7):e0131739, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0131739.

Stephen G. Walker. Sampling the Dirichlet Mixture Model with Slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, January 2007. ISSN 1532-4141. doi: 10.1080/03610910601096262.

Yue Wu, Shuaicheng Zhang, Wenchao Yu, Yanchi Liu, Quanquan Gu, Dawei Zhou, Haifeng Chen, and Wei Cheng. Personalized Federated Learning under Mixture of Distributions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 37860–37879. PMLR, 23–29 Jul 2023.

Yang Yang, Bo Chen, and Hongwei Liu. Memorized Variational Continual Learning for Dirichlet Process Mixtures. *IEEE Access*, 7:150851–150862, 2019. ISSN 2169-3536. doi: 10.1109/access.2019.2947722.

Jiong Zhang, Parameswaran Raman, Shihao Ji, Hsiang-Fu Yu, S.V.N. Vishwanathan, and Inderjit Dhillon. Extreme stochastic variational inference: Distributed inference for large scale mixture models. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 935–943. PMLR, 16–18 Apr 2019.

Xu Zhang, Wenpeng Li, Yunfeng Shao, and Yinchuan Li. Federated Learning via Variational Bayesian Inference: Personalization, Sparsity and Clustering. *arXiv*, 2023. doi: 10.48550/ARXIV.2303.04345.

Jun Zhu, Jianfei Chen, Wenbo Hu, and Bo Zhang. Big Learning with Bayesian methods. *National Science Review*, 4(4):627–651, 05 2017. ISSN 2095-5138. doi: 10.1093/nsr/nwx044.

Daiane Aparecida Zuanetti, Peter Müller, Yitan Zhu, Shengjie Yang, and Yuan Ji. Bayesian nonparametric clustering for large data sets. *Statistics and Computing*, 29(2):203–215, February 2018. ISSN 1573-1375. doi: 10.1007/s11222-018-9803-9.

Appendix A. Variable Selection

As explored in Law et al. (2004) and Tadesse et al. (2005), we can introduce binary feature selection indicators γ_j - feature saliencies - where $\gamma_j = 1$ if and only if the j-th covariate is relevant to the clustering structure, and irrelevant variables have their feature saliencies reduced to zero. The probability density for a data point x_n in cluster k is given by:

$$f(\mathbf{x}_n|\Phi_k) = \prod_{j=1}^{P} f_j(x_{nj}|\Phi_{kj})^{\gamma_j} f_j(x_{nj}|\Phi_{0j})^{1-\gamma_j} \tag{14}$$

$\Phi_{0j} = [\phi_{0j1}, ..., \phi_{0jL_j}]$ are parameter estimates obtained for covariate j under the assumption that there exists no clustering structure in the j-th covariate. These are precomputed using maximum likelihood estimates as seen in Savage et al. (2013). The priors associated with γ_j for $j = 1, ..., P$ are as follows:

$$\gamma_j|\delta_j \sim \text{Bernoulli}(\delta_j) \tag{15}$$

$$\delta_j \sim \text{Beta}(a, a) \tag{16}$$

These irrelevant variables can correspond to noisy/corrupted features, which is particularly useful in real-world cases where we believe there may be data corruption.

Appendix B. Variational Updates with Variable Selection

'E' Step This follows a similar form to the model with no variable selection, where Z takes a multinomial distribution.

$$q^*(Z) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}}, \qquad r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K}\rho_{nj}} \tag{17}$$

$$\ln \rho_{nk} = \mathbb{E}_\pi[\ln \pi_k] + \sum_{j=1}^{P} c_j \mathbb{E}_\Phi[\ln \phi_{kjx_{nj}}] + (1 - c_j)(\ln \phi_{0jx_{nj}}) \tag{18}$$

$c_j = \mathbb{E}_\gamma(\gamma_j)$, where the expectation is taken over the variational distribution for γ.

'M' Step Similarly to the model with no variable selection, the updates for the cluster-specific parameters are given by:

$$q^*(\pi) = \text{Dirichlet}(\alpha_1^*, ..., \alpha_K^*) \tag{19}$$

$$q^*(\phi) = \prod_{k=1}^{K}\prod_{j=1}^{P} \text{Dirichlet}(\epsilon_{kj1}^*, ..., \epsilon_{kjL_j}^*) \tag{20}$$

where for $k = 1, ..., K, j = 1, ..., P, l = 1, ..., L_j$:

$$\alpha_k^* = \alpha_k + \sum_{n=1}^{N} r_{nk}, \qquad \epsilon_{kjl}^* = \epsilon_j + \sum_{n=1}^{N} \mathbb{I}(x_{nj} = l) r_{nk} c_j \tag{21}$$

The updates for γ and δ are given as:

$$q^*(\gamma_j) = \text{Bernoulli}(c_j), \qquad c_j = \frac{\eta_{1j}}{\eta_{1j} + \eta_{2j}} = \mathbb{E}_\gamma(\gamma_j) \tag{22}$$

$$\ln \eta_{1i} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk} \mathbb{E}_{\Phi}[\ln \phi_{kjx_{nj}}]) + \mathbb{E}_{\delta}[\ln \delta_j] \tag{23}$$

$$\ln \eta_{2i} = \sum_{n=1}^{N} \sum_{k=1}^{K} (r_{nk} \ln \phi_{0jx_{nj}}) + \mathbb{E}_{\delta}[\ln(1-\delta_j)] \tag{24}$$

$$q^*(\delta_j) = \text{Beta}(c_j + a, 1 - c_j + a) \tag{25}$$

As before, all expectations throughout are taken over the variational distributions for each parameter and we alternate between 'E' and 'M' steps.

Appendix C. Merge and Delete Moves: Further Detail

C.1. Merge Moves

Merge moves refer to moves that combine observations from two clusters into one unified cluster. Suppose clusters k_1 and k_2 are considered for a merge. We propose a move to reassign all observations from clusters k_1 and k_2 into a single new cluster, k^*. The responsibilities for this new cluster are given by $r_{nk^*} = r_{nk_1} + r_{nk_2}$. All other responsibilities for other clusters remain the same.

We then perform a 'dummy' variational M step, updating the π, ϕ parameters. We run an extra E and an extra M step to allow observations to move in or out of the new cluster. We calculate the ELBO and compare this to the ELBO before the merge move; we accept this move if the ELBO has improved, otherwise we return to the model prior to the merge. If accepted, we reassign cluster k^* to the k_1 position and delete cluster k_2 from the model.

C.1.1. Selecting Candidate Clusters to Merge

We could choose a pair of clusters for a merge at random, but a merge is more likely to be accepted if the two clusters are 'similar'. Making informed decisions on which pair of clusters to propose for a merge can improve efficiency by reducing the number of rejected merge moves. Two methods of assessing similarity between clusters using correlation and divergences between variational distributions are detailed below. Other ways include comparing marginal likelihoods or posterior probabilities of candidate models as in Hughes et al. (2015) and Ueda et al. (2000).

Due to the stochastic nature of the selection from the 'candidate clusters', similar results between different merge candidates are to be expected and the choice of measure is insignificant. Results from a simulation study comparing selection criteria for clusters to be merged are seen in Appendix H.4. Correlation and divergence-based methods performed similarly, but did provide a reduction in overall computation time and an improvement in rates of accepted merges compared to random selection.

Correlation We can use an assessment of correlation between the variational parameters associated with clusters; a notion of correlation is seen also in Hughes and Sudderth (2013).

We construct a correlation score by looking at the correlation between the parameter values ϵ^* in the distributions $q(\phi_{kj})$. In the binary case, each ϵ^*_{kj} is given by a two dimensional vector. For each k, we can take the first value of each of these vectors for $j = 1, ..., P$, concatenate these into one j length vector, ϵ^{corr}_k and then compare the correlation of these concatenated vectors across clusters. The ϵ^{corr}_k values represent the (expected) frequency of 0's in each cluster across all variables in a binary dataset of 0's and 1's.

We select randomly between the 3 cluster pairs with the highest correlation, provided $\text{Corr}(\epsilon^{corr}_{k_1}, \epsilon^{corr}_{k_2})$ is above 0.05. With the categorical case with L categories per variable, we could consider finding the correlation scores for $L-1$ concatenated vectors across all variables for $L-1$ of the categories and taking the mean correlation score. However, this is complicated when variables have different numbers of categories.

Divergence-based Measures We also look at using measures of distance between probability distributions to quantify cluster similarity. For a pair of clusters k_1 and k_2, we look at the divergence between $q(\phi_{k_1 j})$ and $q(\phi_{k_2 j})$ and sum over all variables j for a pair of clusters k_1 and k_2. We then eg. choose randomly between the 3 cluster pairs with the lowest divergences. Many divergences could be considered; in this paper we considered the Kullback-Leibler (KL) divergence and the Bhattacharyya distance. We emphasise that we are not seeking a precise characterisation of the 'distance' but are only making approximate comparisons.

C.2. Delete Moves

Delete moves refer to moves to delete unnecessary extra clusters in the model. Redundant small clusters often remain at local optima in variational algorithms for mixture models.

Let cluster k be a candidate cluster for deletion. For the set of observations in cluster k, ie. $S_k := \{n = 1 \dots N : z_{nk} = 1\}$, we create a new data-frame X_{-k} with the rows from observations in S_k removed, and the column for cluster k removed. We first perform the variational E step after removing all variational parameters relevant to cluster k in the model. This recalculates all responsibilities for the remaining observations. We then perform the variational M step, which updates cluster-specific parameters.

We then return to the original dataframe X and then run an E step using X with the current model parameters, which reassigns all observations in X to new clusters. After another M step, we then calculate the value of the ELBO function. We accept the delete move if the new ELBO is higher.

Note that in both the merge and delete moves described above, there is a small approximation involved in setting $r_{nk} = 0$ exactly for responsibilities associated with clusters 'removed'. Appendix E gives further detail; this approximation simply replaces infinitesimally small numbers (smaller than 10^{-80}) with 0.

C.2.1. Selecting Candidate Clusters to Delete

Smaller clusters are likely to be better candidates for deletion as it is likely these observations have been 'left out'. Extremely small clusters are also less informative in practical applications and may be less reproducible in other similar independent datasets. In this paper, we propose a cluster chosen randomly between all clusters which are smaller than 5% of the dataset; if there are none, we choose randomly between the three smallest clusters.

Appendix D. Global Merge - Further Details

D.1. Random and Greedy Search

Greedy Search We prevent merges between two clusters from the same batch with a greedy search, on the hypothesis that such merges would already have taken place at the local stage. This can be carried out in a sequential manner by examining the correlation/divergences between Cluster 1 of Batch 1 and all clusters in Batch 2. Once Cluster 1 has merged with any cluster in Batch 2, we move to Batch 3 and consider merging with the clusters in this batch. We repeat for all batches, before moving to Cluster 2 of Batch 1 and so on until we have gone through all clusters in Batch 1. We then repeat the process with Batches 2, 3, ..., B, looking only at Batches $b+1, ..., B$ for Batch b.

Random Search Similarly to local merge moves, we calculate correlations/divergences between all K clusters, and pick the merge candidate pair randomly between the 3 pairs with the highest correlation (provided correlation > 0.05) or lowest divergence. There is no restriction on proposing merges between two clusters of the same batch. This stops after a fixed number of global merge moves (e.g. 10) have been rejected consecutively.

D.2. Global ELBO Calculation

For the model with no variable selection, the ELBO for the full dataset is given by:

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{Z,\pi,\Phi}[\ln p(X,Z,\pi,\Phi)] - \mathbb{E}_{Z,\pi,\Phi}[\ln q(Z,\pi,\Phi)] \\ &= \mathbb{E}_{Z,\Phi}[\ln p(X|Z,\Phi)] + \mathbb{E}_{Z,\pi}[\ln p(Z|\pi)] + \mathbb{E}_{\pi}[\ln p(\pi)] + \mathbb{E}_{\Phi}[\ln p(\Phi)] \\ &\quad - \mathbb{E}_{Z}[\ln q(Z)] - \mathbb{E}_{\pi}[\ln q(\pi)] - \mathbb{E}_{\Phi}[\ln q(\Phi)]\end{aligned} \tag{26}$$

where all expectations are taken with respect to the variational distributions for Z, π, Φ. We use the current α^* and ϵ^* values to calculate the values of sufficient statistics:

$$T_k = \sum_{n=1}^{N} r_{nk} = \alpha_k^* - \alpha_0 \tag{27}$$

$$S_{kjl} = \sum_{n=1}^{N} r_{nk}\mathbb{I}(x_{nj} = l) = \epsilon_{kjl}^* - \epsilon_{jl} \tag{28}$$

Each term in Equation (27) is as follows:

$$\mathbb{E}_{Z,\Phi}[\ln p(X|Z,\Phi)] = \mathbb{E}_{Z,\Phi}[\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}(\sum_{j=1}^{P} \ln \phi_{kjx_{nj}})] = \sum_{k=1}^{K}\sum_{j=1}^{P}\sum_{l=1}^{L_j} S_{kjl}\mathbb{E}_{\Phi}[\ln \phi_{kjl}] \tag{29}$$

$$\mathbb{E}_{Z,\pi}[\ln p(Z|\pi)] = \mathbb{E}_{Z,\pi}[\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln \pi_k] = \sum_{k=1}^{K} T_k \mathbb{E}_{\pi}[\ln \pi_k] \tag{30}$$

$$\mathbb{E}_{\pi}[\ln p(\pi)] = -\ln B(\alpha) + \sum_{k=1}^{K}(\alpha_k - 1)\mathbb{E}_{\pi}[\ln \pi_k] \tag{31}$$

$$\mathbb{E}_{\Phi}[\ln p(\Phi)] = \sum_{k=1}^{K}\sum_{j=1}^{P}(-\ln B(\epsilon_{kj}) + \sum_{l=1}^{L_j}(\epsilon_{kjl} - 1)\mathbb{E}_{\Phi}[\ln \phi_{kjl}]) \tag{32}$$

$$\mathbb{E}_{Z}[\ln q(Z)] = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \ln r_{nk} \qquad \text{(assignment entropy)} \tag{33}$$

$$\mathbb{E}_{\pi}[\ln q(\pi)] = -\ln B(\alpha^*) + \sum_{k=1}^{K}(\alpha_k^* - 1)\mathbb{E}_{\pi}[\ln \pi_k] \tag{34}$$

$$\mathbb{E}_{\Phi}[\ln q(\Phi)] = \sum_{k=1}^{K}\sum_{j=1}^{P}(-\ln B(\epsilon_{kj}^*) + \sum_{l=1}^{L_j}(\epsilon_{kjl}^* - 1)\mathbb{E}_{\Phi}[\ln \phi_{kjl}]) \tag{35}$$

Apart from the assignment entropy term in Equation (33), as we have calculated T_k and S_{kjl} via simple linear operations on our model parameters, there are no sums over N and all terms are mainly sums over much smaller K, P, L_j. Furthermore, some terms in $B(\alpha)$, $B(\alpha^*)$, $B(\epsilon_{kj})$, $B(\epsilon_{kj}^*)$ which cancel out. Importantly, X appears nowhere in the calculations; our global merge move allows us to seek a global clustering structure without any full scan over the $N \times P$ matrix X.

D.3. Global Merge - Variable Selection

When finding a global clustering structure in the model with variable selection, our focus is on updating cluster allocation via merging local clusters, so we use the same global ELBO function as in the model without variable selection and only update Z, π, ϕ. Implementing this amounts to a small approximation in the true ELBO for the true Bayesian model. We consider a variable to be 'selected' based on the percentage

of times a variable was selected across the batches. In later simulations, we consider a variable selected if it was selected in either all batches, or all but one batch. This borrows ideas of 'thresholds' for variable selection across multiple VI runs, seen in Rao and Kirk (2024).

Appendix E. Approximations and Priors in MerDel

In Appendix C, we detailed that in the process of performing our merge/delete moves, we 'remove' the clusters by eliminating their parameters and responsibilities from the model code. This could raise concerns pertaining to transdimensional inference. However, we implicitly account for the removed clusters in subsequent calculations. We call these removed clusters 'zombie clusters'.

This is because the cluster-specific parameters α, ϵ are determined by the prior only (as there is no data assigned to the cluster to update cluster-specific parameters). We can accurately calculate ELBO values and track these zombie clusters throughout the algorithm. The key approximation is setting responsibilities for zombie clusters to exactly 0. These values are not usually precisely 0, but are infinitesimally small.

E.1. Emptying a Cluster + M Update

When a cluster k^* is removed (through merge or delete), this is equivalent to assigning $r_{nk^*} = 0$ for all observations n, effectively zeroing column k^* in the $N \times K_{\text{init}}$ responsibility matrix. The subsequent variational M step updates α^*, ϵ^*:

$$\alpha_k^* = \alpha_0 + \sum_{n=1}^{N} r_{nk}, \qquad \epsilon_{kjl}^* = \epsilon_{jl} + \sum_{n=1}^{N} \mathbb{I}(x_{nj} = l) r_{nk} \tag{36}$$

As we have that $r_{nk} = 0$ for all $n = 1, ..., N$ for a removed cluster k, $\alpha_k^* = \alpha_0$ and $\epsilon_{kjl}^* = \epsilon_{jl}$. These variational posteriors for cluster k are solely ruled by the priors for π and Φ.

E.2. The Approximation: E Update

In MerDel, we then perform a variational E update with the full dataset to allow observations to move between clusters after cluster-specific parameter updates. Recall that responsibilities r_{nk} are defined by:

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}}, \qquad \ln \rho_{nk} = \mathbb{E}_{\pi}[\ln \pi_k] + \sum_{j=1}^{P} \mathbb{E}_{\Phi}[\ln \phi_{kjx_{nj}}] \tag{37}$$

The expectations in $\ln \rho_{nk}$ are:

$$\mathbb{E}_{\pi}[\ln \pi_k] = \psi(\alpha_k^*) - \psi(\sum_{k=1}^{K} \alpha_k^*) \tag{38}$$

$$\mathbb{E}_{\Phi}[\ln \phi_{kjx_{nj}}] = \psi(\epsilon_{kjx_{nj}}^*) - \psi(\sum_{l=1}^{L_j} \epsilon_{kjl}^*) \tag{39}$$

Since, for $x > 0$, the digamma function $\psi(x)$ increases approximately exponentially, observations are assigned to clusters with responsibilities close to 1. These clusters therefore have α_k^* values on the scale of the number of observations assigned - sometimes in the thousands - which are much larger than $\alpha_0 < 1$. Therefore $\ln \rho_{nk}$ is much smaller for 'zombie clusters', resulting in r_{nk} values *extremely* close to 0.

These near-zero r_{nk} values persist as we continue through the EM steps; α^* and ϵ^* still remain extremely small for removed clusters (virtually equal to prior values) in comparison to clusters with observations, and r_{nk} values still remain virtually 0. Our approximation sets $r_{nk} = 0$ permanently by removing these columns, improving computational efficiency and memory usage. Since cluster-specific parameters for removed clusters equal their priors when $r_{nk} = 0$, we also remove these.

This approximation has virtually no effect on our model; we run two small simulations comparing the effects of implementing this approximation, and saw that true r_{nk} values were often on the scale of 10^{-70} to 10^{-100} on typical MerDel datasets. After an accepted MerDel proposal, we continue with standard variational EM steps where removed clusters remain permanently eliminated.

This behaviour mirrors the overfitted variational model without merge/delete, where empty clusters naturally maintain near-zero responsibilities and virtually never regain observations. ELBO calculation remains straightforward; terms from zombie clusters either cancel each other out or are trivially calculated.

E.3. Approximations and Priors in FedMerDel

There is no approximation when considering a 'global merge' in FedMerDel, as we never perform any variational E or M steps after performing merges. Responsibilities r_{nk} remain exactly 0 for 'zombie clusters'.

In the global setting, we define a new Dirichlet prior for π over our full dataset with the total number of local clusters initialised: $B \times K_b$, where K_b is the initialised number of clusters in Batch b. This is justified as each cluster has a prior where:

$$p(\pi) \propto \prod_{k=1}^{K} {\pi_k}^{\alpha_0 - 1} \tag{40}$$

with K representing the maximum desired clusters per batch. The maximum clusters globally should be $B \times K_b$, the maximum total clusters when we consider all batches. A fractional prior for each batch would inappropriately constrain the model to K_b clusters across the entire dataset, despite searching K_b clusters per partition. We do not want to constrain the model to force all clusters to merge across batches; in FL, there are often clusters only appearing in certain batches. This is consistent with literature; the same Dirichlet priors for mixing probabilities are used in 'local clusters' and 'global clusters' in (Zuanetti et al., 2018), a similar MCMC model.

Appendix F. Implementation Details

The model builds upon an existing R package, VICatMix (Rao and Kirk, 2024), which applies VI for binary and categorical Bayesian finite mixture models (with no merge/delete moves). Rcpp and RcppArmadillo are used to accelerate computation with C++. MerDel runs on a single-core processor, where, for example, inference for a mixture model fitted to a binary dataset of size $N = 2000$, $P = 100$ generally converges in less than a minute.

FedMerDel allows MerDel to be run independently on a different core per batch of data; reported 'wall-clock times' take this parallelisation into account where clearly stated.

Appendix G. Simulation Set-Up - Additional Information

G.1. Data Generating Mechanism

We simulated synthetic binary data with N observations and P covariates by sampling the probability p of a '1' in each cluster for each variable via a Beta$(1, 5)$ distribution, encouraging sparse probabilities. For noisy variables, the probability of a '1' was also generated by a Beta$(1, 5)$ distribution but this probability was the same for every observation regardless of cluster membership. When simulating categorical data with L categories, we instead used a Dirichlet$(1, ..., L)$ distribution.

G.2. Hyperparameter Choice

In all simulations (real-world and simulated data), we set $\alpha_0 = 0.01$, $\epsilon_j = 1/L_j$ and $a = 2$ (as in Rao and Kirk (2024)) in the priors in Section 2 of the main paper. We did not tune these in this manuscript as all examples are purely illustrative. All algorithms were run with a maximum of 1000 iterations (an iteration being one variational EM step) - although this limit was not reached for any simulation.

Convergence tolerance was set between 0.000005 and 0.00000005 in all simulations. When comparing wall-clock time for convergence, all simulations in a given study had the same convergence tolerance. In all simulations involving FedMerDel, we used the parameters 'laps = 5' for MerDel runs, except when specified.

G.3. Frequency of Merge/Delete Moves

Details of the scenarios for this simulation study are given in Table 3. Simulations 1.1, 1.2, 1.3 were run without variable selection; Simulations 1.4, 1.5 were run with variable selection. K_{init} refers to the initialised value of K in the model. In each scenario, we generated 20 independent datasets and compared 10 different initialisations for each dataset. We used correlation for merge criteria.

Table 3: Table giving parameters for data generation for the first simulation study, 'Frequency of Merge/Delete Moves'.

ID	Relevant Variables	N	P	K_{init}	K_{true}	N per Cluster
1.1	100	1000	60	20	5	100-300
1.2	100	2000	100	20	8	50-800
1.3	100	4000	100	25	10	200-800
1.4	75	1000	100	20	10	50-200
1.5	60	1500	100	20	10	50-200

G.4. Global Merge Simulations

We aimed to evaluate the clustering performance of FedMerDel when splitting a simulated dataset into equally sized batches, and compared this to a parallelised version of MerDel (on the full dataset) as well as the usual MerDel algorithm on the full dataset (variational EM with merge and delete moves). We set 'laps = 5', $P = 100$ covariates, 12 true clusters and 20 initialised clusters in all cases. Correlation was used for all merge criteria. Parameters were kept the same to enable comparison of run-times between different values of N. Details of the values of N tested and number of batches/cores used are given in Table 4.

The full, unparallelised algorithm was only used for the first 4 simulations listed, as it became computationally infeasible for larger values of N with our hardware.

Table 4: Table giving parameters for data generation for the second simulation study, 'Global Merge Simulations', where other parameters are set as in Section G.4.

N	Batches/Cores
20000	5
50000	5
50000	10
100000	5
100000	10
200000	5
200000	10
500000	10

Heterogeneous Data Simulations As the above simulations have observations assigned purely randomly to batches, we also look at situations with heterogeneous (non i.i.d) data in batches, as is common in FL. We consider scenarios where a) 1 of 12 clusters is only seen in one batch and no others with $N = 50,000$, b) 10 clusters are equally split between 5 batches with $N = 50,000$ and c) 10 clusters are equally split between 5 batches, with 2 more clusters which are evenly split across the 5 batches with $N = 20,000$. In all cases, we set $P = 100$. We compared this to parallelised VI on the full data.

Comparisons to Other Unsupervised FL Methods We compared to other unsupervised FL methods for simulated data, including FedEM for a categorical mixture model (Dieuleveut et al., 2021), and an adaptation of k-FED (Dennis et al., 2021) that uses k-modes instead of k-means as appropriate for categorical data. We also compared an approximate version of SNOB (Zuanetti et al., 2018), with local clusters found via Gibbs sampling (with R package PReMiuM (Liverani et al., 2015)), then merged using hierarchical clustering. We looked at randomly distributed simulated data with $N = 20,000$ and $N = 50,0000$, and also looked at heterogeneous data scenario c) in the above paragraph.

Comparisons to Other Unsupervised Centralised Learning Methods To provide additional context for MerDel and FedMerDel's clustering performance, we compared MerDel and FedMerDel to centralized clustering methods suitable for large binary datasets on simulated datasets of size $N = 20,000$ and $N = 50,000$. Methods we looked at were k-modes (Chaturvedi et al., 2001) (recall this is also used for initialisation of MerDel), hierarchical clustering with 'hclust' in R (Murtagh and Contreras, 2011), and DBSCAN (Ester et al., 1996) - in all cases, we fed the algorithm the true number of clusters, 12. We also evaluated latent class analysis (LCA), which previously performed well on similar binary EHR data (Nichols et al., 2022) using the poLCA R package (Linzer and Lewis, 2011).

Varying P We additionally compared the ARI, number of clusters and the time taken for clustering models using parallelised MerDel and FedMerDel when we varied P. In this simulation, we simulated 5 synthetic datasets and ran MerDel and FedMerDel with 5 'shuffles'/initialisations each time. Datasets were all of size $N = 100,000$ with 10 equally sized clusters, and we initialised with 20 clusters in all cases. We split into 5 equally sized batches for FedMerDel, and parallelised over 5 cores for parallelised MerDel.

Less Separable Data To address the challenge where clusters are less easily separated, we also ran simulations where the probability of a '1' was generated by a Beta(2, 2) distribution. Further, we simulated data with this generating distribution and where two clusters - clusters 11 and 12 - had 75% of their covariates with their cluster-specific probabilities being $\pm$ 0.1 of each other. In both of these simulations, we set $N = 50,000$, $P = 100$ and 12 true clusters, initialised with 15 clusters. 5 independent datasets were generated, and FedMerDel was run 3 times on each.

Global Merge with Variable Selection For this simulation with variable selection (on binary data), we employed the 'greedy search' for the global merge, and used correlation to assess similarity between clusters. The two scenarios had simulated data split into 5 equally sized batches with N = 20,000, 50,000, with K = 15 and 10 equally sized true clusters respectively. We set $P = 100$ covariates in both cases, where 80% were relevant to the clustering structure in the first simulation and 75% were relevant in the second simulation. We generated 5 simulated datasets and had 5 different 'shuffles' of the data, as well as running MerDel on the full data 5 times (unparallelised and parallelised), as with other 'global merge' simulations.

As well as looking at ARI, number of resulting clusters and wall-clock time as with other simulations, we looked at the number of selected variables. As described in Section D.3, for FedMerDel, we consider a variable selected if it was selected in either all batches, or all but one batch (where a variable is selected if $c_j > 0.5$, the expectation of the variable selection latent variable (Appendix B)).

G.5. Number of Batches

With datasets of size $N = 100,000$ and $N = 200,000$, we compared the clustering performance when considering $B = 2, 4, 10$ and $B = 4, 8, 20$ batches respectively. Batches were sized equally. We simulated 20 different synthetic datasets with $P = 100$ for each scenario and took 5 different 'shuffles' of the data for every global merge model. Models had 10 unevenly sized 'true' clusters (cluster sizes ranging from 5% to 20% of the dataset) and were initialised with 20 clusters.

Appendix H. Additional Simulation Results

H.1. Frequency of Merge/Delete Moves

Incorporating merge/delete enables faster convergence in terms of wall-clock time: Generally, MerDel models are faster to converge. Figure 5 in Appendix H.1 shows that early accepted merge/delete moves allow for substantial early increases in the ELBO. However, as described in Section 4, there is a trade-off that must be made - models with laps = 1, for example, get close to convergence very fast but then take more time to fully converge. The inclusion of frequent merge/delete moves at the start of the algorithm can also lead to the algorithm jumping to a worse local optimum, reflected in slightly lower ARIs in some cases.

Better ARIs are achieved with merge/delete: Figure 3 and Appendix H.1 show that laps = 2, 5, 10 generally yield higher ARIs, particularly in Simulations 1.1-1.3.

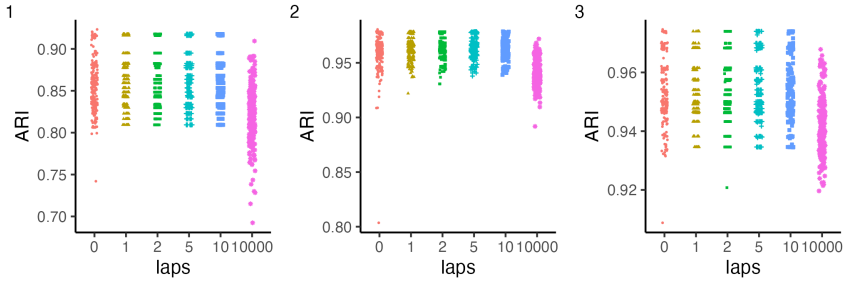


Figure 3: Scatter plot comparing ARIs achieved by each model across all datasets and initialisations in Simulations 1.1, 1.2 and 1.3 (labelled 1, 2, 3 respectively). Each point represents one ARI from one MerDel run.

Merge/delete more accurately estimates the number of clusters: The model with no merge/delete moves tends to vastly overestimate the number of true clusters in the data due to the existence of small unnecessary clusters. MerDel mitigates this, with Figure 4 in Appendix H.1 showing that the model often is able to find the exact true number.

Variational moves are still necessary: The algorithm with purely merge/delete moves performs less well; any result of this algorithm is extremely dependent on initialisation. Observations can never move between two clusters.

Table 5 shows wall-clock time, final log-ELBO and ARI of the resulting clustering structures in Simulations 1.1, 1.2 and 1.3.

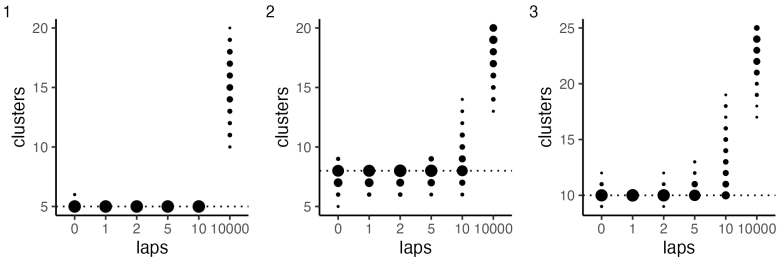
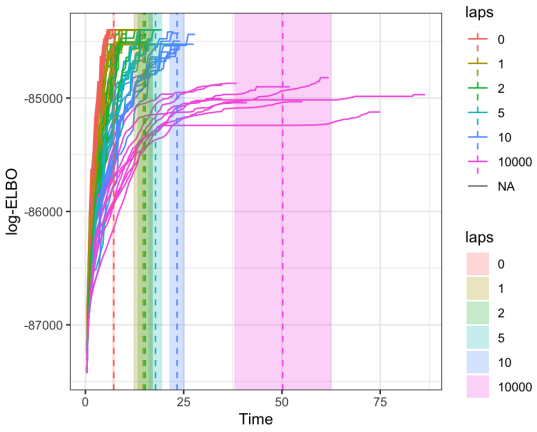


Figure 4: Plot comparing final number of clusters in clustering models in simulations 1.1-1.3.

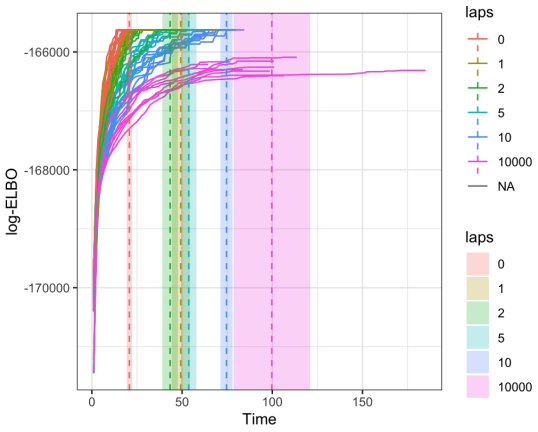
We can clearly see from Figure 5 that a) the original variational model clearly takes far longer than models with merge/delete b) there are steep improvements in ELBO near the start with more frequent merge/delete moves. However, for example, 'laps = 1' is near convergence very early on in both plots, but the algorithm takes longer to actually converge and stop.

Table 5: Performance comparison across Simulations 1.1, 1.2, and 1.3: mean wall-clock time, log-ELBO, ARI and number of clusters with confidence intervals for time and ARI ($\pm 1.96 \times$ standard deviation).

Simulation	Laps	Time (s)	log-ELBO	ARI	Clusters
1.1	0	**3.46 [3.43, 3.49]**	-25357	0.856 [0.852, 0.860]	5.01
	1	6.35 [6.16, 6.55]	**-25354**	**0.858 [0.854, 0.862]**	5.00
	2	5.88 [5.75, 6.01]	**-25354**	**0.858 [0.854, 0.862]**	5.00
	5	6.75 [6.65, 6.85]	**-25354**	**0.858 [0.854, 0.862]**	5.00
	10	8.68 [8.57, 8.80]	**-25354**	**0.858 [0.854, 0.862]**	5.00
	10000	14.0 [13.2, 14.7]	-25617	0.823 [0.818, 0.828]	15.3
1.2	0	**10.9 [10.8, 11.0]**	-81256	0.959 [0.957, 0.961]	7.65
	1	29.1 [26.4, 31.8]	-81245	0.962 [0.960, 0.963]	7.63
	2	23.4 [21.7, 25.1]	-81238	0.962 [0.961, 0.963]	7.70
	5	25.4 [24.1, 26.7]	**-81236**	**0.963 [0.962, 0.964]**	7.84
	10	31.0 [30.0, 32.0]	-81260	0.962 [0.961, 0.963]	8.54
	10000	47.5 [45.0, 50.0]	-81721	0.940 [0.938, 0.942]	18.0
1.3	0	**29.1 [28.8, 29.4]**	-164099	0.953 [0.952, 0.955]	10.0
	1	57.1 [54.8, 59.5]	**-164090**	**0.954 [0.953, 0.955]**	10.0
	2	52.1 [50.7, 53.5]	-164095	0.954 [0.952, 0.955]	10.0
	5	61.8 [60.5, 63.1]	-164100	0.954 [0.952, 0.955]	10.2
	10	76.5 [74.7, 78.3]	-164173	0.953 [0.951, 0.954]	12.0
	10000	122 [112, 113]	-164675	0.943 [0.941, 0.944]	22.7



(a) Simulation 1.2



(b) Simulation 1.3

Figure 5: Graphs showing the ELBO vs wall-clock time for the algorithm for two simulated datasets across all 20 initialisations for each of the different "laps" in Simulations 1.2 and 1.3. The mean end time across all initialisations is shown, as well as an approximate 95% confidence interval.

Another observation we made, especially with Simulation 1.1, is that the variations of MerDel (laps between 1 to 10) frequently identified exactly the same model as one another and achieved higher ARIs than the variational models with no merge/delete moves. This could be indicative that the merge/delete moves are allowing the model to escape local optima and could be reaching a global optimum.

H.1.1. Variable Selection

Table 6 shows wall-clock time, final log-ELBO and ARI of the resulting clustering structures in Simulations 1.4 and 1.5. We still see a speed up in terms of wall-clock time in these simulations, but see that there is more inconsistency with solutions with noisier data, especially as we increase the number of merge/delete moves, and the mean ARI for merge/delete models is generally lower when we incorporate more merge/delete moves. The number of clusters at convergence is also lower than expected. We see from Figure 6 that it seems to be that merge/delete moves lead to less consistent results in the variable selection setting, but most of the best-performing models in terms of ARI are from MerDel models.

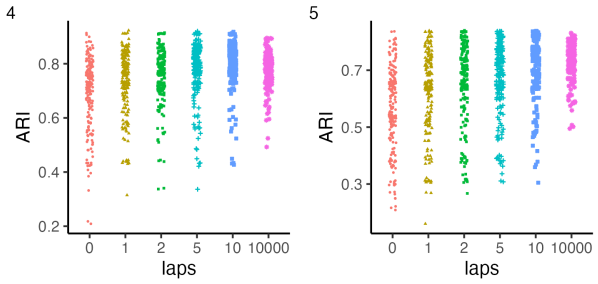


Figure 6: Scatter plot comparing ARIs across all datasets and initialisations in Simulations 1.4 and 1.5.

Table 6: Performance comparison across Simulations 1.4 and 1.5: mean wall-clock time, log-ELBO, ARI and number of clusters with confidence intervals for time and ARI ($\pm 1.96 \times$ standard deviation).

Simulation	Laps	Time (s)	log-ELBO	ARI	Clusters
1.4	0	**16.9 [16.6, 17.2]**	-41913	0.692 [0.674, 0.710]	7.53
	1	91.7 [84.0, 99.3]	-41848	0.731 [0.714, 0.747]	7.86
	2	75.6 [69.8, 81.4]	-41838	0.753 [0.739, 0.767]	8.10
	5	74.4 [70.0, 78.7]	**-41823**	0.773 [0.759, 0.787]	8.48
	10	92.7 [88.4, 96.9]	-41831	**0.792 [0.780, 0.804]**	9.16
	10000	128 [118, 138]	-42372	0.782 [0.774, 0.791]	19.3
1.5	0	**24.0 [23.7, 24.4]**	-61781	0.561 [0.540, 0.581]	6.60
	1	137 [125, 149]	-61664	0.625 [0.605, 0.645]	7.29
	2	110 [101, 118]	-61640	0.651 [0.631, 0.670]	7.67
	5	112 [107, 118]	-61622	0.679 [0.661, 0.696]	8.12
	10	149 [143, 156]	**-61612**	0.710 [0.695, 0.725]	8.90
	10000	213 [199, 227]	-62204	**0.730 [0.721, 0.739]**	19.3

We additionally used F_1 scores to compare the quality of variable selection. Having merge/delete moves tends to improve the quality of the variable selection as quantified by F_1 scores. This is illustrated in Table 7. The gains are made especially in terms of identifying irrelevant, noisy variables, and laps 2 and 5 get the top F_1 scores of 0.977, indicating extremely good feature selection performance.

Table 7: Table comparing the mean F_1 scores, and number of relevant and irrelevant variables successfully found across all simulated datasets and all initialisations in Simulation 1.4 and Simulation 1.5.

	Simulation 1.4			Simulation 1.5		
Laps	Rel	Irrel	F_1 Score	Rel	Irrel	F_1 Score
0	73.4	21.7	0.968	58.8	34.6	0.947
1	73.5	**22.7**	0.975	58.7	**36.0**	0.957
2	73.9	**22.7**	**0.977**	59.2	35.8	**0.959**
5	74.0	22.4	**0.977**	**59.3**	35.5	0.958
10	74.0	22.2	0.975	**59.3**	34.9	0.954
10000	**74.1**	15.5	0.935	**59.3**	23.1	0.872

H.2. Global Merge Simulations - Additional Results

Table 8: 'Global Merge Simulations' results. We report the median and lower/upper quantiles across all 10 'shuffles'/initialisations and all 10 synthetic datasets. FedMerDel/G is the greedy search, FedMerDel/R is the random search for FedMerDel. Note separate independent datasets were generated for N equal but different numbers of batches/cores, accounting for slight differences in ARI. Heterogeneous simulations (a), (b), (c) are described in Appendix G.4.

N /Sim	Model	Batches	ARI	Clusters	Time (s)
20000	full	5	0.943 [0.933, 0.950]	12 [11.8, 12]	106 [92.8, 130]
	par	5	0.943 [0.930, 0.950]	12 [11, 12]	37.0 [33.4, 40.6]
	FedMerDel/G	5	0.920 [0.912, 0.933]	12 [12, 12]	33.1 [28.9, 37.2]
	FedMerDel/R	5	0.920 [0.912, 0.932]	12 [12, 12]	33.2 [29.1, 37.4]
50000	full	5	0.947 [0.941, 0.954]	12 [12, 12]	467 [414, 554]
	par	5	0.948 [0.942, 0.954]	12 [12, 12]	176 [163, 196]
	FedMerDel/G	5	0.945 [0.939, 0.949]	13 [13, 14]	114 [106, 123]
	FedMerDel/R	5	0.945 [0.939, 0.948]	12 [12, 12]	114 [106, 123]
50000	full	10	0.957 [0.955, 0.958]	12 [12, 12]	465 [432, 512]
	par	10	0.956 [0.955, 0.958]	12 [12, 12]	126 [121, 142]
	FedMerDel/G	10	0.951 [0.946, 0.954]	13 [13, 14]	70.6 [66.2, 75.8]
	FedMerDel/R	10	0.951 [0.946, 0.954]	12 [12, 13]	71.3 [66.8, 76.6]
100000	full	5	0.955 [0.944, 0.961]	12 [12, 12]	956 [861, 1116]
	par	5	0.954 [0.944, 0.961]	12 [12, 12]	338 [310, 380]
	FedMerDel/G	5	0.954 [0.943, 0.960]	13 [12, 13]	260 [231, 287]
	FedMerDel/R	5	0.954 [0.943, 0.960]	12 [12, 12]	260 [232, 287]
100000	par	10	0.949 [0.941, 0.959]	12 [12, 12]	249 [231, 284]
	FedMerDel/G	10	0.948 [0.938, 0.952]	13 [13, 14]	131 [123, 143]
	FedMerDel/R	10	0.948 [0.938, 0.952]	12 [12, 13]	132 [124, 144]
200000	par	10	0.952 [0.947, 0.958]	12 [12, 12]	482 [446, 536]
	FedMerDel/G	10	0.951 [0.946, 0.957]	13 [13, 14]	269 [242, 303]
	FedMerDel/R	10	0.951 [0.946, 0.957]	12.5 [12, 13]	270 [243, 305]
500000	par	10	0.950 [0.943, 0.951]	12 [12, 12]	1121 [1056, 1303]
	FedMerDel/G	10	0.949 [0.942, 0.951]	13 [13, 14]	685 [642, 793]
	FedMerDel/R	10	0.949 [0.942, 0.951]	12.5 [12, 13]	688 [646, 797]
1000000	par	20	0.948 [0.943, 0.953]	12 [12, 12]	2119 [2031, 2204]
	FedMerDel/G	20	0.948 [0.943, 0.953]	13.5 [13, 14]	899 [825, 984]
	FedMerDel/R	20	0.948 [0.943, 0.953]	14 [13, 14]	912 [840, 1007]
(a)	par	10	0.945 [0.944, 0.951]	12 [12, 12]	119 [110, 131]
	FedMerDel	10	0.942 [0.936, 0.946]	13 [12, 13]	64.5 [61.5, 70.1]
(b)	par	5	0.955 [0.949, 0.956]	10 [10, 10]	169 [161, 192]
	FedMerDel	5	0.993 [0.992, 0.994]	10 [10, 10]	131 [127, 165]
(c)	par	5	0.950 [0.944, 0.950]	12 [12, 12.8]	69.5 [65.8, 77.3]
	FedMerDel	5	0.988 [0.985, 0.990]	13 [12, 13]	53.2 [48.3, 58.3]

H.2.1. Comparisons to Other Unsupervised Centralised Learning Methods

While k-modes, hierarchical clustering and DBSCAN were fast and clearly superior in terms of computational efficiency (eg. k-modes took around 25 seconds on datasets of size $N = 50,000$, all performed very poorly with ARIs close to 0 compared to the true simulated labels, suggesting these were barely better than random cluster assignment. These methods also face limitations; they often require manual specification of the number of clusters, have limited interpretability and struggle with high dimensionality.

Using the poLCA R package on a simulated dataset with N = 50,000, LCA achieved a mean ARI of 0.867, underperforming both FedMerDel and parallelised MerDel. Furthermore, this required around 10 minutes per model. LCA requires users to fit multiple models and compare with criteria such as BIC to determine the number of clusters. Crucially, this method is not directly applicable to FL scenarios.

H.2.2. Varying P

We saw a similar pattern to N where as P increases, the gains made from FedMerDel are increased compared to parallelised MerDel.

Table 9: 'Global Merge Simulations' results, where all datasets are of size $N = 100,000$ (split into 5 batches/parallelised over 5 cores) and have 10 equally sized clusters, but P (number of covariates) is varied. We compare FedMerDel to parallelised MerDel. We report the median and lower/upper quantiles across 5 'shuffles'/initialisations for 5 synthetic datasets. Results are for the random search; greedy searches differed insignificantly.

N	Model	ARI	Clusters	Time (s)
60	par	0.782 [0.753, 0.796]	10 [10, 10]	206 [194, 231]
	FedMerDel	0.780 [0.751, 0.794]	10 [10, 10]	153 [130, 169]
80	par	0.874 [0.872, 0.880]	10 [10, 10]	285 [256, 333]
	FedMerDel	0.872 [0.870, 0.879]	10 [10, 10]	190 [182, 231]
120	par	0.972 [0.972, 0.973]	10 [10, 10]	316 [292, 345]
	FedMerDel	0.972 [0.971, 0.972]	10 [10, 10]	208 [203, 229]
200	par	0.999 [0.999, 0.999]	10 [10, 10]	465 [453, 469]
	FedMerDel	0.999 [0.999, 0.999]	10 [10, 10]	295 [291, 302]

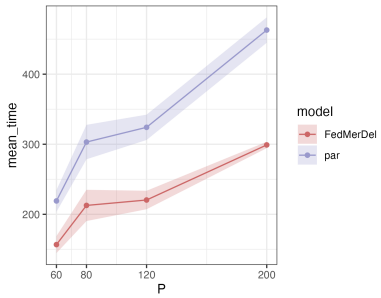


Figure 7: Plot comparing the mean time taken by parallelised MerDel ('par') and FedMerDel ('shard') as we vary P, with a 95% confidence interval (mean $\pm$ 1.96 $\times \frac{s.d}{\sqrt{n}}$). Results for FedMerDel are with the greedy search.

H.2.3. Less Separable Clusters

In both simulation scenarios with the alternative generating distribution to allow for less cluster separability, results still remained robust as before, where ARIs remained close to 0.99. Furthermore, in the second scenario where 75% of covariates were similar across clusters 11 and 12, in all cases, FedMerDel separated the observations from these clusters into distinct clusters.

H.2.4. Global Merge with Variable Selection

Results showed similar performance in terms of ARI between FedMerDel and MerDel on the full dataset (Table 10). In these scenarios (with relatively low N) FedMerDel notably was almost always able to correctly identify relevant and irrelevant variables, outperforming other methods, which usually wrongly identified some irrelevant variables as relevant. Global merges took less than one second in all cases.

H.3. Number of Batches

Results from this simulation study are visualised in Figure 8. A Kruskal-Wallis test testing for any significant difference between the ARI for different numbers of batches for $N = 100,000$ gives a p-value of 0.125. For $N = 200,000$, the p-value is 0.0938. One way to potentially improve inference for the number of clusters for higher values of K_{init} would be to look at higher frequencies of merge/delete moves.

Table 10: 'Global Merge with Variable Selection' simulation results. We report the median and lower/upper quantiles across all 5 'shuffles'/initialisations and all 5 synthetic datasets. Global merge uses the 'greedy search'. 'Rel' and 'Irrel' refer to the number of correctly identified relevant and irrelevant covariates respectively.

N	Model	ARI	Clusters	Time (s)	Rel	Irrel
20000	full	0.865 [0.862, 0.868]	15 [15, 15]	782 [687, 835]	80 [80, 80]	18 [17, 18]
	par	0.865 [0.862, 0.868]	15 [15, 15]	207 [187, 228]	80 [80, 80]	17 [16, 17]
	FedMerDel	0.850 [0.844, 0.857]	15 [15, 15]	223 [212, 242]	80 [80, 80]	20 [20, 20]
50000	full	0.885 [0.876, 0.885]	10 [10, 10]	1607 [1493, 1767]	75 [75, 75]	23 [22, 24]
	par	0.885 [0.876, 0.886]	10 [10, 10]	462 [440, 527]	75 [75, 75]	23 [22, 24]
	FedMerDel	0.881 [0.873, 0.883]	10 [10, 10]	630 [563, 672]	75 [75, 75]	25 [25, 25]

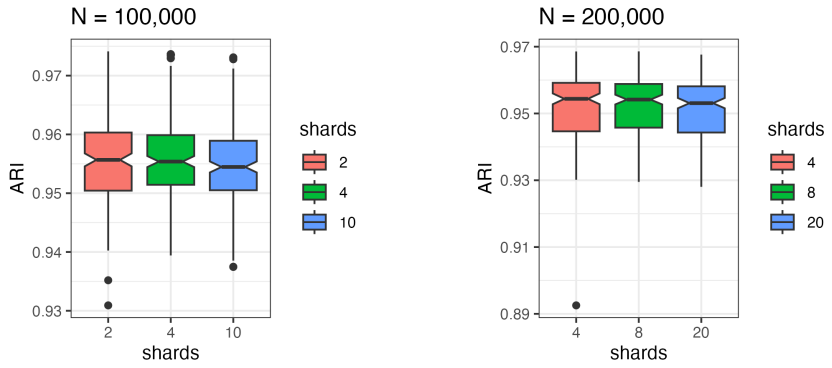


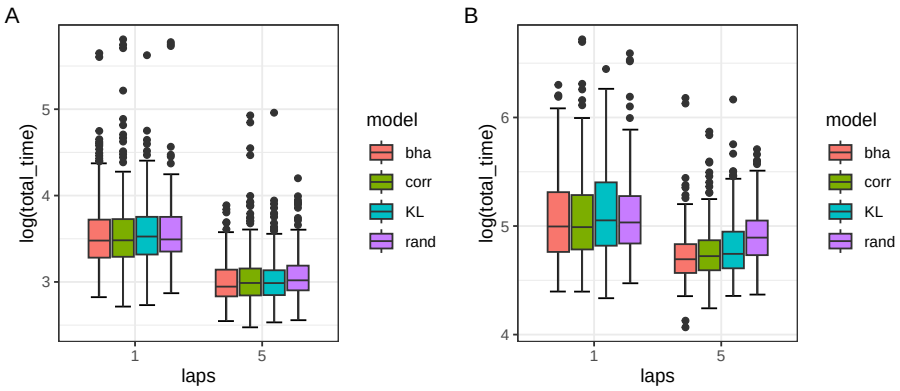
Figure 8: Boxplot showing the distribution of ARIs across different numbers of batches in the 'Number of Batches' simulation.

H.4. Comparing Merge Criteria

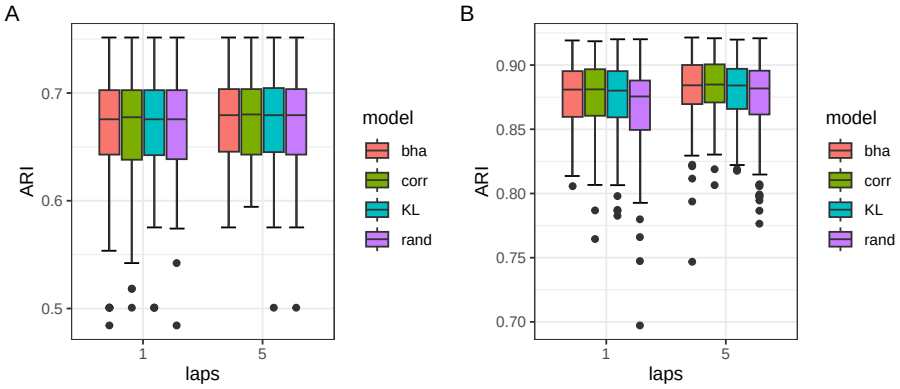
In this simulation, we compared methods for selecting candidate clusters for merging (Section C.1.1): correlation (corr'), Kullback-Leibler divergence (KL'), Bhattacharyya distance (bha'), and random selection (rand').

We ran two studies with $N = 2000$ and different complexity: (A) $K_{\text{true}} = 10$, $K_{\text{init}} = 20$, $P = 50$; (B) $K_{\text{true}} = 20$, $K_{\text{init}} = 40$, $P = 100$ (Figure 9). In both cases, we simulated 30 independently generated binary datasets. For each dataset, we tested 4 models with 10 initializations each and laps = 1, 5.

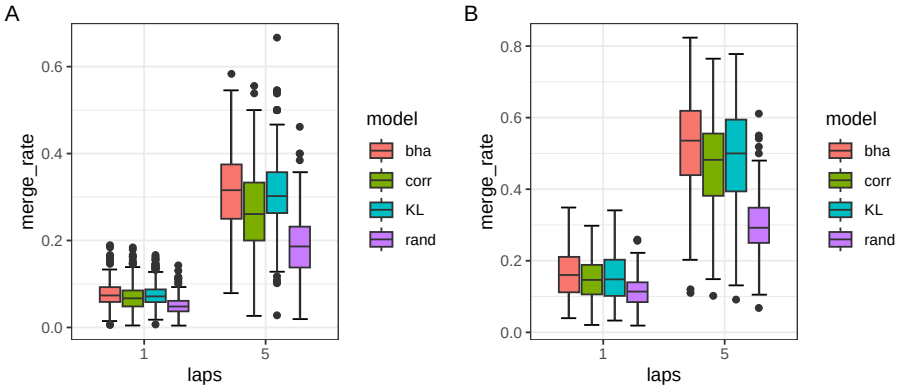
Figure 9 shows no significant differences between methods in time and accuracy, though random selection took slightly longer (especially with laps = 5 and more clusters) and achieved slightly lower median ARIs in study B. Random selection had significantly lower merge acceptance rates, while correlation had lower rates than divergence measures. However, higher computational complexity of divergences resulted in similar total algorithm times.



(a) Comparing log(total time (s))



(b) Comparing ARI



(c) Comparing 'merge rate', defined as proportions of merges accepted.

Figure 9: Boxplots comparing the distributions of ARI scores, (the logarithm of) total wall-clock time until convergence, and 'merge rates'. All initialisations for each model, simulated dataset, and 'laps' value are included, where the boxplots illustrate the median, quantiles, and range across all clustering solutions.

H.5. Categorical Data Simulations

H.5.1. Frequency of Merge/Delete Moves

Details of the scenarios for this simulation study with categorical data (analogous to Section G.3) are given in a table below. Both simulations were run without variable selection. In both scenarios, we generated 20 independent datasets and compared 10 different initialisations. Table 12 shows results; the model with no merge/delete moves (laps = 10000) performed markedly worse in ARI.

Table 11: Parameters for data generation for frequency of merge/delete moves with categorical data.

ID	N	P	K_{init}	K_{true}	N per Cluster	Categories
cat.1	2000	100	20	8	50-800	4
cat.2	4000	100	25	10	200-800	3

Table 12: Comparisons of mean wall-clock time, mean final log-ELBO, mean ARI and mean number of clusters of final labelling structure across all runs in Simulations cat.1, cat.2 with confidence intervals for time and ARI (+- 1.96 x standard deviation).

Simulation	Laps	Time (s)	log-ELBO	ARI	Clusters
cat.1	0	**14.4 [14.2, 14.5]**	-238524	0.961 [0.951, 0.971]	7.72
	1	27.6 [25.6, 29.5]	-238378	0.994 [0.993, 0.995]	7.48
	2	24.5 [23.1, 25.8]	-238356	**0.995 [0.994, 0.996]**	7.56
	5	29.2 [28.5, 29.8]	**-238352**	**0.995 [0.994, 0.996]**	7.71
	10	38.7 [37.8, 39.5]	-238513	0.986 [0.984, 0.988]	8.37
	10000	79.2 [74.6, 83.8]	-241501	0.831 [0.827, 0.835]	20
cat.2	0	**34.8 [34.1, 35.4]**	-360586	0.991 [0.988, 0.995]	10.1
	1	47.0 [46.0, 48.1]	**-360544**	**0.999 [0.999, 1.00]**	10.0
	2	49.8 [48.8, 50.7]	**-360544**	**0.999 [0.999, 1.00]**	10.0
	5	66.1 [65.1, 67.0]	-360568	0.998 [0.998, 0.999]	10.2
	10	91.1 [89.5, 92.8]	-360795	0.991 [0.990, 0.992]	12.1
	10000	172 [162, 181]	-312628	0.939 [0.937, 0.941]	25

H.5.2. Global Merge Simulations

We employed random search for global merging using KL divergence to assess cluster similarities. Three scenarios used simulated data with N = 20,000, 50,000, 100,000 observations, K = 20, 10, 12 true clusters, and 4, 5, 5 equally-sized batches respectively. All scenarios had $P = 100$ covariates with 3 categories each. With 5 datasets and 10 runs per model, results showed near-perfect clustering. FedMerDel and parallelised MerDel improved speed, with FedMerDel showing increasing efficiency gains as N increased.

Table 13: 'Global Merge Simulations' with categorical data results. We report the median and lower/upper quantiles across all 10 'shuffles'/initialisations and all 5 synthetic datasets. Note the higher merge times and total times with $N = 20,000$ is likely due to more initialised clusters (30 vs. 20 in the larger N settings).

N	Model	ARI	Clusters	Time (s)	Global Merge Time (s)
20000	full	0.999 [0.999, 0.999]	20 [20, 20]	387 [373, 418]	
	par	0.999 [0.999, 0.999]	20 [20, 21]	146 [128, 157]	
	FedMerDel	0.998 [0.99, 0.998]	20 [20, 20]	124 [118, 131]	3.91 [3.82, 4.14]
50000	full	0.999 [0.999, 1.00]	10 [10, 10]	374 [373, 375]	
	par	0.999 [0.999, 1.00]	10 [10, 10]	127 [125, 128]	
	FedMerDel	0.999 [0.999, 1.00]	10 [10, 10]	83.5 [80.7, 84.1]	1.29 [1.28, 1.34]
100000	full	1.00 [0.999, 1.00]	12 [12, 12]	946 [857, 1050]	
	par	1.00 [0.999, 1.00]	12 [12, 12]	293 [276, 339]	
	FedMerDel	0.999 [0.990, 1.00]	12 [12, 13]	243 [219, 280]	2.57 [2.49, 2.64]

Appendix I. MNIST Study

I.1. Simulation Set-Up

We applied FedMerDel to the MNIST collection of $N = 60000$ images of handwritten digits 0-9 (LeCun et al., 2011). As in Mansinghka et al. (2016), we downsampled each image to 16×16 pixels using bilinear interpolation via TensorFlow, and represented this as a 256-dimensional binary vector. Furthermore, to reduce computation on irrelevant pixels (e.g. those in corners of the image) we removed all pixels with fewer than 100 non-zero values across the whole dataset, leaving 176 covariates . We split this into 6 batches of $N = 10000$ and set $K = 20$ maximum clusters per batch. True labels for the MNIST digits were discarded.

I.2. Results

We ran FedMerDel on 5 different 'shuffles' of the data into batches with $K = 20$ maximum clusters in each batch, and our resulting models after a global merge consisted of 27-33 clusters. The observed variance in clustering outcomes reflects the genuine difficulty of this unsupervised task - handwritten digits exhibit substantial within-class variation when represented as binary vectors, and this variance is comparable to other unsupervised clustering methods on MNIST. It is unrealistic to expect to find 10 clusters exactly due to the variation of handwritten digits within numbers. The increased number of clusters was not an artefact of the global merging; a run of MerDel on the MNIST test set of $N = 10,000$ digits with 35 maximum clusters found between 33-35 clusters where most merges/deletes were rejected. Other unsupervised clustering methods also found a high number of clusters for MNIST (Hughes and Sudderth, 2013; Ni et al., 2020).

Figure 12 depicts the distribution of true number labels within each cluster, and demonstrates subclustering within digits in the 'best' model out of our 5 clustering structures. This is chosen as the model with the highest ELBO, an approach used in other variational models (Ueda and Ghahramani, 2002; Hughes and Sudderth, 2013). FedMerDel achieved more consistent performance on certain well-separated digits: for example, 95.95% of 1's and 92.46% of 6's were classified into clusters predominantly made up of that digit, while digits with similar shapes (4, 7, 9) were more challenging to distinguish. These numbers have quite similar shapes and have a variety of shapes (Figure 11). We expected imperfect performance due to the reduction of the image to a 1D vector; every pixel is treated as independent of one another, and we lose all information from the 2D images pertaining to correlation between nearby pixels. Further classification rates can be found in Appendix I.3, as well as a comparison to other scalable clustering methods.

This analysis serves as a supplementary benchmark on real-world data with ground truth labels, while our primary application domain remains EHR data where our simulated experiments demonstrate much lower sensitivity to batch partitioning.

We can use the posterior distribution of our model to probabilistically produce new digits from each cluster, showing the generative potential in variational Bayes; see Figure 13.

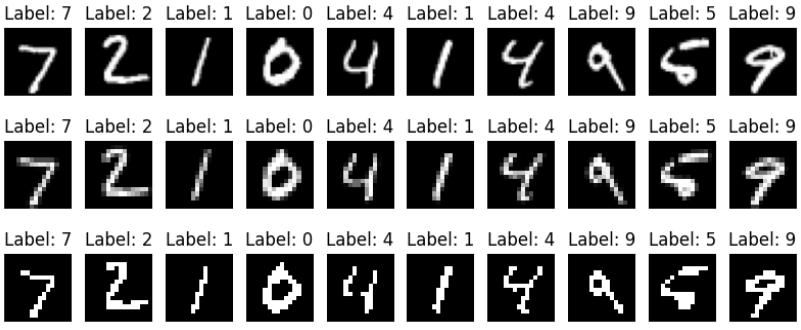


Figure 10: Example of the first 10 MNIST test-set digits visualised in their original image format (28x28), the same digits with the dimensionality reduction applied (16x16), and the same dimension reduced digits converted to a binary format.

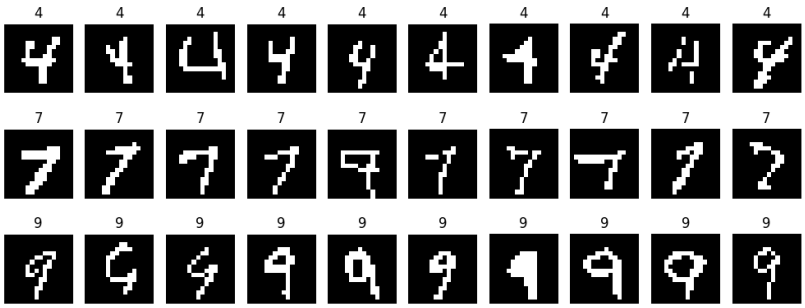


Figure 11: 10 randomly selected digits of 4, 7 and 9 from the MNIST dataset, showing similarities in their shapes.

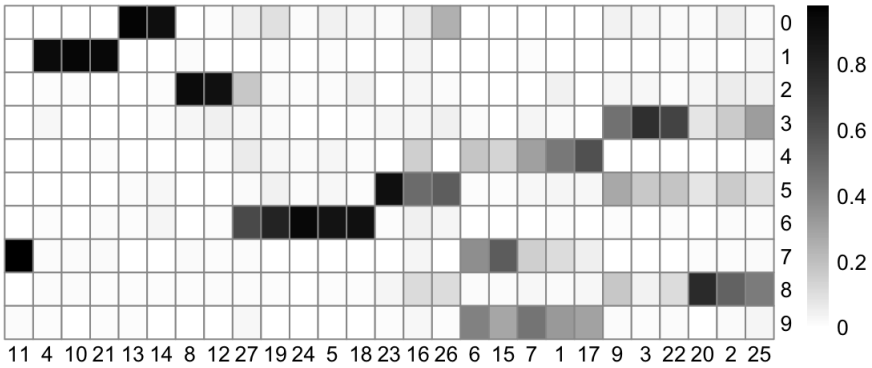


Figure 12: A heatmap showing the correspondence between clusters and true numbers in the clustering model with the best ELBO for the MNIST data. A darker cell colour in entry (i, j) indicates a higher percentage of samples from number i are in the given cluster j. The exact number corresponds to the percentage of cluster j made up of the given number.

I.3. Supplementary Results - Classification Rates

In Table 14, the models included are:

- **FedMerDel_rand:** FedMerDel, with 6 equally sized batches of data with 20 maximum clusters in each batch. A random search was used for the global merge. All models took between 77-126 mins.
- **FedMerDel_greedy:** FedMerDel, but with a greedy search used for the global merge. The same batches were used as with FedMerDel_rand above. All models took between 77-127 minutes.
- **FedMerDel_30r**: FedMerDel with 6 equally sized batches of data, but now with 30 maximum clusters in each batch. These models came with high computational time with 60 extra clusters being processed, and reduced scope for interpretation of clusters. Resulting models had between 52-70 clusters. This model used a random search.
- **FedMerDel_30g**: Same model and batches as above, but with a greedy search. These models had between 52-61 clusters.
- **kmeans_xx_yy**: Data was projected to a lower dimension with Multiple Correspondence Analysis (MCA). We then implemented the k-means algorithm on this lower-dimensional data. xx represents the number of MCA dimensions kept in the model: 10 (explaining 30% of variance) or 17 (explaining 40% of variance). yy represents the number for k, the number of clusters, where we test 20 and 35. All cases use 20 maximum iterations for k-means, and 20 different initialisations, and present the clustering with the best total within-cluster sum of squares. This method was quick (taking no longer than a minute) and generally performed well, but did not separate numbers such as 2, 3 and 6 as well as FedMerDel.

- **EM_yy**: We applied an EM algorithm for frequentist model estimation to fit a finite mixture model to the MNIST binary data via the R package *flexmix* (Leisch, 2004) and used the BIC (Bayesian Information Criterion) for model selection across 10 initialisations. yy represents the number of clusters in the finite mixture model, and we test 20 (10.2 mins) and 35 (55.4 mins).

Generally, FedMerDel performed similarly well to comparator methods.

Table 14: Table comparing classification rates (%) for each digit across different clustering models. The model analysed further was the 4th FedMerDel_rand run as the model with the highest ELBO of all the FedMerDel_rand and FedMerDel_greedy runs. Classification rate is defined as the % of digits classified into a cluster primarily consisting of that digit.

MODEL	**RUN**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
FedMerDel_rand	1	83.4	95.5	83.7	75.5	49.6	37.8	90.4	62.2	65.6	**55.3**
FedMerDel_rand	2	84.1	94.8	**86.3**	65.2	54.8	38.6	90.1	56.0	67.3	47.8
FedMerDel_rand	3	**86.4**	95.4	85.8	73.9	47.6	34.6	**92.7**	63.0	64.4	49.4
FedMerDel_rand	4	81.4	**96.0**	83.8	71.7	51.7	**47.2**	92.5	67.7	**70.2**	44.8
FedMerDel_rand	5	82.7	94.6	83.4	**83.4**	**57.7**	34.5	92.0	**74.5**	52.4	34.7
FedMerDel_greedy	1	83.4	95.5	83.7	75.5	53.7	37.8	90.4	62.2	65.6	**51.4**
FedMerDel_greedy	2	84.1	94.8	**86.3**	70.2	54.8	33.6	90.1	56.0	64.5	47.8
FedMerDel_greedy	3	**86.4**	95.4	85.8	73.9	47.6	34.6	**92.7**	63.0	64.4	49.4
FedMerDel_greedy	4	81.4	**96.0**	83.8	71.7	51.7	**47.2**	92.5	67.7	**70.2**	44.8
FedMerDel_greedy	5	82.7	94.6	83.4	**83.4**	**57.7**	34.5	92.0	**74.5**	52.4	34.6
FedMerDel_30r	1	**91.8**	94.5	85.7	80.5	54.5	51.7	**94.3**	71.0	68.8	50.3
FedMerDel_30r	2	88.6	95.3	**85.8**	76.3	56.3	46.2	93.2	70.1	**69.0**	**59.7**
FedMerDel_30r	3	91.3	**95.4**	85.2	**83.0**	**59.8**	**53.8**	93.1	**78.5**	65.2	34.5
FedMerDel_30g	1	**91.8**	94.5	85.7	80.5	54.5	51.7	**94.3**	69.0	68.8	52.9
FedMerDel_30g	2	88.6	95.3	**85.8**	74.9	56.3	44.8	93.2	70.1	**71.6**	**59.7**
FedMerDel_30g	3	91.3	**95.4**	85.2	**84.2**	**62.4**	**53.8**	93.1	**76.9**	60.7	34.5
k-means_10_20		81.4	97.8	65.9	74.6	54.3	31.1	83.2	70.8	40.2	39.9
k-means_10_35		86.5	95.8	**73.5**	72.9	**77.6**	**43.4**	**86.7**	**81.2**	49.5	27.7
k-means_17_20		83.1	**98.6**	63.7	60.9	51.9	33.4	85.0	79.7	46.3	32.6
k-means_17_35		**91.5**	97.8	68.3	**77.9**	62.1	23.5	83.6	75.7	**57.0**	**50.0**
EM_20		82.1	95.6	83.7	84.9	**62.2**	29.7	91.8	72.4	57.4	38.0
EM_35		**87.6**	**96.1**	**90.2**	**85.8**	51.8	**51.9**	**94.8**	**81.8**	**72.3**	**62.6**

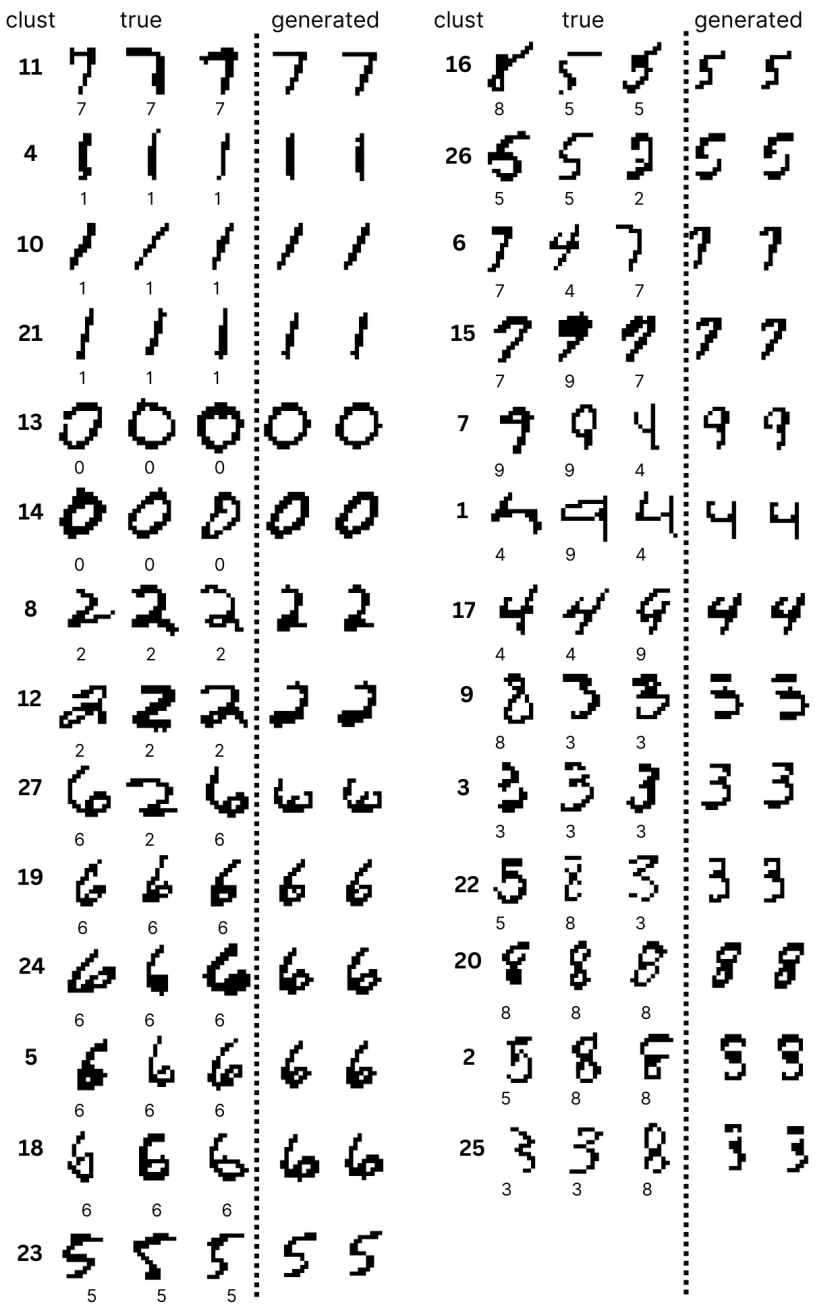


Figure 13: 3 randomly selected observations and 2 generated digits (from the variational distribution) from each cluster in the clustering model with the best ELBO for the MNIST data.

Appendix J. EHR Data - Additional Information

EHR data were obtained from The Health Improvement Network database. The start date for our study was 01 January 2010 and end date was 01 January 2017. The dataset comprised primary care health data for 289,821 individuals over 80 years old who had at least two of the long term health conditions considered in our analysis. The full list of health conditions, and their coding, is provided in Table 17. Prevalence of the conditions in Figure 2 is in Table 16. 64.8% of individuals contributing to the analysis dataset were female, 35.2% were male.

J.1. Characterisation of Random Search Clusters

Note the mean mortality rate across the full dataset was 0.48, and the mean number of conditions per person was 6.19. Mortality rate is defined as the number of people with a registered death date on or before 31/12/2020.

Table 15: Characterising the clusters seen in Figure 2 by the mean number of conditions and mortality rate.

Cluster	Mean No. of Conditions	Mortality
(i) cancer	6.90	0.49
(ii) af & arrhythmia	8.48	0.57
(iii) pvd & aortic aneurysm	8.77	0.61
(iv) stroke	8.46	0.54
(v) af & arrhythmia & stroke	11.3	0.61
(vi) stroke & blindness	12.6	0.67
(vii) blindness	8.69	0.53
(viii) dementia	6.14	0.61
(ix) asthma & copd	6.30	0.54
(x) 'generally multimorbid'	4.57	0.39
(xi) 'generally multimorbid'	9.40	0.41
(xii) 'generally multimorbid'	7.85	0.55

Table 16: Table providing prevalence as a percentage for the conditions in Figure 2. The most prevalent condition not included in this table is gout, at a prevalence of 7.63%, which is a condition often found as a comorbidity, but it is not expected that this should form a separate cluster. Other conditions are less common.

Coding	Prevalence	Coding	Prevalence	Coding	Prevalence
cancerall	13.8	tia_stroke	18.5	anxiety	9.38
ca_skin	12.6	eczema	16.0	dementia	13.1
arrythmia	18.3	deaf	27.0	oa	39.3
af	16.3	blind	5.94	osteoporosis	14.4
hypertension	63.2	cataract	34.9	ckd	33.2
hf	10.1	glaucoma	9.77	asthma	12.2
ihd	26.8	amd	8.47	copd	8.20
pvd	7.64	retinopathy	8.08	hypothyroid	13.1
aortic_aneurysm	1.84	diverticuli	14.6	t2dm	14.7
tia	9.55	depression	13.6	bph	9.43
nos_stroke	10.4				

J.2. Heterogeneous EHR Simulations

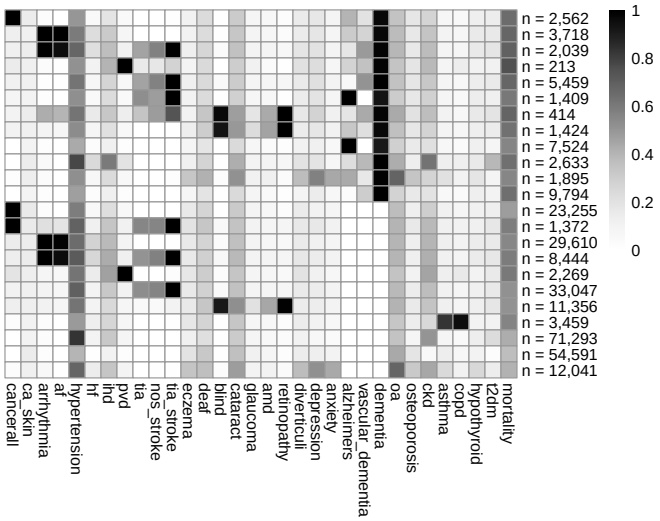


Figure 14: Results of global EHR clustering where individuals with and without dementia were separated into different clusters. 15 total similarly sized batches were randomly generated after the data was divided. Random merge was used. Similar clusters are seen, but split by those with and without dementia as expected. Notably, an asthma/COPD cluster emerges in patients without dementia, but this is not seen for patients with dementia. Similar mortality/mean number of conditions as in Appendix J.1 could be found to infer clinical risk based on having dementia as an additional comorbidity or not. To improve visualisation, only the most prevalent health conditions are shown.

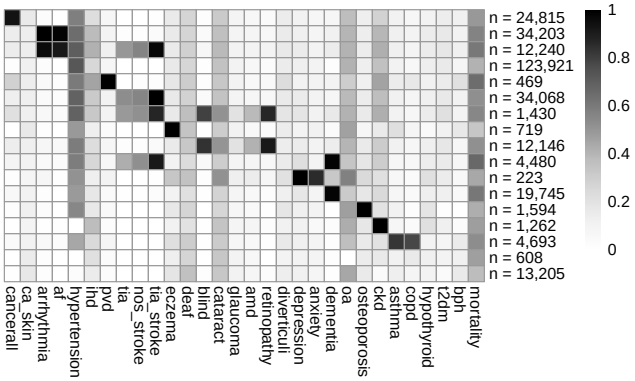


Figure 15: Results of global EHR clustering where a symmetric 15-dimensional Dirichlet(1) distribution was used to simulate heterogeneity. This was done by taking the 12 clusters from the analysis in the main manuscript, and assigning observations from each cluster into 15 new batches according to the Dirichlet distribution. Random merge was used. Core clusters are still clearly retained (eg. cancer, respiratory disorders, stroke only, mixed stroke + AF/arrhythmia) while smaller clusters emerge, likely driven by shard specific overrepresentation. For example, a small multimorbid cluster emerges with no osteoarthritis. To improve visualisation, only the most prevalent health conditions are shown.

J.3. Greedy Search Global Merge

In Figure 3 of the main manuscript, we illustrated the global clusters identified using the "random search" approach to perform the global merge. In Figure 16 below, we provide the corresponding results obtained using the "greedy search" approach.

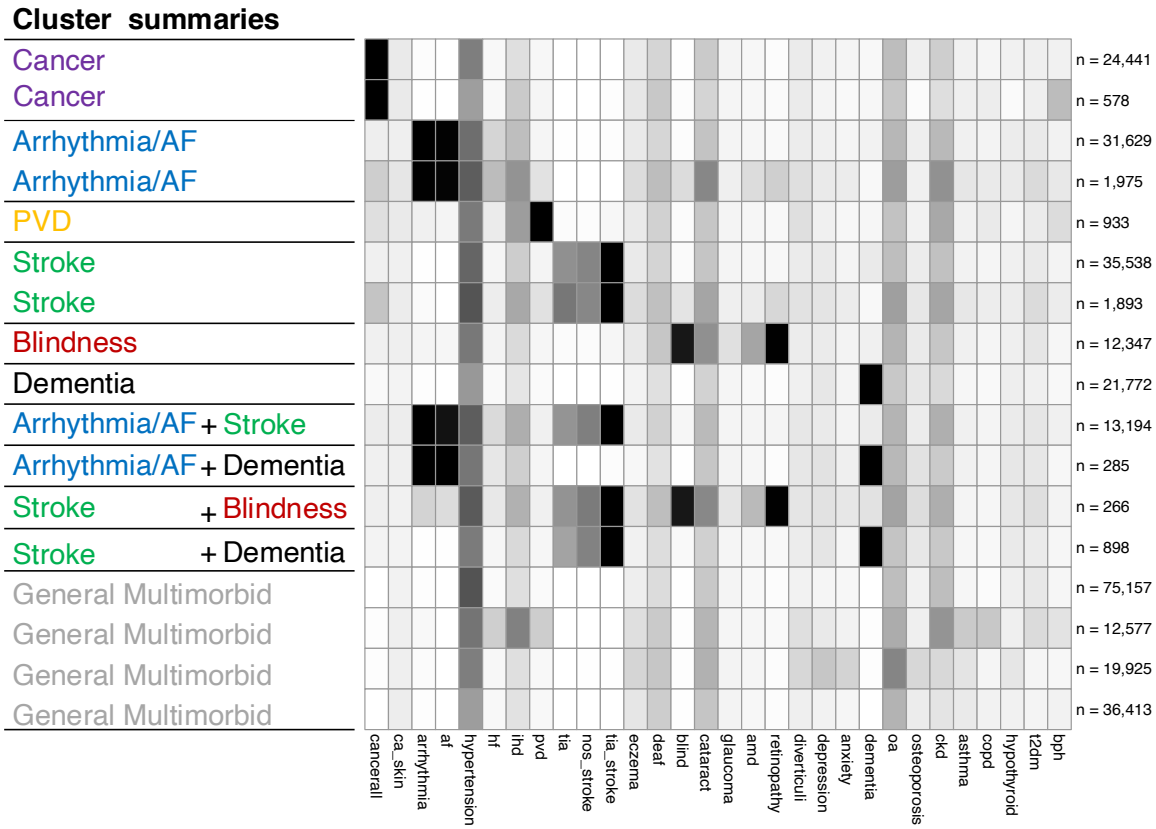


Figure 16: Results of global EHR clustering using the greedy search approach to perform the global merge. Columns are health conditions, and rows correspond to clusters. Row labels (shown on the right) indicate the number of individuals in each cluster. Summary names for each of the clusters are provided on the left of the plot. To improve visualisation, only the most prevalent health conditions are shown.

J.4. Full List of Health Conditions

Table 17: Table providing the full list of health conditions, and their coding used in the analysis and figures.

Coding	Condition	Coding	Condition
cancerall	Any cancer diagnosis	endometriosis	Endometriosis
ca_lung	Lung cancer	pcos	Polycystic ovary syndrome
ca_breast	Breast cancer	pernicious_anaemia	Pernicious anaemia
ca_bowel	Bowel cancer	vte	Venous thromboembolism
ca_prostate	Prostate cancer	pulmonary_embolism	Pulmonary embolism
lymphoma	Lymphoma	coagulopathy	Coagulopathy
leukaemia	Leukaemia	depression	Depression
ca_skin	Skin cancer	anxiety	Anxiety
melanoma	Melanoma	smi	Serious mental illness
ca_metastatic	Metastatic cancer	substance_misuse	Substance misuse
arrhythmia	Arrhythmia	alcohol_problem	Alcohol problem
af	Atrial fibrillation	adhd	Attention deficit hyperactivity disorder
hypertension	Hypertension	eating_disorder	Eating disorder
hf	Hear failure	learning_disability	Learning disability
ihd	Ischemic heart disease	alzheimers	Alzheimer's disases
valve_disease	Heart valve disease	vascular_dementia	Vascular dementia
cardiomyopathy	Cardiomyopathy	dementia	Dementia
con_heart_disease	Congenital heart disease	parkinsons	Pakinson's disease
pvd	Peripheral vascular disease	migraine	Migraine
aortic_aneurysm	Aortic aneurysm	ms	Multiple sclerosis
tia	Transient ischaemic attack	epilepsy	Epilepsy
isch_stroke	Ischaemic stroke	hemiplegia	Hemiplegia
haem_stroke	Haemorrhagic stroke	cfs	Chronic fatigue syndrome
nos_stroke	Stroke (not otherwise specified)	fibromyalgia	Fibromyalgia
tia_stroke	TIA stroke	oa	Osteoarthritis
eczema	Eczema	osteoporosis	Osteoporosis
psoriasis	Psoriasis	polymyalgia	Polymyalgia
vitiligo	Vitiligo	ra	Rheumatoid arthritis
alopecia	Alopecia	sjogren	Sjögren's syndrome
rhin_conjunc	Rhinoconjunctivitis	sle	Systemic lupus erythematosus
sinusitis	Sinusitis	systematic_sclerosis	Systematic sclerosis
deaf	Deafness	psoriatic_arthritis	Psoriatic arthritis
blind	Blindness	ank_spond	Ankylosing spondylitis
cataract	Cataract	gout	Gout
glaucoma	Glaucoma	ckd	Chronic kidney disease
amd	Age-related macular degeneration	copd	Chronic obstructive pulmonary disease
retinopathy	Retinopathy	asthma	Asthma
uveitis	Uveitis	osa	Obstructive sleep apnea
scleritis	Scleritis	bronchiectasis	Bronchiectasis
peptic_ulcer	Peptic ulcer	pulmonary_fibrosis	Pulmonary fibrosis
ibd	Inflammatory bowel disease	hyperthyroidism	Hyperthyroidism
ibs	Irritable bowel syndrme	hypothyroid	Hypothyroidism
liver_disease	Liver disease	t1dm	Type 1 diabetes mellitus
nash_nafl	Nonalcoholic steatohepatitis/ Nonalcoholic fatty liver	t2dm	Type 2 diabetes mellitus
diverticuli	Diverticulitis	hiv	Human immunodeficiency virus
coeliac	Coeliac disease	bph	Benign prostatic hyperplasia
pancreatitis	Pancreatitis	erectile_dysfunction	Erectile dysfunction