Particle Monte Carlo methods for structured mixture models for multiple datasets

Nathan Cunningham (University of Warwick)

under the supervision of
Prof. David Wild (University of Warwick)
Prof. Jim Griffin (University of Kent)

September 14, 2016

Abstract

Cluster analysis is a popular technique for discovering group structure in a dataset. In many real-world cases, the analysis may be helped by incorporating information from a number of different data sources and incorporating data of different types. Classic clustering algorithms, such as k-means and hierarchical clustering, are geared towards continuous data and are not suitable for mixed data types. Previous work on clustering multiple datasets using a one-at-a-time approach has been found to potentially show slow-mixing properties. Sequential Monte Carlo (SMC) methods have been proposed as a means of overcoming this. In this report, we consider the case of multiple dataset integration using SMC methods to update cluster allocations. We compare the results to those obtained from previous algorithms and find our algorithm to perform comparably in terms of cluster allocation, but less favourably in terms of Markov chain diagnostics and computational time.

1 Introduction

Cluster analysis is a commonly used statistical technique for the uncovering of structure within data. The goal is to partition the data into a collection of meaningful groups such that observations within groups are similar to each other while being dissimilar to observations in other groups.

Although it is used in a variety of fields, cluster analysis has proved powerful in the context of genomic data, where it can be used to discover groups of genes which tend to co-express under similar conditions, or groups of patients who share common molecular phenotypes. Recent years have seen great increases in the quantity and variety of genomics data being collected. While cluster analaysis can be applied to these data in order to uncover underlying group structure, independent clustering of separate datasets overlooks the shared information between datasets. Previous work by Kirk et al. [9] proposed a method of multiple dataset integration (MDI) in which information could be shared across a number of datasets of different types on the degree of agreement in group structure between them. They find that sharing clustering information across datasets allowed for the uncovering of more detailed group structures. However, the type of one-at-a-time cluster update as proposed in the MDI algorithm may demonstrate slow-mixing properties. Posterior sampling from mixture models has proved challenging with Markov chain Monte Carlo (MCMC) methods, largely caused by the difficulty of exploring the posterior distribution of cluster membership labels, and Sequential Monte Carlo (SMC) methods [2] have been proposed to overcome this problem and Griffin[7] demonstrates that these methods can lead to good mixing properties with a small number of particles.

In this report we consider the application of particle filter methods for the joint clustering of all cluster membership labels in the class of MDI models. Primarily, we build on the work by Kirk et al. [9] which focused on the problem of multiple dataset integration, and the work by Griffin[7] which focused on the application of SMC methods to clustering.

Section 2 discusses previous work in cluster analysis and multiple dataset integration, section 3 proposes a method of conducting cluster analysis using particle Monte Carlo methods in the context of multiple datasets, section 4 presents the results of applying the algorithm to some example data and compares these results with those of the original MDI algorithm, section 5 concludes and considers the direction of future work.

2 Background

In this section, I will review previous work done in cluster analysis, for single and multiple datasets, and the use of sequential Monte Carlo methods for clustering.

Cluster analysis can be broadly described as the analysis of datasets to uncover a hidden underlying structure. Typically these clusters will be found to minimise some measure of within-cluster distance, such that observations within clusters are more alike, in some sense, with each other than than they are with observations in other clusters. The problem of cluster analysis is different from the problems of classification and discriminant analysis as it is typically carried out in an unsupervised manner. That is, in classification there typically is a set of true labels and, given the data we wish to infer some rules defining membership to these labels. In cluster analysis no such ground truth is typically available and we wish to infer some structure amongst the observations. Clustering algorithms are largely divided into hierarchical, which aims to impose a nested structure on the data, and partitional algorithms, which aim to divide the data into meaningful groups[8].

2.1 Hierarchical cluster analysis

Hierarchical clustering algorithms aim to impose a hierarchical structure on the data. In agglomerative hierarchical clustering, beginning with all observations in individual clusters the algorithm iterates through each of the data points, merging the most similar observations into a shared cluster until, at the end, all observations belong to an individual cluster. This is usually done through some form of distance metric, e.g. standard Euclidean distance. The output of a hierarchical clustering algorithm can be examined in a dendrogram and a final clustering allocation can be inferred heuristically. Divisive hierarchical clustering performs the same operations, but beginning with all observations in a single cluster and iteratively dividing until all observations exist in individual clusters.

2.2 K-means

A classic partitional clustering algorithm is the k-means algorithm, which aims to partition the data into k clusters. The optimal allocation minimises the within-cluster sum of squares, which is the sum of the squared distances from each observation within a cluster to the centre of the respective cluster. This specification of the loss function as the sum of squares applies well to numeric data but does not generalise to other types of data as, for example, a sum of squared distances is not meaningful for ordinal data. The classic k-means algorithm begins with a random initialisation of observations to clusters and the algorithm iterates through: calculating cluster centres for each of the k clusters; and moving observations to the cluster which minimises the squared distance between the cluster center and the observation, until convergence. Pena et al.[14] demonstrate the sensitivity of the output of a k-means algorithm to this random initialisation.

2.3 Finite Mixture Model

Finite mixture models model the data as arising from a mixture of N components, with the data underlying each component arising from the same parametric distribution, e.g. the Gaussian distribution. They have the following form:

$$p(x_i) = \sum_{c=1}^{N} \pi_c f(x_i|\theta_c) \tag{1}$$

where the π_c values are the mixing proportions, f is a chosen parametric distribution with parameter θ_c. The parameters of the model, π and θ can be updated using the expectation maximisation (EM)

algorithm. Typically the choice of distribution to use and the number of components to model are based on heuristic approaches, however Fraley and Raftery[5] propose a more systematic approach.

2.4 Nonparametric Mixture Model

Allowing the number of components in a finite mixture model, N, to tend to infinity realises the specification of the nonparametric mixture model where

$$\begin{aligned} x_i|c_i &\sim f(x_i|c_i, \theta_i) \\ c_i &\overset{i.i.d}{\sim} \pi \end{aligned} \tag{2}$$

Where c_i identifies a component which is responsible for the observation x_i. Not all components need have observations attached to them, and, as such they may specify potentially infinite numbers of components. Of these the most popular is the Dirichlet process mixture model [4] where π is given a Dirichlet process prior. These nonparametric mixture models provide greater flexibility as the number of clusters can be inferred and need not be specified a priori.

2.5 Clustering with multiple data sources

One of the problems facing many of these clustering algorithms is the issue of incorporating information from different sources, and potentially data of different types. In the classic approaches, hierarchical and k-means clustering, the individual datasets may be clustered independently or concatenated and treated as a single dataset. These approaches may provide spurious results in the case of mixed data types. Ahmad and Dey[1] proposed a variation of the k-means algorithm for clustering a mixture of numerical and categorical data. Where previous work had proposed treating each of the data types as equivalent, they propose a modification of the loss function that incorporates costs for each of numeric and categorical data.

McParland et al.[12] propose a mixture of factor analysers model for mixed data (MFA-MD). They consider the clustering of a mixture of binary, ordinal, and nominal items from survey data. Further work[13] added the ability to cluster continuous data. They proposed a finite mixture model to perform cluster analysis at the level of latent variables constructed from a factor analysis of the data.

In their paper, Kirk et al.[9] propose an unsupervised method for the integration of multiple datasets of different types (MDI). They model each dataset as arising from a Dirichlet-multinomial allocation mixture model which is a finite approximation to the Dirichlet process mixture model. They aim to infer a shared cluster structure across a number of different datasets and data types. While it requires the specification of a maximum number of components, N it does not require an explicit specification of a fixed number of clusters to be fit to the data, instead this is inferred by the algorithm. This gives the model greater flexibility than other models, such as k-means where specification of a particular k is necessary. In the original paper, this limit is suggested as one-half of the number of observations, a midpoint between the minimum number of clusters (1), and the maximum number of clusters (the number of observations). This is a trade-off between computational cost and model flexibility. They introduce a latent component allocation variable, c, where $c_i, k \in [1, \ldots, N]$ is the index of the component responsible for observation $x_{i,k}$. Their model can be viewed as a correlated clustering model, wherein the model aims to infer the degree of dependence between the clustering structures of different datasets. To this end, they introduce a parameter Φ which learns the degree of similarity between the clustering structures in different datasets. These values are then used to upweight the probabilities of observations belonging to specific clusters based on their concordance across datasets. The model implicitly matches up component labels, so there is correspondence in labels across datasets. Their algorithm uses a Gibbs sampling approach, alternating between updating the cluster allocations and updating the hyperparameters (e.g. the Φ values); their approach is similar to that presented in Algorithm 1 in Section 3. In updating the cluster allocations, however, their algorithm iterates through each data point, considers the likelihood of the data belonging to each of the clusters and moves it to that with the greatest likelihood. These types of one-at-a-time clustering algorithms may exhibit slow mixing properties.

2.6 Sequential Monte Carlo clustering methods

Sequential Monte Carlo (SMC) methods have been proposed as a solution to the slow-mixing properties of one-at-a-time clustering algorithms. SMC methods are a set of simulation methods often applied in the context of state-space models wherein they approximate the posterior at a time t using an estimate of the posterior at time $t-1$. Particle Markov chain Monte Carlo methods[2] use these SMC methods within MCMC methods for inference. Under mild assumptions they leave the target density of interest invariant and lead to convergent algorithms. Griffin[7] demonstrates the applicability of these methods to cluster analysis in the case of nonparametric mixture models with Dirichlet priors and normalised random measure with independent increments priors. Griffin finds that these methods can lead to good mixing properties with a small number of particles.

2.7 Point estimates from Bayesian clustering models

Bayesian nonparametric approaches to clustering provide a posterior over the entire space of clusterings, although typically a point estimate along with some associated measure of uncertainty are of interest. Selecting this point estimate is not a trivial exercise as there is no agreed method for selecting it. Doing the intuitive thing and assigning an observation to a cluster based on the empirical probability of doing so from an MCMC chain can lead to nonsensical results due to the problem of label-switching. Consider the trivial example where a model is fitting two clusters, labelled 1 and 2. An accurate algorithm may assign each observation to the correct cluster, but with cluster labels randomly switched between iterations. Interpreting these results as implying each data point being equally likely to be assigned to each of cluster 1 and cluster 2, would not be informative.

In identifying a final cluster allocation Kirk et al. follow the approach proposed by Fritsch et al.[6] which suggests using the cluster allocation which maximises the posterior expected adjusted Rand index (ARI). The ARI is a measure of the level of agreement between two allocations adjusted for the level of agreement expected by chance. The ARI is similar to the also commonly-used Binder's loss[3] which penalises disagreement between clusters.

Recent work by Wade and Ghahramani[16] deals with the issue of presenting some measure of confidence around a point estimate by approaching the situation from an information and decision theoretic view. They consider a loss function associated with estimating a true cluster allocation by some point estimate, suggesting the use of the variation of information (VI) as this loss function. The VI compares the information contained within two clusterings with the information shared between them. Due to the impossibility of evaluating a loss function over all possible cluster allocations they propose a greedy search algorithm which allows for the specification of a ball of size ϵ around an allocation, which can be used to identify allocations which are, in some sense, 'close' to the proposed allocation.

3 Methods

Given observed data $x_{1,k}, \ldots, x_{n,k}$ we wish to infer a common group structure across the $k = 1, \ldots, K$ datasets. As in [9] we introduce a latent component allocation variable, c, where $c_i \in [1, \ldots, N]$ represents the component from which observation i arose. For a single dataset, k, we specify the model as follows:

$$\begin{aligned} &x_{i,k}|c_{i,k}, \theta \sim F(\theta_{c_{i,k}}) \\ &c_{i,k}|\pi \sim Multinomial(\pi_{1,k}, \ldots, \pi_{N,k}) \\ &\pi_{1,k}, \ldots, \pi_{N,k} \sim Dirichlet(\alpha/N, \ldots, \alpha/N) \\ &\theta_c \sim G^0 \end{aligned} \tag{3}$$

where F is the distribution corresponding to density f, θ_c is the parameter for component c with prior $G^{(0)}$, π is a vector of mixing proportions, and N is the number of clusters to be fit to the data.

To allow the algorithm to model the dependence between cluster allocations across datasets, Kirk et al.[9] introduce a parameter, Φ, to model the dependencies at the level of the $c_{i,k}$ component allocation variables. The value of $\phi_{k,l}$ reflects the level of dependence between the structure in dataset k and dataset l. The assumed structure of the data is reflected in Figure 1. From this, it can be seen observation $x_{i,k}$ is assumed to arise from a distribution with parameter $\theta_{c,k}$, which is defined by its allocation component

$c_{i,k}$. The prior probability of an observation arising from component $c_{i,k}$ is π_k which is itself assigned a $Dirichlet(\alpha/N, \ldots, \alpha/N_k)$ prior.

As in the original MDI algorithm [9], the clustering structure is inferred via a Gibbs sampling approach as in 1. The hyperparameters, Φ and π are updated, then conditional on these values, a new clustering allocation is proposed.

The algorithm assumes the data arises as a structure as presented in Figure 1. This structure is inferred as detailed in Algorithms 1 and 2. The algorithm aims to infer both a clustering allocation for each dataset, c_k, and Φ, the measure of dependence in structure across datasets.

Unlike the original MDI algorithm, the clustering allocation is inferred via a particle filter approach as suggested by Griffin[7]. For observation $x_{i,k}$, the algorithm proposes a cluster allocation, $c_{i,k}^{(m)}$ for each of $m = 1, \ldots, M$ particles in each of $k = 1, \ldots, K$ datasets, given previous observations $x_{1:(i-1),k}$ and previous cluster allocations $c_{1:(i-1)}$:

$$q(c_{i,k}^{(m)} = a) \propto f(x_{i,k}|c_{i,j}^{(m)} = a) \times \pi_{i,k,a} \tag{4}$$

It should be noted that observations within each dataset are assigned independently. Dependence across datasets is accounted for in the calculation of the particle weights:

$$\xi^{(m)} = \xi^{(m)} \times \prod_{k=1}^{K-1} \prod_{l=k+1}^{K} (1 + \phi_{k,l} \mathbb{1}(c_{i,k} = c_{i,l})) \prod_{k=1}^{K} \sum_{a=1}^{N} \pi_{i,k,a} f(x_{i,k}|c_{i,k}^{(m)} = a) \tag{5}$$

As such, the weight attached to each particle is upweighted depending on the Φ values measure of the dependence in structure between the datasets.

At each step, the effective sample size is calculated (ESS) which can be interpreted as the number of independent samples needed to produce estimates with the same Monte Carlo error as the SMC algorithm[7]. If this drops below a certain threshold, αM, the particles are reweighted according to normalised weights and cluster allocations, $c_{1:i,k}^{m}$, are updated for all K datasets, and M particles. Full details of the algorithm are in Algorithm 2.

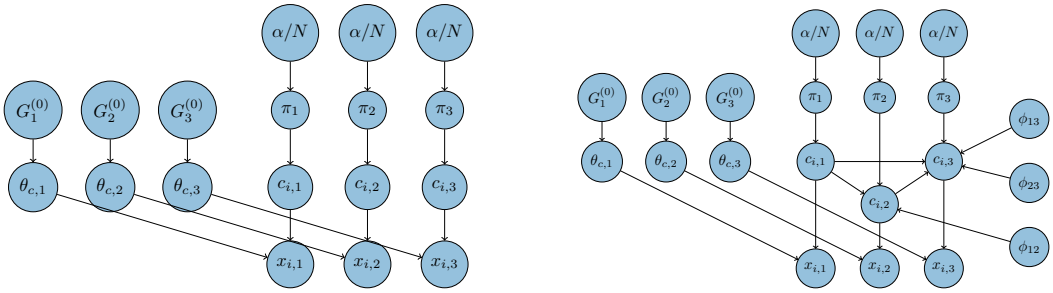

Figure 1: Graphical representation of the model in the case where $K = 3.$, as described in [9] for the independent case (left) and the MDI model (right). $x_{i,k}$ represents the i^{th} observation in dataset k, which is generated by mixture component $c_{i,k}$. The prior probabilities associated with the component allocation variables $[c_{1,k}, \ldots, c_{N,k}]$ are assigned a prior probability π_k, which is itself assigned a symmetric Dirichlet prior with parameter α/N. The parameter $\theta_{c,k}$ is assigned a G_k^0 prior. In the case of the MDI model, the $\phi_{l,k}$ parameter models the dependence between cluster allocations in dataset l and dataset k. This is in contrast to the independent case, where there is no information shared between the data sources.

Algorithm 1 Gibbs sampler

1: Initialise $\boldsymbol{\pi_0}$ matrix of prior allocation weights and $\boldsymbol{\phi_0}$ matrix of dataset concordance values
2: **for** i = 1, ..., number of iterations **do**
3: Conditional on $\boldsymbol{\pi_{i-1}}$ and $\boldsymbol{\phi_{i-1}}$ update the cluster labels, $c_{i,\cdot}$, using algorithm 2
4: Conditional on c_i update $\boldsymbol{\pi_i}$ and $\boldsymbol{\phi_i}$
5: **end for**

Algorithm 2 Particle filter to update cluster allocations

Inputs:
A matrix of prior component allocation probabilities $\boldsymbol{\pi}$, a matrix of prior dataset dependence measures $\boldsymbol{\Phi}$, and a threshold α
Initialize:
$\xi^{(1)}, \ldots, \xi^{(M)} = 1$
for i = 1, ..., n **do**
  for m = 1, ..., M **do**
    for k = 1, ..., K **do**
      Sample $c_{i,k}^{(m)}$
      $q(c_{i,k}^{(m)} = a) \propto f(x_{i,k}|c_{i,k}^{(m)} = a) \times \pi_{i,k,a}$
    end for
    $\xi^{(m)} = \xi^{(m)} \times \prod_{k=1}^{K-1} \prod_{l=k+1}^{K} (1 + \phi_{k,l} \mathbb{1}(c_{i,k} = c_{i,l})) \prod_{k=1}^{K} \sum_{a=1}^{N} \pi_{i,k,a} f(x_{i,k}|c_{i,k}^{(m)} = a)$
  end for
  Calculate $ESS = \frac{(\sum_{m=1}^{M} \xi^{(m)})^2}{\sum_{m=1}^{M} \xi^{(m)2}}$.
  If ESS $< \alpha N$ resample particles according to $\zeta^{(m)} = \frac{\xi^{(m)}}{\sum_{m=1}^{M} \xi^{(m)}}$ and update $s_{1:i}^{(m)} (m = 1, \ldots, M)$ using MCMC. Set $\xi^{(1)}, \ldots, \xi^{(M)} = 1$.
end for
Select a final cluster label allocation according to $\zeta^{(m)} = \frac{\xi^{(m)}}{\sum_{m=1}^{M} \xi^{(m)}}$

4 Example application

In this section, we present the results of applying the particle filter implementation of the MDI algorithm (pMDI) as outlined above to a number of synthetic datasets. We compare the results with those of the original MDI algorithm applied to the same data. Specifications of the running conditions for the algorithm are contained in the appendix.

4.1 Data

A number of datasets were simulated using code provided from the original MDI paper. A group structure was first simulated from a random Dirichlet process, and then four datasets, two Gaussian and two multinomial, were simulated according to this group structure. The data are visualised in Figure 3. Although these plots represent just a single feature, it is easy to realise the importance of inferring the clustering structure from more than one data source. Consider cluster 1 and cluster 2 in `Gaussian Test Data 1`; given the level of overlap in the two groups it would be difficult to distinguish between these clusters using just this dataset. However, there is a greater level of separation in the clusters across the other datasets.

4.2 Results

4.2.1 Computational cost

The pMDI algorithm was found to be considerably more computationally expensive than the original MDI. While the original MDI algorithm completes a single iteration of the Gibbs sampler outlined above in approximately 1.5 seconds, the pMDI algorithm is considerably slower even after accounting for the number of particles (m) used. With just a single particle, performance is comparable to the original implementation, however even setting m to a modest 10 particles disproportionately increases the time to complete a single iteration of the Gibbs sampler to approximately 110 seconds. This means that the pMDI algorithm will be much slower to learn the level of dependence in the grouping structure across data sets.

4.2.2 Allocation accuracy

The final clustering allocation is summarised in Figure 2 for the pMDI algorithm (top) and the original MDI algorithm (bottom) . The 4 panels along the right-hand side show the posterior similarity matrix given by the clustering allocation for each of the datasets, while the main panel reflects the consensus across datasets. On all panels the x and y axes each represent the observations in the dataset, ordered identically across each axis. The darkness of a point (i, j) reflects how frequently observation i is allocated to the same cluster as observation j. It can be seen that each of the datasets is found to have the same structure, as each of the panels show this, and there is great agreement across the iterations of the MCMC chain, evidenced by the distinct squares in the output. The inferred cluster structure is identical across the two algorithms, although the observations are not ordered identically in the two datasets. Given the data are generated from a known group structure the inferred allocation was compared to the true structure and found to match exactly the true underlying structure.

4.2.3 Markov chain diagnostics

Using tools for the analysis of MDI output by Mason et al.[11][10] we examined the convergence of the allocation output. Figure 4 compares the convergence of the Markov chain resulting from pMDI and the original MDI algorithm. Each panel represents the pairwise allocation agreement across datasets, with the MCMC iteration number on the x-axis and the cumulative proportion of agreements on the y-axis. Using terminology from Savage et al.[15] we say a gene is "fused" across datasets $k = 1, \ldots, K$ if $c_{i,1} = c_{i,2} = \cdots = c_{i,K}$. Therefore, each line represents the cumulative proportion of MCMC steps at which a particular observation in the data (a gene in the context of genomics data) was fused across the relevant datasets. It can be seen that the output from the original MDI algorithm has complete fusion across runs of the algorithm. However, this is not the case for the pMDI algorithm, where fusion is not observed for all observations. This is particularly prevalent for the `Gaussian 1` data set. It appears that the algorithm has not fully converged, and it would appear to be heading towards less than complete fusion.

The posterior mass associated with the number of fused genes across datasets is presented in Figure 5. The x-axis represents the number of genes fused, while the y-axis indicates the relative frequency. The dashed line represents the mean number of fused genes. Again, the output of the original MDI appears more promising, with all observations fused across all MCMC iterations. For the pMDI algorithm, particularly in the case of `Gaussian 1` there are many cases in which only a small number of observations fused across datasets.

While convergence of the MCMC chain is an important diagnostic tool for any MCMC algorithm, it is not necessarily reflective of ideal performance. While MDI shows complete agreement in allocations across all runs of the MDI algorithm, it could always be the case that algorithm has simply converged to an incorrect allocation.

5 Conclusions and future work

In this report we have presented pMDI, an extension of previous work on multiple data integration by Kirk et al.[9] which incorporates the work of Griffin[7] in using a particle filter to update the cluster allocations. On example data the algorithm is found to perform comparably to the original MDI work in uncovering the underlying structure in multiple datasets of different types. There is, however, some evidence that pMDI does not converge to a stable allocation as swiftly as the original MDI algorithm.

Furthermore, the use of the particle filter for updating cluster allocations is considerably more computationally expensive than the one-at-a-time update performed in the original MDI algorithm. Due to time constraints, the examples in this report were carried out using just 10 particles, but running the algorithms on more complex data using a more meaningful number of particles may prove to be prohibitively costly in terms of computational time. As the measure of dependence in the allocations across datasets, the Φ values, are only updated once per iteration of the Gibbs sampler, the algorithm, as is, will be slow to learn the level of structural dependence across datasets. An important next step in the development of this algorithm is to improve the computational speed through the implementation of parallel computation methods.

As a starting point, the pMDI algorithm has been set up to handle Gaussian data and multinomial data. Given a particular strength of the MDI algorithm is the ability to cluster a variety of different data types, future work will involve adapting the algorithm to cluster further data types.

A further improvements to the algorithm may include the updating of the cluster allocations via a particle Gibbs approach. Griffin[7] found that doing so with adaptive resampling produced relatively uncorrelated samples with a small number of particles.

While the performance of the algorithm in uncovering the true group structure of the data is promising, it should not be forgotten that this was on simulated data. A true test of the efficacy of the model will be in its application to real genomic datasets.

References

[1] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] David A Binder. Bayesian cluster analysis. *Biometrika*, 65(1):31–38, 1978.

[4] Michael D Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical association*, 90(430):577–588, 1995.

[5] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[6] Arno Fritsch, Katja Ickstadt, et al. Improved criteria for clustering based on the posterior similarity matrix. *Bayesian analysis*, 4(2):367–391, 2009.

[7] JE Griffin. Sequential Monte Carlo methods for mixtures with normalized random measures with independent increments priors. *Statistics and Computing*, pages 1–15, 2014.

[8] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[9] Paul Kirk, Jim E Griffin, Richard S Savage, Zoubin Ghahramani, and David L Wild. Bayesian correlated clustering to integrate multiple datasets. *Bioinformatics*, 28(24):3290–3297, 2012.

[10] Sam Mason. Mdi++ Bayesian data integration of biological data. `https://github.com/smason/mdipp`, 2014.

[11] Samuel A Mason, Faiz Sayyid, Paul DW Kirk, Colin Starr, and David L Wild. Mdi-gpu: accelerating integrative modelling for genomic-scale data using gp-gpu computing. *Statistical applications in genetics and molecular biology*, 15(1):83–86, 2016.

[12] Damien McParland, Isobel Claire Gormley, Tyler H McCormick, Samuel J Clark, Chodziwadziwa Whiteson Kabudula, and Mark A Collinson. Clustering South African households based on their asset status using latent variable models. *The Annals of Applied Statistics*, 8(2):747, 2014.

[13] Damien McParland, Catherine M Phillips, Lorraine Brennan, Helen M Roche, and Isobel Claire Gormley. Clustering high dimensional mixed data to uncover sub-phenotypes: joint analysis of phenotypic and genotypic data. *arXiv preprint arXiv:1606.05107*, 2016.

[14] José Manuel Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[15] Richard S Savage, Zoubin Ghahramani, Jim E Griffin, J Bernard, and David L Wild. Discovering transcriptional modules by Bayesian data integration. *Bioinformatics*, 26(12):i158–i167, 2010.

[16] Sara Wade and Zoubin Ghahramani. Bayesian cluster analysis: Point estimation and credible balls. *arXiv preprint arXiv:1505.03339*, 2015.

Appendix

Probability models for different datasets

The original MDI algorithm provided methods for clustering a number of different data types: Gaussian processes, multinomial, bag-of-words, and Gaussian data. This report, as a starting point, has focused on Gaussian data and multinomial data. The details of the probability models used is described below.

Gaussian data

The data underlying a Gaussian dataset are assumed to arise from a discrete mixture of multivariate normal distributions.

$$x_{i,k}|c_{i,k}, \mu_k, \Sigma_k \sim \mathcal{N}(\mu_{\mathbb{k}}, \Sigma_k) \tag{6}$$

We consider the log probability of a single data point arising from an individual cluster as follows:

$$p(c_i = k|x_{1:i}, c_{1:(i-1)}, \mu_k, \Sigma, a) = -\frac{1}{2}(x_i - \mu_k)^T \Sigma^{-1}(x_i - \mu_k) \tag{7}$$

Priors are placed on the parameters of the multivariate normal distribution, $\mu_k = (0, \ldots, 0)$ and $\Sigma_k = \mathbb{I}$, the identity matrix.

For an individual cluster, the mean is calculated as:

$$\mu_k = \frac{\sum_{i=1}^{n_k} x_{i,k}}{an_k + (1-a)} \tag{8}$$

Where a is a measure of central tendency and is given a beta prior. n_k is the number of observations currently occupying cluster k. As such, μ is calculated as the ratio of the mean of the observations currently assigned to the cluster and a cluster with one observation with mean zero.

The covariance matrix is updated as:

$$\Sigma_k = \frac{\mathbb{I}}{(n_k a) + (1-a)} \tag{9}$$

Features are, therefore, treated as independent with variance decreasing as cluster size increases. Representing the covariance matrix as such has shown to give reasonable results while avoiding the need for multiple costly computations of inverse covariance matrices.

Multinomial data

Categorical data can be modelled in the algorithm by supplying the argument 'Multinomial' as data type. For each gene, we observe Q features, each of which take a value r from the set $\{1, \ldots, R\}$. We denote the cluster-specific probability of getting value r for feature q by θ_{rq}, such that $\sum_{r=1}^{R} \theta_{rq} = 1$. We adopt a $Dirichlet\{\beta_{1q}, \ldots, \beta_{Rq}\}$ prior for $\theta_{1q}, \ldots, \theta_{Rq}$. The Dirichlet prior hyperparameter, β_{rq} is taken as 0.5. Denote the resulting prior probabilities as b_{rq}

Assuming independence between features, the probability that x_i arose from cluster k

$$p(c_i = k|x_{1:i}, c_{1:(n-1)}, b_{1q}, \ldots, b_{Rq}, a) = \prod_{q=1}^{Q} \frac{(\sum_{j=1}^{n_k} \mathbb{1}(x_{j,k} = x_i)a) + (b_{rq}(1-a))}{(n_k a) + (1-a)} \tag{10}$$

This is a weighted average of the empirical probability of the value of x_i in the cluster j, and the prior probability.

Running specifications

The algorithm was implemented in MATLAB and run on a Linux machine running Fedora 22 with 8GB of RAM, and an 8-core Intel Core i7 processor. Due to time restrictions the algorithm was left to run for relatively few iterations (350) of the Gibbs sampler. Furthermore, the iterations were run with just $m = 10$ particles. Although Griffin[7] found the particle filter algorithms to run successfully with relatively few particles, we feel 10 may be too few. The first half of the MCMC chain was discarded as burn-in.

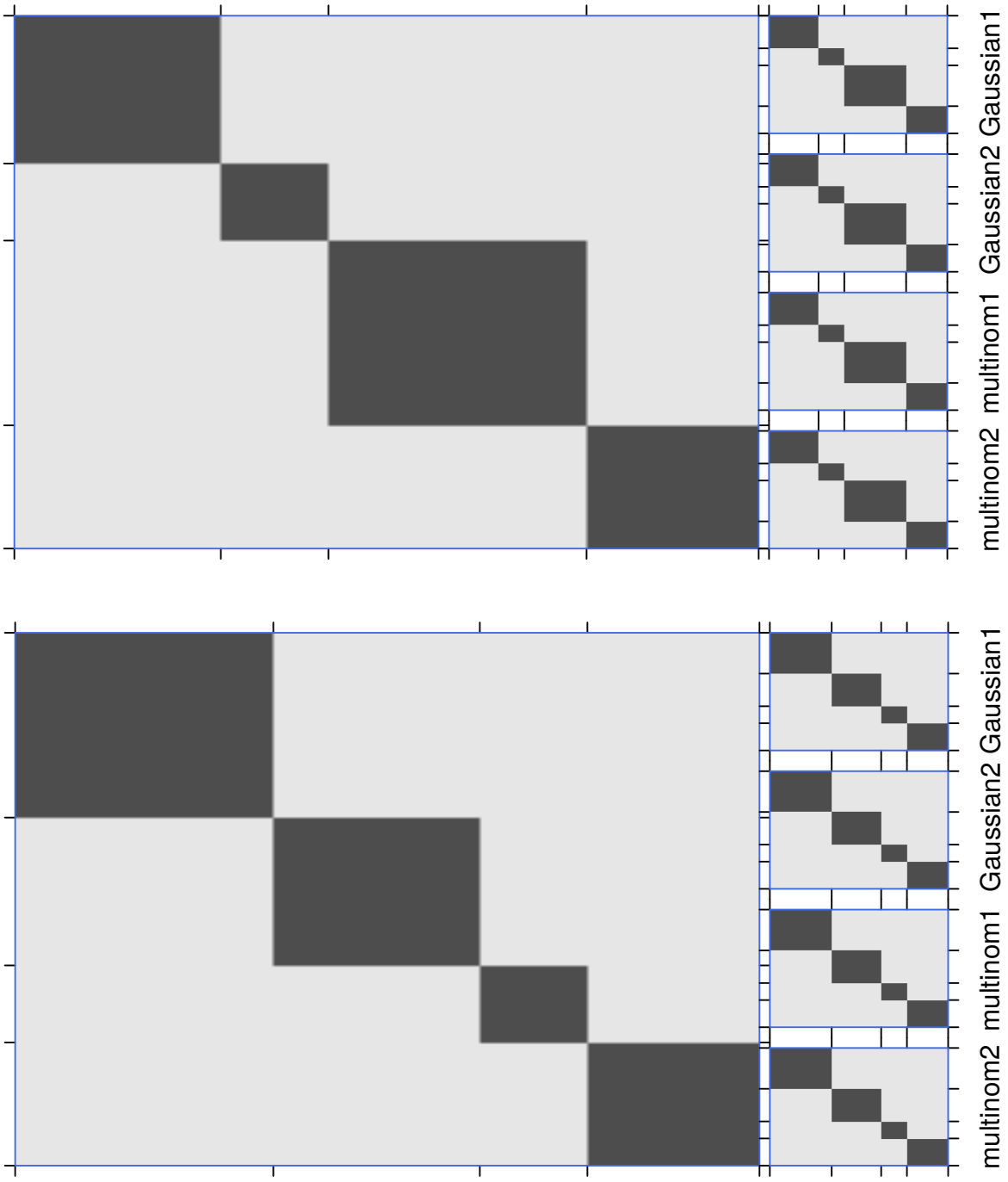

Figure 2: The results of running the pMDI algorithm (top) and the MDI algorithm (bottom) on four example datasets: two gaussian datasets and two multinomial datasets. The four panels down the right represent the clustering allocation suggested for each dataset individually. With observations running along both axes, the darkness of a square indicates the frequency with which two clusters are assigned to the same allocation across iterations of the algorithm. The ticks on the axes indicate the inferred cluster partition. The main panel represents the consensus agreement across datasets. The allocations inferred in each plot are identical despite the different orderings of the observations.

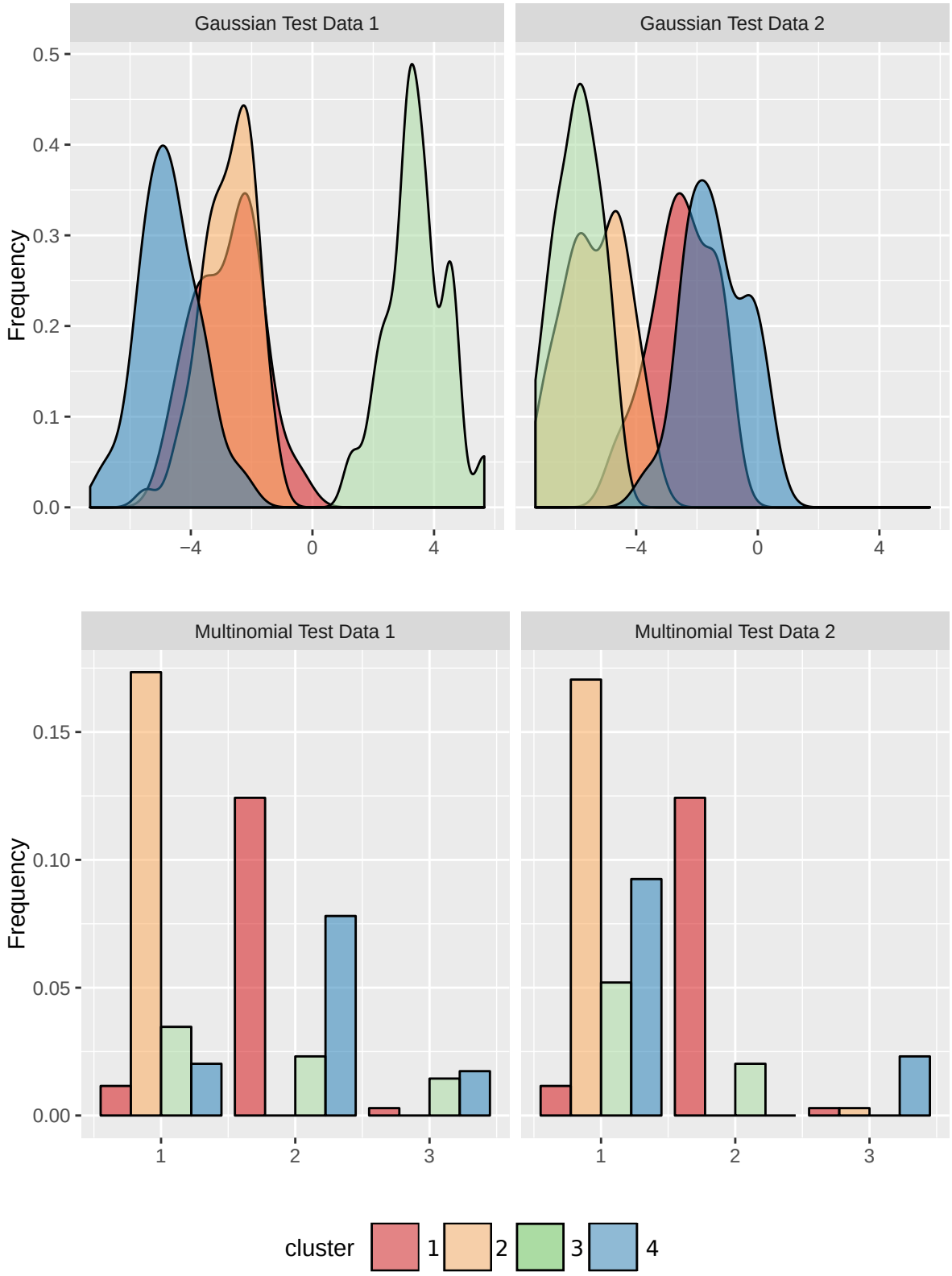

Figure 3: One feature from the simulated data used to test the MDI algorithm. The benefit of sharing clustering information across datasets can be seen by considering clusters 1 & 2 in `Gaussian Test Data 1`: given the overlap, it would be difficult to distinguish between the clusters without the added information gained from the additional datasets.

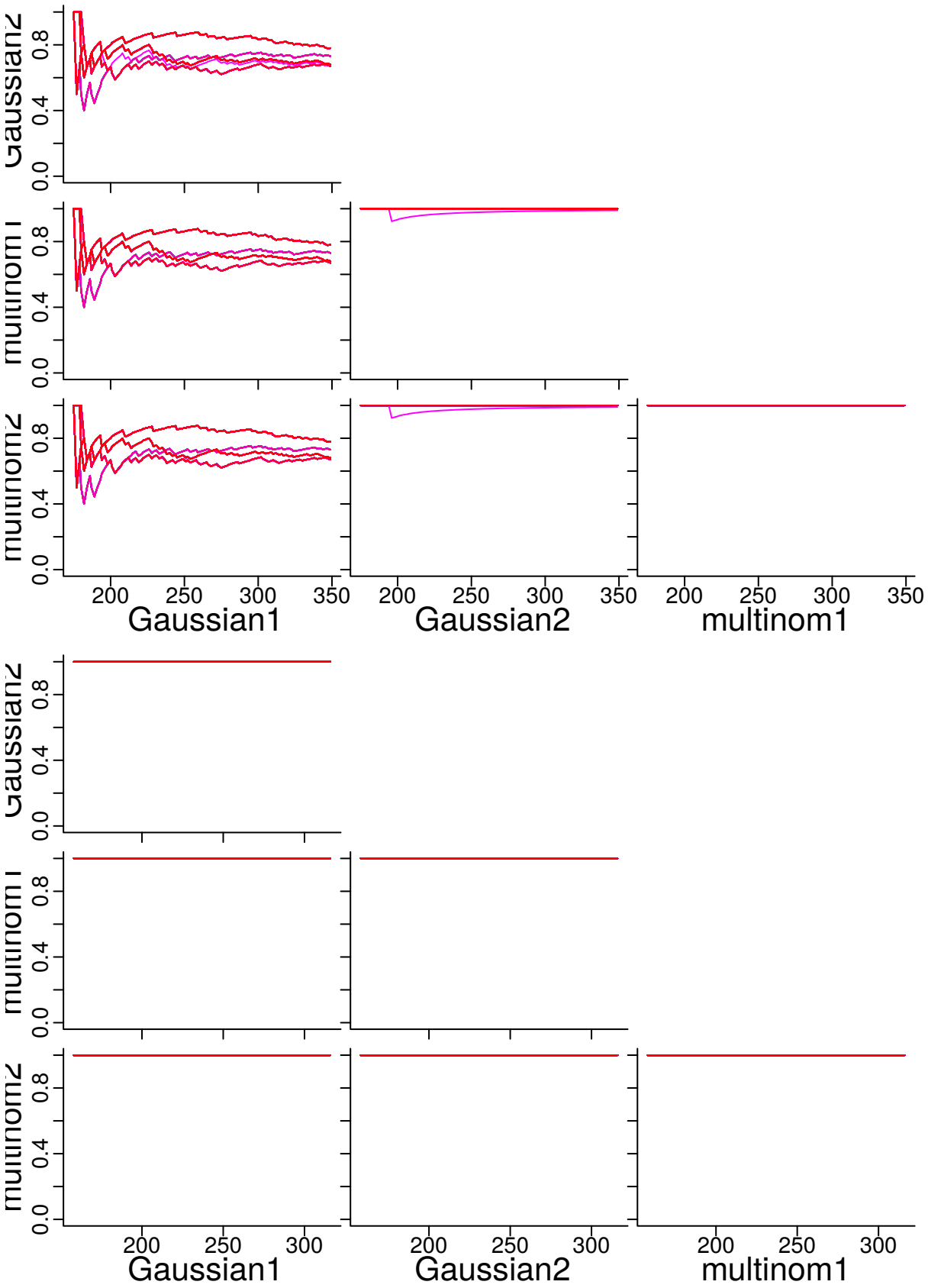

Figure 4: The pairwise 'fusion' across datasets in the pMDI algorithm (top) as compared with the original MDI (bottom). Lines indicate the cumulative fusion of individual genes in the two datasets. Code to generate plots sourced from [10].

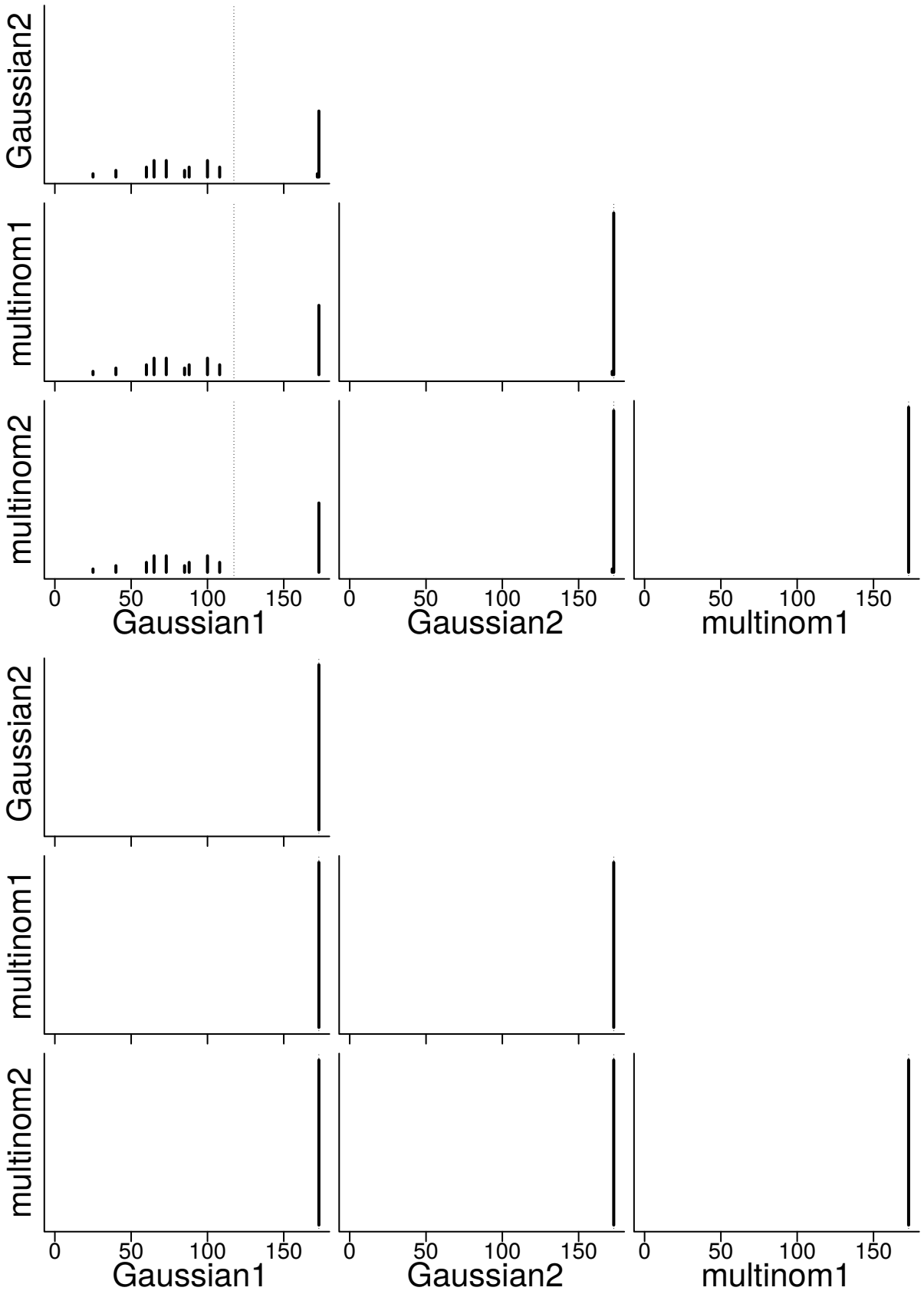

Figure 5: Posterior mass associated with the number of fused genes across datasets in pMDI (top) and the original MDI (bottom). The x-axis represents the number of fused genes, the y-axis is the relative frequency. The dashed line represents the mean number of fused genes. Code to generate plots sourced from [10].