Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

User-Informed Clustering for Cell Population Identification in Cytometry and Sequential Outlier Identification for Gaussian Mixture Modelling

Ultán P. Doherty

Supervised by Dr. Arthur White

and co-supervised by Prof. Rachel M. McLoughlin.

A thesis submitted to Trinity College Dublin, the University of Dublin in partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Discipline of Statistics and Information Systems,
School of Computer Science and Statistics

2026

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Signed: Ultán Doherty Date: 14/01/2026

Summary

In this thesis, we review a broad variety of automated methods for unsupervised and user-informed cell population identification in flow cytometry, then we present a novel user-informed clustering method for this problem and extend it using a constrained mixture model approach. Next, we introduce a novel method for sequentially identifying outliers while fitting Gaussian mixture models, before extending this to an analogous method for Gaussian linear cluster-weighted models.

Flow cytometry is a technique employed in immunology to study the impact of diseases and treatments at a single-cell level by indirectly measuring each cell's expression levels for a range of protein markers. Cell population identification is the process of grouping together sets of cells with similar protein marker expression levels in order to investigate how specific types of cells are affected by a disease or treatment.

We begin by reviewing model-based clustering methods that have been developed and adapted to suit the problem of cell population identification in flow cytometry. Model-based methods cluster data by fitting a mixture of probability distributions to the data set and computing the data points' probabilities of being sampled from each component of the mixture. We proceed to discuss a selection of non-model-based methods which implement various centroid-based, non-parametric density-based, and hierarchical clustering techniques. Then, we examine a handful of methods which have been developed to identify cell populations with the aid of user-provided descriptions of those populations.

Following on from our review of existing user-informed methods for cell population

identification, we introduce a novel method for identifying cell populations based on user-provided descriptions. gateTree constructs a constrained binary tree based on a simple table describing each targeted population as positive, negative, or undefined for every marker. We demonstrate gateTree's strong performance when applied to a range of simulated and real data sets. Then, we implement a constrained mixture model approach to extend a gateTree solution and identify populations which have not been described by the user.

Next, we develop a novel method for sequentially performing outlier identification while fitting Gaussian mixture models. outlierMBC chooses an optimal number of low-density points to classify as outliers based on the distribution of each component's scaled squared sample Mahalanobis distances. We apply this method to simulated and real data sets and compare it to leading outlier identification methods for Gaussian mixture modelling. We also extend this approach to the context of Gaussian linear cluster-weighted models, which combine multivariate clustering with multiple linear regression.

Finally, we discuss some ideas for future extensions of our methods and potential areas of further research.

Abstract

Whether obscured by scientifically insignificant noise or distorted by the presence of outliers, the true group structure of a data set can often be difficult to uncover in real-world data. In this thesis, we examine a challenging setting in which clustering algorithms have been applied to solve a real-world problem, that of cell population identification in flow cytometry. These data sets present a variety of challenges which can prevent clustering algorithms from identifying biologically meaningful populations. We discuss some of these issues in the context of how model-based clustering methods have been adapted to try to overcome them. Model-based clustering is a statistical, probabilistic approach for data clustering which models heterogeneous data using a mixture of known probability distributions with parameters estimated based on the data. Next, we review several non-model-based methods which have been developed for this problem, and a handful of methods which have been designed to utilise expert knowledge about the cell populations targeted by the researcher. Taking advantage of immunologists' knowledge of the cell populations in their data allows a clustering algorithm to focus on biologically significant differences between cells and to construct a set of cell populations which suits the context of the experiment and the needs of the researcher. We develop our own user-informed clustering method for flow cytometry, gateTree, and demonstrate its ability to accurately identify populations that have been described in simple and biologically meaningful terms. We then extend this user-informed approach to constrained mixture models and discuss their strengths and weaknesses. In the latter part of this thesis, we broaden our focus to examine the general problem of outliers in Gaussian mixture modelling, rather than exclusively dealing with flow cytometry data. We present

outlierMBC, a novel method for sequentially identifying outliers in Gaussian mixture models, and then extend it to Gaussian linear cluster-weighted models. We demonstrate its ability to accurately identify outliers and compare it to a range of leading methods for model-based outlier identification. The novel methods developed and presented in this thesis share a common aim to circumvent some of the noisiness and complexity present in real-world data sets and uncover a meaningful and practical clustering solution.

Acknowledgements

I would like to begin by acknowledging the support and guidance of my supervisor, Dr. Arthur White. I am particularly grateful for Dr. White's patience, optimism, and encouragement throughout my postgraduate studies, as well as for the knowledge he has imparted during my undergraduate degree, a summer internship, and four years of research.

I am thankful to my secondary supervisor, Prof. Rachel M. McLoughlin for her expertise, enthusiasm, and insights. Prof. McLoughlin's counsel on immunology and flow cytometry was invaluable for my research.

I would like to thank Prof. Paul D. McNicholas for his hospitality during my research visit to McMaster University, as well as his advice and contributions during our ensuing collaboration. I am also grateful to the students in Prof. McNicholas' research group for welcoming me during my time in Hamilton.

Thank you to Dr. Boris P. Hejblum and Dr. Jason Wyse for examining my thesis and to Prof. Ciaran Mc Goldrick for chairing my viva. I thoroughly enjoyed discussing my research with you all.

I greatly appreciate the feedback and direction given by my thesis committee, Dr. Jason Wyse and Dr. Conor M. Finlay.

I am grateful to Tracey J. Claxton and Dr. Charlotte M. Leane for creating and manually gating the haemodialysis and *S. aureus* data sets featured in Chapter 5, as well as for their helpful explanations of these data.

Thanks also to all of the statistics students and faculty in Trinity College Dublin for their company and camaraderie in the Lloyd Building over the last four years.

I would like to acknowledge Research Ireland – Taighde Éireann for funding my research through a Government of Ireland Postgraduate Scholarship.

I am profoundly grateful to my parents and my brother, Iarlaith, for the support and encouragement which has made my research possible. Thanks also to Iarlaith for his help with proofreading this thesis.

I am thankful for the backing of my friends and extended family.

Finally, thank you to Nina Chen for her love and support during my undergraduate and postgraduate studies.

Publications

The following articles, arising from this work, are published or available online.

Doherty, U. P., McLoughlin, R. M., & White, A. (published, 2025). Challenges and Adaptations of Model-Based Clustering for Flow and Mass Cytometry. WIREs Computational Statistics.

Doherty, U. P., McLoughlin, R. M., Claxton, T., & White, A. (online, 2024). gateTree: A user-informed tree algorithm for population identification in flow cytometry. Proceedings of the 38th International Workshop on Statistical Modelling.

Doherty, U. P., McNicholas, P. D., & White, A. (online, 2025). Model-Based Sequential Outlier Identification for Linear Cluster-Weighted Models. Proceedings of the 39th International Workshop on Statistical Modelling.

The following articles, also arising from this work, are in the final stages of preparation.

Doherty, U. P., Leane, C. M., Claxton, T. J., McLoughlin, R. M., & White, A. gateTree: A user-informed tree algorithm for cell population identification in flow cytometry.

Doherty, U. P., McNicholas, P. D., & White, A. Model-Based Clustering with Sequential Outlier Identification using the Distribution of Mahalanobis Distances.

Contents

1 Introduction **1**

1.1 Background 1

1.1.1 Clustering 1

1.1.2 Cell Population Identification in Flow Cytometry 2

1.1.3 User-Informed Clustering 3

1.1.4 Model-Based Clustering 4

1.2 Research Aims 5

1.3 Research Contributions and Impacts 6

1.3.1 A Review of Model-Based Clustering for Cytometry 6

1.3.2 Novel Methods for User-Informed Clustering 7

1.3.3 Novel Methods for Identifying Outliers in Model-Based Clustering . 8

1.4 Thesis Structure 9

2 Model-Based Clustering for Flow Cytometry **13**

2.1 Introduction 13

2.1.1 Overview 13

2.1.2 Mass Cytometry Data Example 16

2.2 Model-Based Clustering 18

2.2.1 Background 18

2.2.2 Advantages of Model-Based Clustering 23

2.3 Non-Gaussian Populations: Alternative Models 24

2.4 Multiple Samples: Hierarchical Models 26

2.5 Unknown Populations: Model Selection 30
2.6 Rare Populations: Novel Model Fitting 33
2.7 Large Data Sets: Binning and Factor Analysis 35
2.8 Other Challenges Related to Manual Gating 36
2.9 Conclusion 38

3 Non-Model-Based Methods for Flow Cytometry 43
3.1 Introduction 43
3.2 Methods 44
3.2.1 FLOCK 44
3.2.2 flowMeans 45
3.2.3 SPADE 46
3.2.4 FlowSOM 47
3.2.5 PhenoGraph 47
3.2.6 X-Shift 48
3.2.7 cytometree 48
3.3 Conclusion 49

4 User-Informed Methods for Flow Cytometry 53
4.1 Introduction 53
4.2 Methods 55
4.2.1 flowDensity 55
4.2.2 ACDC 56
4.2.3 Bayesian Trees with Mondrian Process Priors 57
4.2.4 GateMeClass 60
4.3 Conclusion 64

5 A User-Informed Tree Algorithm for Flow Cytometry 65
5.1 Introduction 65
5.2 Methodology 67
5.2.1 Recursive Univariate Splitting 67

5.2.2 User-Informed Tree Constraints . . . 69
5.2.3 Multi-Sample Information Propagation . . . 72
5.2.4 Event Trimming . . . 73
5.2.5 Visualisation . . . 74
5.2.6 Related Methods . . . 76
5.3 gateTree Algorithms . . . 78
5.3.1 Setup . . . 78
5.3.2 Split Variable Choice . . . 78
5.3.3 Split Location Estimation . . . 79
5.4 Application and Results . . . 83
5.4.1 Simulated Data . . . 83
5.4.2 Benchmark Data: Levine32 . . . 86
5.4.3 Novel Experimental Data: Haemodialysis . . . 89
5.5 *Staphylococcus aureus In Vivo* Infection Experiment . . . 91
5.5.1 Experiment Details . . . 91
5.5.2 gateTree Application . . . 92
5.5.3 gateTree vs Manual Gating Comparison . . . 92
5.5.4 Differential Analysis . . . 94
5.6 Conclusion . . . 99
5.6.1 Discussion . . . 99
5.6.2 Extensions . . . 100

6 Constrained Gaussian Mixtures for Flow Cytometry 103

6.1 Introduction . . . 103
6.2 Methodology . . . 104
6.3 Levine32 Data Application . . . 107
6.3.1 Identifying Target Populations . . . 107
6.3.2 Identifying Unknown Populations . . . 108
6.4 healthyFlowData Application . . . 110
6.5 Conclusion . . . 114

7 Sequential Outlier Identification for Gaussian Mixture Models 117

7.1 Introduction . . . 117

7.2 Outliers in Model-Based Clustering . . . 118

7.3 Methodology . . . 120

7.3.1 outlierMBC . . . 120

7.3.2 Related Methods . . . 121

7.3.3 Sample Mahalanobis Distance Dissimilarity . . . 123

7.3.4 Gross Outliers . . . 123

7.3.5 backtrack Mechanism . . . 124

7.3.6 Initialisation . . . 124

7.4 Illustrative Examples . . . 126

7.5 TCLUST Simulation . . . 128

7.5.1 TCLUST Simulation Setup . . . 128

7.5.2 TCLUST Simulation Results . . . 128

7.6 OCLUST Simulation . . . 132

7.7 Swiss Banknote Data . . . 134

7.8 healthyFlowData . . . 136

7.9 Conclusion . . . 139

8 Sequential Outlier Identification for Linear Cluster-Weighted Models 141

8.1 Introduction . . . 141

8.2 Method . . . 142

8.3 Simulated Data Example . . . 143

8.4 Real Data Example with Simulated Outliers . . . 145

8.5 Conclusion . . . 147

9 Conclusion 149

9.1 Summary . . . 149

9.1.1 Overview . . . 149

9.1.2 User-Informed Cell Population Identification . . . 150

9.1.3 Sequential Outlier Identification for Gaussian Mixtures 151
9.2 Further Work 152
9.2.1 Two-Dimensional Splitting in gateTree 152
9.2.2 gateTree Split Location Shrinkage 154
9.2.3 Trimodal Splitting in gateTree 156
9.2.4 gateTree-Informed Bayesian Priors 157
9.2.5 Combining gateTree and outlierMBC via Constraints 157
9.2.6 Non-Gaussian Extensions of outlierMBC 158
9.2.7 Robust Model Parameters in outlierMBC 159
9.2.8 Model Selection for outlierMBC 159
9.2.9 Reductions in Computational Time 161

A1 Distribution of Scaled Squared Sample Mahalanobis Distances 179

A1.1 Summary 179
A1.2 Definitions 179
A1.3 Proof Outline 180
A1.4 Lemmas 181
A1.5 Theorem Proofs 186

List of Figures

2.1 This pairs-style plot displays three of the thirty-two variables from the Levine32 data set (Weber and Soneson, 2019). These variables correspond to the surface expression of the CD19, CD8, and CD3 markers, which are important for distinguishing between the cell populations in this data set. Only events from the seven largest manually identified cell populations or gates are coloured. The unassigned events are plotted in the background as black points, while events from the seven smaller gates are shown in grey. 19

2.2 This scatter plot displays a reduced dimension representation of the Levine32 data set (Weber and Soneson, 2019). There are thirty-two variables in this data set and fourteen manually identified cell populations or gates. Events assigned to the seven largest gates are coloured. The unassigned events are plotted in the background as black points, while events from the seven smaller gates are shown in grey. 20

2.3 Three CD3 vs CD19 scatter plots of a flow cytometry sample from the healthyFlowData R package (Azad, 2013). The plots show how the data is clustered by a three-component, bivariate Gaussian mixture model, implemented by the mclust R package (Fraley et al., 2022). They display the two stages of fitting a mixture model: estimating each cluster's distribution and assigning each event to the cluster with the most suitable distribution. In the left scatter plot, the data is uncoloured but three distinct clusters are visible. In the middle scatter plot, three coloured, concentric ellipses are imposed over each cluster, showing the 33^{rd}, 66^{th}, and 99^{th} percentile level sets of each cluster's density according to a Gaussian distribution. In the right scatter plot, the three clusters are coloured and the CD3+CD19− and CD3−CD19+ clusters are labelled as "T Cells" and "B Cells", respectively. 22

2.4 Four CD4 vs CD8 scatter plots of healthyFlowData (Azad, 2013) samples from four individuals. The first sample from each subject is shown. These plots illustrate that corresponding cell populations from different subjects in the same study can have different shapes, sizes, and locations despite being analysed in the same way. Each plot is coloured according to a four-component Gaussian mixture model implemented by the mclust R package (Fraley et al., 2022). Three concentric ellipses are imposed over three of the four clusters. These are the 33^{rd}, 66^{th}, and 99^{th} percentile level sets of the Gaussian density for the clusters identified as "B Cells", "CD4+ T Cells", and "CD8+ T Cells". 27

2.5 Three plots illustrating how flowMatch (Azad et al., 2012) matches clusters from different samples and constructs a template Gaussian mixture model with one component per meta-cluster, or set of clusters. A four-component Gaussian mixture model was fitted to the first sample of each of the four subjects from the healthyFlowData R package (Azad, 2013) using the mclust R package (Fraley et al., 2022). Three components from each sample's mixture model were provided to flowMatch. Left: 66$^{\text{th}}$ percentile ellipses for twelve components from four samples are overlaid on the same plot. Middle: The same twelve ellipses coloured as three sets of four clusters according to a meta-cluster assignment by flowMatch. Right: 66$^{\text{th}}$ percentile ellipses for the three template components constructed by flowMatch for each meta-cluster. 29

2.6 A graphical summary of the challenges and adaptations of using model-based clustering for flow cytometry. Five challenges (green) are centred around "Model-Based Clustering for Cytometry" (pink). Two adaptations (purple) arise from each of these challenges. 39

4.1 Plots of the Mondrian tree method's beta prior distributions in the different settings outlined in Table 4.3, using the values of ϕ_0 and ϕ_1 mentioned in Ji et al. (2018). 59

4.2 A plot of the probability density functions of three hypothetical univariate Gaussian distributions to illustrate GateMeClass' cluster reassignment of events. The vertical dashed lines mark the component means and divide the x-axis into four intervals. Each interval has one or two reassignment rules, shown below the x-axis. For example, in the left-hand interval, $(-\infty, \hat{\mu}_1)$, events assigned to components 2 or 3 of the Gaussian mixture model are reassigned to cluster 1. 63

5.1 The tree constructed by gateTree for the Levine32 data and the population description table that it was informed by. 71

5.2 Plot of the gateTree valley split and kernel density estimate of the CD19 marker for the live lymphocytes of the fifth Rechallenged sample in the *S. aureus* experiment discussed in Section 5.5. 75
5.3 Scatter plots of a five-dimensional sample from the simulation study showing the bimodal and nested or telescoping nature of the simulated populations. 84
5.4 A line plot with error bars illustrating the cross-sample means of the unweighted sample mean F_1 values for every simulation scenario and method. 86
5.5 Column plots of each method's F_1 values per gate for the two Levine32 samples. For both samples, each method's unweighted mean F_1 value across populations is displayed as a horizontal line. 88
5.6 Visual comparison of the order in which the first nine channels were used by manual gating versus by gateTree for the *S. aureus* data. 93
5.7 IL-17A vs Vγ4 scatter plots of the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified by gateTree and manual gating. It can be observed, particularly for the Convalescent and Rechallenged groups, that some cells with mid Vγ4 expression levels have been assigned to the Vγ4 $\gamma\delta$ T-cell population by the manual gating and to the Vγ6 $\gamma\delta$ T-cell population by gateTree. . 95
5.8 Scatter plots of the sample population proportions and sample mean IL-17A values for the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified via manual gating and by gateTree. 97
5.9 Column plots of the adjusted p-values obtained from pairwise Wilcoxon rank-sum tests using the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified via manual gating and by gateTree. The negative base-10 logarithm of each adjusted p-value is plotted. 97

6.1 F_1 scores for constrained and unconstrained Gaussian mixture models for each medium-sized gate when the constrained set corresponding to that cell population is excluded. The green horizontal line displays the F_1 value achieved by gateTree for this population as a benchmark, however, none of the three models being evaluated were given information about the desired population, unlike gateTree. 110
6.2 Generalised pairs plot of healthyFlowData Sample A1 coloured according to gateTree. 112
6.3 Generalised pairs plot of healthyFlowData Sample A1 coloured according to a six-component constrained GMM with three pre-labelled constrained sets from gateTree. Every event from the gateTree populations was included in the constrained sets ($\texttt{zone\_percent} = 100$). 112
6.4 Generalised pairs plot of healthyFlowData Sample A1 coloured according to a six-component unconstrained GMM. 114

7.1 A pair of scatter plots for two variables from the six-dimensional Swiss Banknote data set with points coloured according to clustering solutions from two-component Gaussian mixture models. The two plots compare the solution without identifying outliers (left) to the outlierMBC solution (right) which identifies twenty outliers. Density level sets, in the form of ellipses, are displayed for each component. 119
7.2 Scatter plots of the illustrative data sets with their points coloured according to the outlierMBC solutions and with outlier classification errors circled in yellow. 127
7.3 Dissimilarity curves for the two illustrative data sets. The dissimilarity values on the y-axis have been rescaled so that the minimum has a value of 1. . 127
7.4 Scatter plots for the two-dimensional models (seed $= 1$) with equal component sizes. Observations' colours correspond to their simulation source and their shapes correspond to the outlierMBC-minimum classification. 130
7.5 Clustering and outlier identification solutions for the s3 data set. 135

7.6 Dissimilarity curve for outlierMBC applied to the subject A, sample 1 data set ($n = 19,321$) from the healthyFlowData R package. Our minimum mechanism chose the number of outliers to be 1204, while our 'backtrack' mechanism selected 1089. 137

7.7 Pairs plot of the subject A, sample 1 data set ($n = 19,321$) from the healthyFlowData R package. Events are coloured according to a four-component Gaussian mixture model with 1089 outliers identified by outlierMBC using our 'backtrack' mechanism with 2000 as the maximum number of outliers. 137

8.1 Scatter plot of the 120 simulated true points and 12 simulated outliers. It is coloured according to the outlierMBC-LCWM backtrack solution which has correctly identified all 12 outliers with no false positives. The minimum solution instead chose to remove 13 points, with one false positive. 144

8.2 Scatter plot of the pine data set after removing 12 data points with extreme values (Diameter $>$ 335 or Height $>$ 22), then adding 15 simulated outliers. It is coloured according to the outlierMBC-LCWM solution which has perfectly identified the 15 simulated outliers. The minimum and backtrack solutions are identical for this data set. 145

8.3 Dissimilarity curves for outlierMBC-LCWM applied to the modified pine data set displayed in Figure 8.2 after removing five gross outliers using our `find_gross` function. The solid black line represents $\{\hat{D}_m\}_{m=6}^{30}$ (Equation 8.7), while the dotted blue line and the dashed purple line represent $\left\{\hat{D}_m^{(U)}\right\}_{m=6}^{30}$ and $\left\{\hat{D}_m^{(V)}\right\}_{m=6}^{30}$ (Equation 8.8). 146

9.1 Scatter plots of uniformly distributed points coloured according to whether they would have a value of greater than 0.5 under a series of bivariate to univariate transformations. 155

9.2 BIC plot for the Swiss Banknote data set, comparing one-, two-, three-, and four-component models. According to this graph, if less than fifteen potential outliers are removed, then the three-component model has a better BIC, but once fifteen or more are removed, then the two-component model has a better BIC. This illustrates the connection between component number and outlier number, since the third component consists primarily of low-density observations which can be deemed to be outliers. 160

List of Tables

2.1 This table lists model-based clustering methods which have been designed specifically for flow cytometry and/or mass cytometry. For each method, the probability distribution of its mixture model components is listed, its approach for multiple samples is mentioned, and additional details are provided. 17

3.1 The non-model-based methods discussed in Chapter 3. 43

3.2 The parameters required by the non-model-based methods, both data-driven and user-specified. 51

4.1 The user-informed methods discussed in Chapter 4. 54

4.2 A basic example of a 'cell type-marker' table of the format introduced by Lee et al. (2017). Here, we are stating that the B-cell population has a positive expression level for the CD19 marker but a negative expression level for the CD3 marker. We have left the B-cell population's expression levels for the CD4 and CD8 markers undefined. We denote such a table by $\mathcal{T}$. . 55

4.3 The values of the three hyperparameters (right) for different combinations of signs which are present for a given marker (left). 58

5.1 The population means and sizes for the three-dimensional (left) and six-dimensional (right) simulated data sets. 84

5.2 Each method's unweighted mean F_1 scores for the two Levine32 samples. 88

5.3 The five population description tables used by gateTree for each pathway of the haemodialysis data. 90

5.4 The unweighted mean F_1 scores for each method and population across all twenty-four samples from the haemodialysis data. 90
5.5 The experimental conditions for each of the four groups in the *S. aureus* data. 91
5.6 The population descriptions used by gateTree for the *S. aureus* data. . . . 92
5.7 Order in which channels were used by manual gating versus by gateTree for the *S. aureus* data. 93
5.8 Mean size, precision, and recall values for gateTree across samples within each experimental group from the *S. aureus* data. 94

6.1 Mean F_1 values across the eight target populations for gateTree, an unconstrained Gaussian mixture model, and two constrained GMMs. The number of mixture components and the size of the constrained sets were both varied. 109
6.2 Each model's best F_1 value for the four cell populations without constrained sets and the number of components used to achieve that solution. 109
6.3 The cell population descriptions used to inform gateTree when applying it to healthyFlowData Sample A1. 111
6.4 Contingency table comparing the unconstrained six-component GMM solution to the gateTree solution. 113
6.5 Contingency table comparing the constrained six-component GMM solution to the gateTree solution. 113

7.1 Setting options for the TCLUST Simulation. 129
7.2 Component mean vectors and covariance matrices for the TCLUST simulation study. The values of the covariance constants a, b, c, d, e, and f are provided in Table 7.3. 129
7.3 Covariance constants, (a, b, c, d, e, f), for the TCLUST simulation study. 129

7.4 Mean values for the TCLUST simulation study of Adjusted Rand Index (ARI), F_1 Measure with respect to outlier classification only (Outlier F_1), and Numbers of False Positives (#FP), False Negatives (#FN), and Outliers (#Outliers). 131
7.5 Sizes and properties of the data sets from the OCLUST simulation study. 132
7.6 ARI values from the OCLUST simulation study. 133
7.7 Outlier F_1 values from the OCLUST simulation study. 133
7.8 Number of false positives from the OCLUST simulation study. 133
7.9 These tables show each method's cluster assignment of the genuine banknotes (left) and the counterfeit banknotes (right) from the Swiss Banknote data set. 136
7.10 Contingency table comparing the four-component outlierMBC solution to the gateTree solution from Section 6.4. 138

8.1 Simulation information for the data set in Section 8.3. 144

9.1 A potential format for a table containing the information required for the user to describe multiple populations in terms of several variable pairs. The details in this table are for illustrative purposes only and are not based on a real data example. It may be beneficial to allow the user to specify some of the angles themselves, as we have done for the CD19 splits in the table. 154

1 Introduction

1.1 Background

1.1.1 Clustering

Data plays an increasingly prominent role in modern society. As we become ever more reliant on data, both in everyday life and in scientific research, our ability to interpret and analyse complex data sets is critical to ensuring that we make correct decisions and draw accurate conclusions. In many real-world applications, observations are not homogeneously distributed within a data set. Instead, there may be multiple distinct subgroups of data points. Clustering algorithms aim to uncover this group structure directly from the unlabelled data, while assigning those unlabelled data points to these groups. Since it does not learn a known class structure from labelled training data, clustering is an unsupervised learning approach, unlike its supervised counterpart, classification. The groups uncovered by a clustering algorithm are referred to as clusters, however, there is no universal agreement on how to define a cluster. Some definitions involve connected regions of high density separated by regions of low density, while others focus on points in the same cluster being more similar than points in different clusters. Although competing definitions of what constitutes a cluster may seem similar and often align in simple scenarios, they give rise to distinct approaches which can diverge to a surprising degree in more complicated settings. Hennig (2015) offers a discussion on various cluster definitions and clustering philosophies.

1.1.2 Cell Population Identification in Flow Cytometry

Flow cytometry and mass cytometry are high-throughput single-cell analysis techniques used in immunological research to indirectly measure the expression level of a range of protein markers for each cell in a tissue sample. Researchers study the impact of diseases at a cellular level in the search for novel and effective treatment options. In flow cytometry, the cells are stained with fluorescent antibodies and passed through a series of lasers in single file, then the fluorescence emitted by each cell is measured. Each fluorescent antibody consists of a unique combination of an antibody, which binds to a specific protein marker, and a fluorochrome, which emits a certain fluorescence wavelength when excited by a suitable laser. This allows immunologists to associate the intensity of a particular fluorescent wavelength with the expression level of a corresponding protein marker. In mass cytometry, the fluorochromes are replaced with heavy-metal isotopes and instead of measuring fluorescence, the time-of-flight in a mass spectrometer is measured. Individual observations in flow and mass cytometry are often referred to as events, instead of cells, since they may also consist of debris or a clump of multiple cells.

By identifying groups of cells, or cell populations, with similar protein marker expression levels, immunologists can compare the abundance and expression levels of specific cell types and subtypes between subjects in different experimental groups. For example, if they observe that subjects infected with a disease have lower numbers of a certain cell type than healthy patients, then they may infer that the disease is causing the deaths of these cells, such as the effect of HIV on CD4+ T Cells. Often the cell populations being identified are well-established in the immunological literature and when experts manually group events together through a process called gating, they do so based on these populations' expected characteristics rather than based solely on the distribution of the data. Moreover, when planning a cytometry experiment, immunologists usually select protein markers to analyse which will enable them to isolate cell types of interest, rather than choosing markers based on an external criterion and searching naïvely for clusters of cells. This enables immunologists to build on previous research into these cell populations.

By providing insights at a cellular level into the mechanisms through which diseases act, flow cytometry serves as an invaluable tool for disease research and the development of future treatments.

1.1.3 User-Informed Clustering

Despite the development of a variety of cell population identification algorithms, the uptake of automated methods in the cytometry community has been limited (Cheung et al., 2021). Manual gating remains the most popular method of identifying cell populations in flow cytometry data today, despite well-documented issues with its reproducibility and concerns about subjectivity and bias, as well as the time-consuming nature of this task (Saeys et al., 2016). However, one aspect of cell population identification that sets it apart from general clustering problems is the immunological research community's extensive knowledge of the populations being identified. As well as possessing biological expertise from the research literature, experienced flow cytometry practitioners also have hands-on experience of manually assigning events to populations via gating. Given immunologists' apparent preference for manually identifying cell populations over using unsupervised clustering algorithms, one under-explored area of research is the development of methods which can use information about the cell populations being targeted by the researcher.

There are many classification methods that can be applied to flow cytometry data, which learn a class structure from a manually gated sample, before assigning the events of a new sample to these classes. These include both generic machine learning algorithms and cytometry-specific methods, such as DeepCyTOF (Li et al., 2017), flowLearn (Lux et al., 2018), and CytOpT (Freulon et al., 2023). However, when we discuss user-informed methods in this thesis, we are referring to approaches which do not require training data. In other words, the information about the cell populations with which they are provided does not come in the form of a manually gated sample. We have chosen to focus on these user-informed clustering methods, rather than classification algorithms, because the events in flow cytometry samples do not have a true 'ground truth' class, and therefore

any training samples must be manually gated. This means that the issues of subjectivity and irreproducibility that are inherent to the manual gating process could implicitly influence the classification of new samples through the gating of the training data. In particular, it would be difficult to interpret the effect of differences in the gating of the training sample on the classification of new data. Moreover, understanding the differences between how two distinct training samples were gated by two different immunologists would require an expert qualitative analysis of many carefully plotted visualisations. Hence, we have chosen to focus on methods which utilise user information that is provided in a transparent and interpretable manner. For example, a manual gating strategy (Malek et al., 2015) or a simple table of cell population descriptions (Lee et al., 2017; Ji et al., 2018; Caligola et al., 2024).

1.1.4 Model-Based Clustering

Model-based clustering approximates the heterogeneous distribution of observations within a data set using a collection of probability distributions called a mixture model. The probability distributions within a mixture model are referred to as mixture components. The density of observations in the data space is represented by the mixture density, a convex combination of the components' probability density functions with respect to their mixture proportions. The value of a component's mixture proportion is the prior probability of assigning an unseen data point to that component. The probability density function of a mixture model has the form

$$f(\boldsymbol{x}_i|\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i|\boldsymbol{\theta}_g), \tag{1.1}$$

where π_g is the mixture proportion for component g and $f_g(\boldsymbol{x}_i|\boldsymbol{\theta}_g)$ is the probability density function, f_g, of component g, parametrised by $\boldsymbol{\theta}_g$, and evaluated at a point $\boldsymbol{x}_i$. The model parameters and component assignment probabilities can be iteratively estimated using the Expectation-Maximisation (EM) algorithm (Dempster et al., 1977). Alternatively, a Bayesian approach would be to define prior distributions for the model

parameters and then estimate their posterior distributions given the data using Markov Chain Monte Carlo (MCMC) sampling. Fitting mixture models to data can be computationally intensive, particularly in high dimensions. However, factor analysis techniques can alleviate some of the computational burden by reducing the number of parameters for high-dimensional data. In contrast to more heuristic methods, model-based clustering's probabilistic foundations and statistical modelling framework more readily facilitate statistical inference (Gormley et al., 2023). Moreover, they provide an understanding of how the data are distributed and how they could have been generated.

1.2 Research Aims

The objective of this research is to develop novel clustering methods which can utilise user-provided information or assumptions to circumvent scientifically insignificant noise and uncover meaningful group structures in complex real-world data. In particular, we focus on developing a pair of methods for user-informed cell population identification in flow and mass cytometry and a pair of methods for sequential outlier identification in Gaussian mixture modelling. In the first setting, we take advantage of user-provided descriptions of the cell populations being targeted in order to overcome the wide variety of challenges associated with cytometry data sets and focus on the differences between events which are of scientific interest to the researcher. In the latter setting, we leverage the Gaussianity assumptions of Gaussian mixture models and Gaussian linear cluster-weighted models to derive beta distributions for the sample Mahalanobis distances and studentised residuals, allowing us to remove outliers in a methodical and cautious manner which is justified within a model-based, statistical framework. In both cases, information about the clusters, in the form of simple descriptions or Gaussianity assumptions, is used by our methods to allow them to construct appropriate clusters without being misled by outliers or other complex data issues.

1.3 Research Contributions and Impacts

1.3.1 A Review of Model-Based Clustering for Cytometry

The first contribution of this thesis is a review of model-based clustering methods which have been designed to address challenges associated with identifying cell populations in flow cytometry data. This work has been published in WIREs Computational Statistics under the title "Challenges and Adaptations of Model-Based Clustering for Flow and Mass Cytometry".

Automated cell population identification lies in the intersection between clustering, a well-established research topic in statistics and machine learning, and flow cytometry, a widely utilised technique in immunology. The interdisciplinary nature of this intersection can be daunting for newcomers to the area. As well as coming to terms with the prerequisite biological terminology and knowledge, a researcher seeking to contribute in this area will also be confronted with a vast array of clustering algorithms. Many of these methods have different origins and philosophies, depending on whether they arose from the statistical community or the computer science community. In this particular area, even algorithms from the same class of clustering method often differ considerably from each other as a result of the variety of cytometry-specific adaptations which have been developed to address challenges related to this data. This is particularly true of the model-based clustering approaches which have been adapted for cell population identification.

The probabilistic framework of model-based clustering lends itself to customisation by swapping the mixture components' probability distributions or modifying the model fitting process. Our review of the model-based clustering methods that have been developed for cell population identification offers a starting point for researchers entering this area. By providing an overview of the existing methods and how they relate to each other, this review can assist statistical researchers in identifying research gaps or building on previous developments when contributing new algorithms. Alternatively, it can aid researchers

working directly with cytometry data in selecting which of the existing algorithms is best suited to their applications. We believe that this review will have a positive impact by supporting research in this interdisciplinary field. Moreover, since flow cytometry is primarily used in immunological research, and specifically in understanding how diseases affect patients at a cellular level, any research in this field has the potential to contribute towards meaningful societal and medical advances.

1.3.2 Novel Methods for User-Informed Clustering

Our exploration of how to allow user information to guide cell population identification algorithms in flow cytometry has produced both a novel tree-based clustering method and an approach for constraining mixture models with respect to the tree's clusters. gateTree, our tree-based cell population identification algorithm is implemented as an R package which is available on GitHub (`github.com/UltanPDoherty/gateTree2`), and a manuscript introducing this method is in the final stages of preparation for submission to Cytometry Part A. We also implemented the positively constrained Gaussian mixture model described in Melnykov et al. (2016) in another R package on GitHub (`github.com/UltanPDoherty/mustlinkMix`).

The prevalence of manual gating demonstrates that immunologists working with flow cytometry data have extensive knowledge of the cell populations that they are trying to identify. Despite this, most of the algorithms which have been developed for cell population identification have been fully unsupervised. Our tree-based method utilises a table describing the cell populations being targeted (Lee et al., 2017) to inform the clustering process. We have demonstrated that this method performs well on simulated data, publicly available benchmark data, clinical human data, and *in vivo* infection data. By taking advantage of the biological knowledge possessed by the immunologists conducting the experiment, while also making data-driven choices, we hope that gateTree can provide a fruitful compromise between manual gating and fully unsupervised clustering algorithms and have a positive impact on future immunological research. Meanwhile, our constrained mixture model approach is able to build on the gateTree solution and extend

it to cell populations that have not been described by the user. In doing so, it has the potential to explore the data and discover novel cell populations.

1.3.3 Novel Methods for Identifying Outliers in Model-Based Clustering

This thesis' final contribution is a pair of novel methods for sequentially identifying outliers in model-based clustering. Our primary method is designed for Gaussian mixture models but we also developed an extension for Gaussian linear cluster-weighted models. These methods' approach does not require assumptions about the distribution of the outliers and it does not require the number of outliers to be pre-specified. Both methods are underpinned by a theoretical result about the beta distribution of scaled squared sample Mahalanobis distances, while the linear cluster-weighted model extension also relies on an analogous result involving scaled squared studentised residuals. Moreover, their sequential outlier removal is intuitive and is accompanied by interpretable graphical outputs. A paper introducing the Gaussian mixture model method and demonstrating its strong performance compared to other leading algorithms has been submitted to JCGS, the Journal of Computational and Graphical Statistics. Both algorithms are implemented in the outlierMBC R package, which is published on CRAN.

Since this work is not specifically tailored to cell population identification, it has the potential to have a more general impact in a wider array of applications and contexts. Outliers are known to have a particularly detrimental effect on model-based clustering methods and by contributing to the existing literature in this area, our novel method will have a positive research impact in mixture modelling.

1.4 Thesis Structure

Chapter 2: Model-Based Clustering for Flow Cytometry

This chapter is a review of the challenges and adaptations of applying model-based clustering methods to cytometry data. Identifying cell populations in flow cytometry data using automated algorithms is a challenging problem which has motivated researchers for almost two decades. Many different approaches from the fields of statistics, machine learning, and data science have been adapted for use on flow cytometry data. The model-based clustering community has shown a particular interest in flow cytometry data as a motivating example. As a result, there have been many cytometry-specific adaptations developed for mixture models, which are explored in this chapter.

Chapter 3: Non-Model-Based Methods for Flow Cytometry

In this chapter, we discuss a selection of the leading non-model-based methods for cell population identification in cytometry. The selected methods were FLOCK (Qian et al., 2010), flowMeans (Aghaeepour et al., 2011), SPADE (Qiu et al., 2011; Qiu, 2017), FlowSOM (Van Gassen et al., 2015), PhenoGraph (Levine et al., 2015), X-Shift (Samusik et al., 2016), and cytometree (Commenges et al., 2018). These methods cover a range of non-model-based clustering approaches including k-means, self-organising maps, non-parametric density estimation, and hierarchical clustering. In particular, many of these algorithms innovatively combine elements of multiple non-model-based approaches. They demonstrate that despite the focus on model-based adaptations in Chapter 2, non-model-based methods can display similar adaptability.

Chapter 4: User-Informed Methods for Flow Cytometry

In this chapter, we examine four methods which utilise additional information provided by the user about the cell populations being identified. The methods discussed are flowDensity (Malek et al., 2015), ACDC (Lee et al., 2017), a Bayesian tree method with Mondrian Process priors (Ji et al., 2018), and GateMeClass (Caligola et al., 2024).

flowDensity mimics a user-provided manual gating strategy by implementing recursive univariate partitions while determining the split locations itself but partitioning the same variables as the manual gating, in the same order. ACDC and the Mondrian tree method are both informed by a 'cell type-marker table' which consists of one row per cell type and one column per marker and defines each cell type-marker pair to be positive $(+1)$, negative (-1), or undefined (0). GateMeClass extends this table to include an expression level by allowing four options for the table entries: '$*$' (any), '$+$' (positive/hi), '$-$' (negative/lo), and 'm' (medium/mid). These three methods are designed to identify subsets of events whose characteristics match the descriptions in their table. ACDC does so using a combination of a lattice-like simultaneous partitioning of the markers, the PhenoGraph algorithm (Levine et al., 2015), and random walk classification. GateMeClass also simultaneously partitions the data, then performs a k-nearest neighbour classification. Finally, Ji et al. construct binary trees to perform recursive univariate partitioning within a Bayesian framework.

Chapter 5: A User-Informed Tree Algorithm for Flow Cytometry

This chapter introduces gateTree, a novel method which constructs a constrained decision tree based on the table of user-provided population descriptions proposed by Lee et al. (2017) and adopted by Ji et al. (2018) to inform the priors for their binary trees. In contrast to the Bayesian framework in which Ji et al. (2018) designed their method, we build our tree algorithmically, allowing us greater flexibility and adaptability. For example, we incorporate univariate splitting mechanisms based on kernel density estimates and Gaussian mixture model boundaries, similarly to flowDensity (Malek et al., 2015) and cytometree (Commenges et al., 2018), respectively. Our use of straightforward constraints also makes it easier to interpret the effect of the user-provided table on the cell populations which are identified. In this chapter, we apply gateTree to simulated data and a range of real data sets and demonstrate its strong performance at replicating manual gating and identifying clinically relevant cell populations.

Chapter 6: Constrained Gaussian Mixtures for Flow Cytometry

In Chapter 6, following the strong performance of our constrained tree method in Chapter 5, we investigate the idea of constraining a mixture model. We build on gateTree's success by implementing pairwise constraints between events based on their population assignment by gateTree. We employ the positive constraints approach proposed by Shental et al. (2003) and Melnykov et al. (2016), and further discussed by Bouveyron et al. (2019), to keep each subset of events identified by gateTree together while fitting a Gaussian mixture model. We also implement another approach of pre-labelling the constrained events. One benefit of using a gateTree-constrained mixture model, instead of gateTree itself, is that more components can be fitted than the number of constrained sets, allowing it to potentially identify unknown populations. We apply these constrained mixtures to a real data set and evaluate how well it identifies populations both with and without constrained sets. For the populations with a constrained set, we compare the constrained model's performance to gateTree and an unconstrained model, whereas for those without a constrained set, we compare it to the unconstrained model only. Then, we perform an exploratory analysis of another real data set to illustrate this model's ability to construct new clusters while maintaining the populations identified by gateTree.

Chapter 7: Sequential Outlier Identification for Gaussian Mixture Models

In the final part of this thesis, we sought to focus on a single aspect of applying model-based clustering to challenging real-world data, namely the presence of outliers. As part of a collaboration with Prof. Paul McNicholas in McMaster University, we developed a novel method for identifying outliers while fitting Gaussian mixture models. Our approach involves sequentially removing potential outliers and refitting a Gaussian mixture model, while comparing the observed and theoretical distributions of sample Mahalanobis distances for each component. The optimal number of outliers is then retrospectively identified based on a graph of the aggregated Wasserstein dissimilarities between these distributions. In this chapter, we introduce our novel method and present a demonstration of its performance compared to other leading methods when applied to simulated and real

data.

Chapter 8: Sequential Outlier Identification for Linear Cluster-Weighted Models

Cluster-weighted models combine multiple regression and multivariate clustering by jointly modelling several explanatory variables and a single response variable. This chapter presents an extension of our sequential outlier identification method from Gaussian mixture models to Gaussian linear cluster-weighted models. To extend outlierMBC to this framework, we consider the theoretical distribution of the scaled squared studentised residuals from the response variable, as well as the theoretical distribution of the scaled squared sample Mahalanobis distances from the explanatory variables. This is possible due to the fact that both of these quantities follow beta distributions.

Chapter 9: Conclusion

In Chapter 9, we provide a summary of this thesis and discuss a variety of possible directions for future work and extensions of our current work. These ideas include trimodal and two-dimensional splitting mechanisms for gateTree, robust modifications for outlierMBC, computational improvements for both methods, and the possibility of combining them in the future.

2 Model-Based Clustering for Flow Cytometry

2.1 Introduction

2.1.1 Overview

In flow and mass cytometry, population identification is the process of attempting to group together events according to their cell type based on the similarity of their protein marker expression levels. The identification of cell populations in cytometry data is crucial as it facilitates downstream analysis of how specific cell types differ between subjects from different experimental groups. In particular, it allows the frequency of each cell type and other population-specific statistics to be computed so that the properties of corresponding cell populations from different samples can be compared.

The traditional and most popular approach to population identification is sequential manual gating. This process involves choosing a pair of variables, visualising the data set in a two-dimensional scatter plot of those variables, and manually drawing a boundary around a subset of cells which contains the population of interest. This subset or gate can be refined by choosing another pair of variables, using them to visualise this subset only, and further subsetting the data using another manually drawn boundary. This process is repeated for each population of interest in the data set. The protein markers which are analysed are often chosen in order to facilitate manual gating strategies which will isolate known populations of interest. The development of automated methods for population

identification has been motivated by a number of issues with sequential manual gating (Saeys et al., 2016). Manually drawing boundaries can lead to problems with bias and irreproducibility as different researchers are likely to draw different boundaries (Maecker et al., 2012; Finak et al., 2016). Two-dimensional plots fail to display the complex interactions between large sets of variables. The sequential subsetting involved in this process may prevent populations which are excluded early in the sequence from being analysed with respect to later variable pairs. Finally, for data sets with high numbers of variables, the number of variable pairs is too large for experts to examine them all. To avoid these problems, algorithms have been designed to carry out population identification using data clustering methods from statistics and machine learning.

Population identification for cytometry is a complex clustering problem which has a substantial real-world impact through its role in immunological research. Stubbington et al. (2017) identify immunophenotyping through flow cytometry as the most prominent technique for the categorisation of immune cells, while Saeys et al. (2016) describe flow cytometry as the "workhorse for high-throughput quantitative analysis of cells". Among the challenges posed by cytometry data are that some cell populations do not follow a multivariate Gaussian distribution, that clustering solutions must be consistent across multiple samples within an experiment, that the number of populations is not known *a priori*, that there are rare populations, and that the large size of the data sets can cause computational issues.

Model-based clustering is one of the most prominent statistical approaches to clustering. Its mathematical foundations and uncertainty quantification make it an appealing approach to this problem. Conversely, cytometry data is increasingly used as motivation for novel model-based methodology, particularly in relation to the development of non-Gaussian mixture models. Cytometry data motivated the skew-t mixture model introduced by Pyne et al. (2009) and was used for the primary examples in the "Beyond the Multivariate Gaussian Distribution" section of Gormley et al. (2023) and the "Non-Gaussian Model-based Clustering" chapter of Bouveyron et al. (2019). This thesis chapter examines the interdisciplinary overlap between these two research topics. In

particular, it provides an overview and discussion of the challenges involved in employing a model-based clustering approach for population identification in flow and mass cytometry. We begin by providing preliminary information on model-based clustering in Section 2.2, which we introduce in the context of population identification for flow and mass cytometry data. We discuss some of the benefits and drawbacks of model-based clustering in Section 2.2.2. Then, in Sections 2.3 to 2.7, we put forward a series of challenges associated with clustering cytometry data and discuss the adaptations which model-based clustering algorithms designed for population identification have developed or adopted in order to address these issues. Table 2.1 provides a list of model-based clustering methods which have been developed specifically for flow and mass cytometry. Some of the problems mentioned can be viewed as examples of general challenges in clustering, such as those discussed in Melnykov (2013). Challenges in clustering cytometry data which arise from using manual gating as a benchmark are discussed in Section 2.8.

Model-based clustering is only one of the clustering approaches which has been employed for population identification. The challenges of analysing flow cytometry and the potential for statistical methods to replace manual gating have been discussed for a number of years (Lock, 2003). Leading machine learning algorithms include FlowSOM (Van Gassen et al., 2015), which clusters using a self-organising map (Kohonen, 1990), and PhenoGraph (Levine et al., 2015), which performs non-parametric density-based clustering using a form of nearest-neighbours community detection. Several methods based on k-means clustering have also been developed, such as flowMeans (Aghaeepour et al., 2011), as well as tree-based methods, such as cytometree (Commenges et al., 2018). A selection of non-model-based cell population identification algorithms will be discussed in more detail in Chapter 3.

A 2013 review by the FlowCAP Consortium applied a wide range of model-based and non-model-based clustering algorithms to a selection of flow cytometry data sets (The FlowCAP Consortium et al., 2013). A further FlowCAP review was carried out by Aghaeepour et al. (2016). Weber and Robinson (2016) applied another set of methods, which included some of those featured in FlowCAP-I, to a set of mass cytometry data sets

with multiple populations of interest and also to two flow cytometry data sets with a single population of interest only. A similar review was carried out by Liu et al. (2019). These three reviews involved quantitatively comparing the performance of clustering algorithms on benchmark cytometry data sets, rather than discussing the methodology of these algorithms. More recently, Lin and Hejblum (2021) reviewed how different Bayesian parametric and non-parametric mixture modelling approaches can be applied to cytometry data. Liu et al. (2020) provided another noteworthy review of automated population identification algorithms. In this chapter, we highlight the methodological and computational adaptations that have been made across the model-based clustering literature which address the challenges of analysing flow cytometry data.

2.1.2 Mass Cytometry Data Example

The following plots display a 32-dimensional mass cytometry data set which we will refer to as Levine32 (Levine et al., 2015). This data set is publicly available via the HDCytoData R package (Weber and Soneson, 2019). After removing duplicate data points, these data consist of 118,888 distinct events. 72,463 of these events have been manually assigned to 14 gates and the remaining 46,425 events have been left unassigned. These gates vary dramatically in size, ranging from 182 events to 18,131 events. Figure 2.1 shows a pairs-style plot of three variables corresponding to surface marker expression of CD19, CD8, and CD3. The seven largest gates are individually coloured while the seven smallest gates are plotted in grey. Differences in the expression levels of these three markers between the large gates can be seen. For example, Gate 8 (Green) appears to have higher levels of CD3 than Gate 9 (Yellow) and Gate 10 (Light Blue), and higher levels of CD8 than Gate 7 (Orange). However, it does seem that it would be difficult to identify seven distinct gates within the grey points. Gates 7, 8, 9, and 10 all contain more than 10,000 events (18,131, 13,205, 12,320, and 16,111, respectively), whereas the smallest seven gates collectively contain less than 5000 events (4,676). It can also be observed that some of the boundaries between the gates are straight lines, which may seem somewhat artificial. These are a result of the manual gating approach used to

Table 2.1: This table lists model-based clustering methods which have been designed specifically for flow cytometry and/or mass cytometry. For each method, the probability distribution of its mixture model components is listed, its approach for multiple samples is mentioned, and additional details are provided.

Method	Citation	Distribution	Multi-Sample
ASPIRE	Dundar et al. (2014)	Gaussian	Joint
↪ Dirichlet Process directly estimates the number of populations.			
BayesFlow	Johnsson et al. (2016)	Gaussian; Merge	Joint
↪ Hierarchical Bayesian multi-sample model.			
CYBERTRACK	Minoura et al. (2019, 2021)	Gaussian	Pooled
↪ Designed for longitudinal data.			
CytoFA	Lee (2019)	Skew-Normal	None
↪ Factor analysis and trimming improve computational efficiency.			
FLAME	Pyne et al. (2009)	Skew-t	Meta
↪ PAM meta-clustering of separately clustered samples.			
flowClust	Lo et al. (2008)	t; Box-Cox	None
↪ Box-Cox transformation to mitigate skewness.			
flowMerge	Finak et al. (2009)	t; Box-Cox; Merge	None
↪ flowClust with post-hoc cluster merging.			
HDPGMM	Cron et al. (2013)	Gaussian; Merge	Joint
↪ Dirichlet Process directly estimates the number of populations.			
HMM-VB	Lin and Li (2017)	Gaussian; Merge	None
↪ Hidden Markov Model on Variable Blocks motivated by rare clusters.			
immunoClust	Sörensen et al. (2015)	t; Merge	Meta
↪ Divisive model fitting suits rare populations.			
JCM	Pyne et al. (2014)	Skew-t	Joint
↪ Random effects joint clustering model.			
LAMBDA	Abe et al. (2020)	Gaussian	None
↪ Model can incorporate clinical information as a covariate.			
NPflow	Hejblum et al. (2019)	Skew-t	Sequential
↪ Dirichlet Process and sequential posterior-as-prior approach.			
SWIFT	Naim et al. (2014)	Gaussian; Split; Merge	None
↪ Iterative sampling approach suits rare populations.			
Tailor	Ionita et al. (2021)	Gaussian; Merge	Pooled
↪ Data binning reduces computation time.			

obtain these gates.

Figure 2.2 is a scatter plot of a two-dimensional UMAP representation of the Levine32 data set. The same colouring scheme as that of Figure 2.1 has been used to highlight the seven largest manual gates. Non-linear dimension reduction methods such as UMAP (McInnes et al., 2020) and t-SNE (Maaten and Hinton, 2008) can be useful tools for exploring these high-dimensional data sets (Becht et al., 2019). Machine learning algorithms have also been developed which cluster the data based on a lower-dimensional representation of the data from a non-linear dimension reduction method (Shekhar et al., 2014). However, it can be challenging to tune the parameters for non-linear dimension reduction algorithms (Kobak and Berens, 2019; Wattenberg et al., 2016). The UMAP algorithm is able to distinguish between the large gates in Figure 2.1, with some overlap between similar pairs of gates. However, as mentioned previously with respect to Figure 2.1, it is not clear how the grey points would be divided into seven clusters.

2.2 Model-Based Clustering

2.2.1 Background

Clustering is a form of unsupervised learning which involves grouping together observations within a data set (Hartigan, 1975; Hennig et al., 2015). However, there is no universally agreed definition of a cluster (Hennig, 2015; McNicholas, 2016) and as a result there are a variety of different clustering approaches. One popular approach is to assume each cluster can be represented using a probability distribution and then attempt to estimate the parameters of these distributions based on the data. This is called model-based clustering and the combination of probability distributions is known as a mixture model (Celeux and Govaert, 1995; Fraley and Raftery, 2002; McLachlan and Peel, 2000). Model-based clustering is a well-established field of research dating back at least as far as Pearson (1894), with key developments in Fisher (1922) and Dempster et al. (1977). McNicholas (2016), Bouveyron et al. (2019), McLachlan et al. (2019), and Gormley et al. (2023) provide detailed reviews of model-based clustering and mixture models.

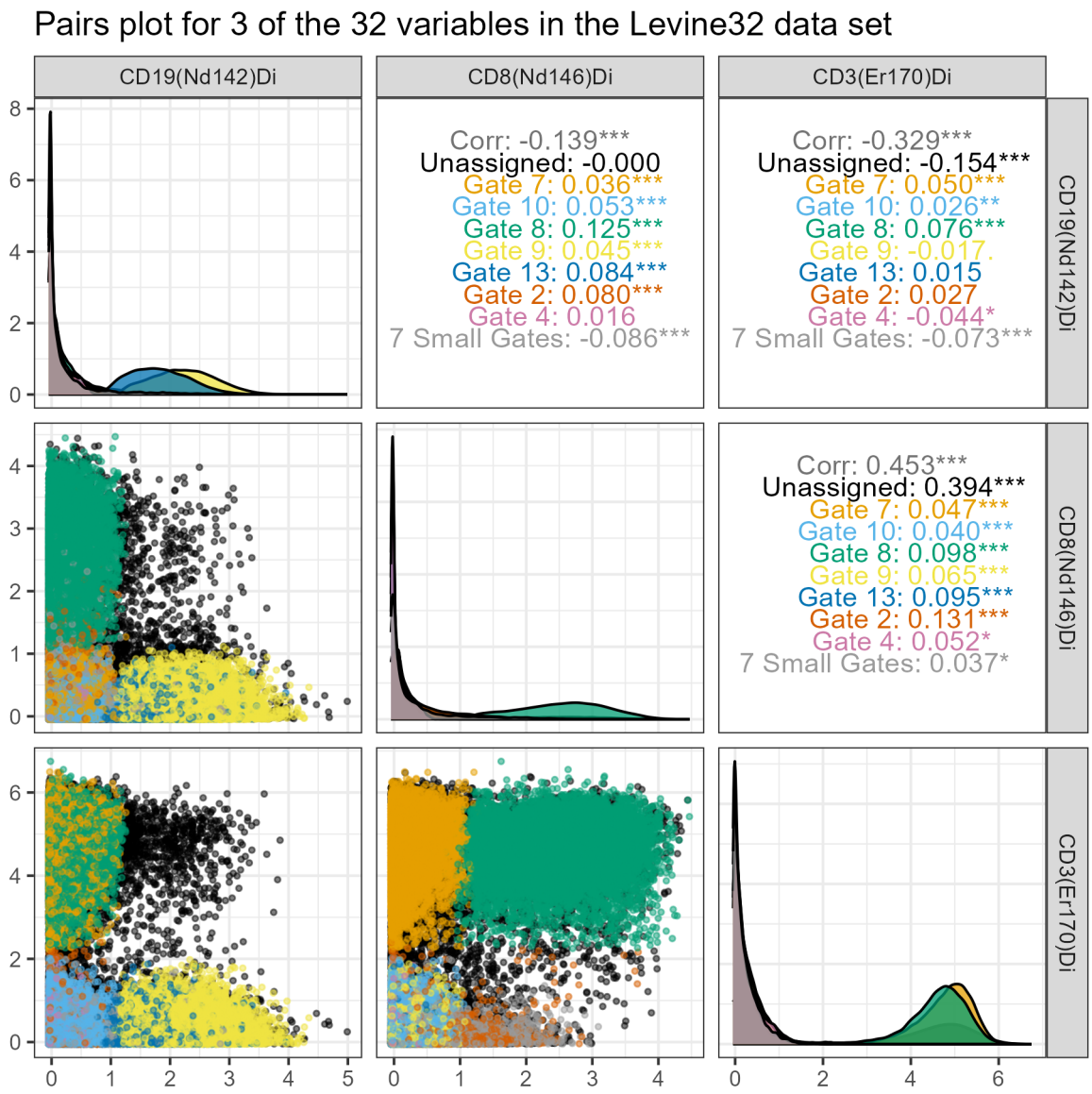


Figure 2.1: This pairs-style plot displays three of the thirty-two variables from the Levine32 data set (Weber and Soneson, 2019). These variables correspond to the surface expression of the CD19, CD8, and CD3 markers, which are important for distinguishing between the cell populations in this data set. Only events from the seven largest manually identified cell populations or gates are coloured. The unassigned events are plotted in the background as black points, while events from the seven smaller gates are shown in grey.

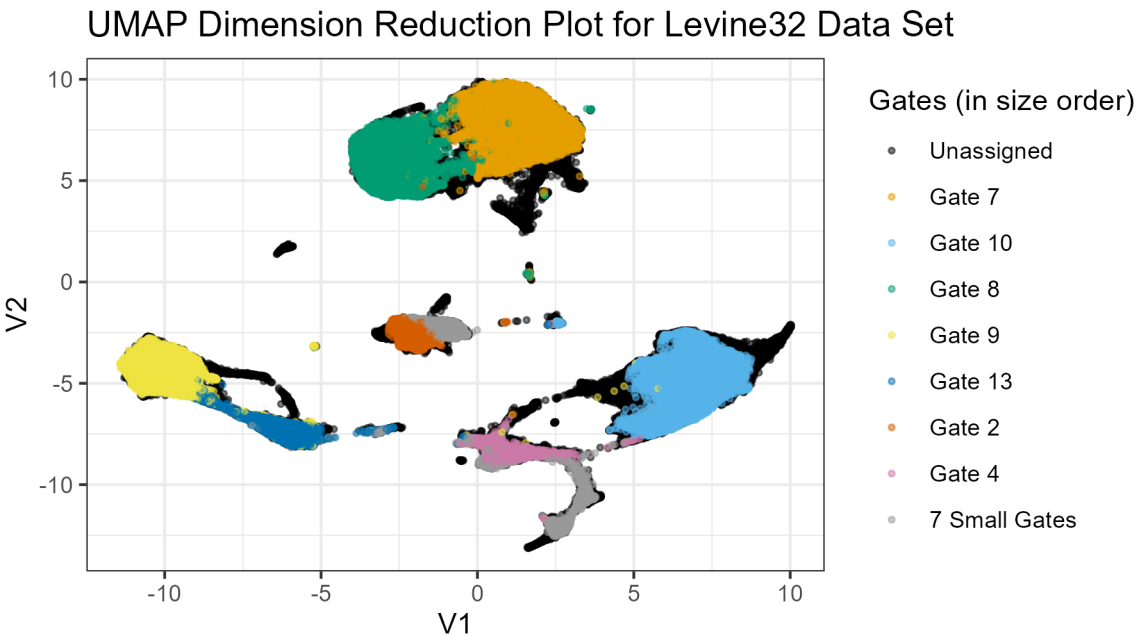


Figure 2.2: This scatter plot displays a reduced dimension representation of the Levine32 data set (Weber and Soneson, 2019). There are thirty-two variables in this data set and fourteen manually identified cell populations or gates. Events assigned to the seven largest gates are coloured. The unassigned events are plotted in the background as black points, while events from the seven smaller gates are shown in grey.

In a cytometry context, model-based clustering methods view each cell population as a sample drawn from a probability distribution. That is, they assume that the spread of cells in a population can be approximately modelled by a probability density function. The form of this function will be predetermined by the choice of probability distribution, but its parameter values will be computed based on the data. The probability density function for a finite mixture model with G components is

$$f(\boldsymbol{x}_i|\boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{x}_i|\boldsymbol{\theta}_g). \tag{2.1}$$

Ideally, each component would correspond to a separate cell population. However, in some cases, a clustering solution in which a cell population is modelled by several mixture components may provide a better model fit. This can occur if the shape of that cell population violates the assumptions of the probability distribution used in that mixture model. The mixing proportion, π_g, is the prior probability of any unspecified event belonging to component g. In other words, π_g is the proportion of events in component g.

The parameters of component g are denoted by $\boldsymbol{\theta}_g$ and its probability density function is $f_g(\cdot|\boldsymbol{\theta}_g)$. $f_g(\boldsymbol{x}_i|\boldsymbol{\theta}_g)$ is the density of component g evaluated at event i, $\boldsymbol{x}_i$.

The most common and traditional model-based clustering method is a Gaussian mixture model. This is a finite mixture model with probability density function of the form given by Equation 2.1, where each component is modelled by a multivariate Gaussian distribution. In d dimensions, the multivariate Gaussian distribution is parametrised by a $d \times 1$ mean vector, $\boldsymbol{\mu}$, and a positive-definite $d \times d$ variance-covariance matrix, $\boldsymbol{\Sigma}$. The level sets of its probability density function, that is, the curves along which the density is equal, are given by symmetrical ellipsoids, centred at $\boldsymbol{\mu}$ with shape determined by $\boldsymbol{\Sigma}$. The probability density function for component g of a Gaussian mixture model evaluated at event i, $\boldsymbol{x}_i$, is defined as

$$f_g(\boldsymbol{x}_i|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) = (2\pi)^{-\frac{d}{2}}|\boldsymbol{\Sigma}_g|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_g)^T\boldsymbol{\Sigma}_g^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_g)\right). \tag{2.2}$$

Mixture models produce a probabilistic or soft clustering. This means that they compute the probability of each event being assigned to any of the components. The formula for the estimated posterior probability of event i being assigned to component g given the data is

$$\hat{z}_{ig} = \mathbb{P}(\text{event } i \in \text{component } g|\boldsymbol{x}, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}) = \frac{\hat{\pi}_g f_g(\boldsymbol{x}_i|\hat{\boldsymbol{\theta}}_g)}{\sum_{h=1}^{G}\hat{\pi}_h f_h(\boldsymbol{x}_i|\hat{\boldsymbol{\theta}}_h)}, \tag{2.3}$$

where $\hat{\boldsymbol{\pi}}$ and $\hat{\boldsymbol{\theta}}$ represent estimates for the model parameters. To obtain a hard clustering, we assign each event to the component to which it has the highest posterior probability of being assigned. These component assignment probabilities also give us some insight into the level of uncertainty for each individual event and, therefore, how well the model fits the data. Figure 2.3 shows a scatter plot of the CD3 and CD19 protein markers for the healthyFlowData flow cytometry data set (Azad, 2013) before clustering, with the symmetrical ellipsoids of a three-component, bivariate Gaussian mixture model, and coloured according to a hard clustering from that mixture model.

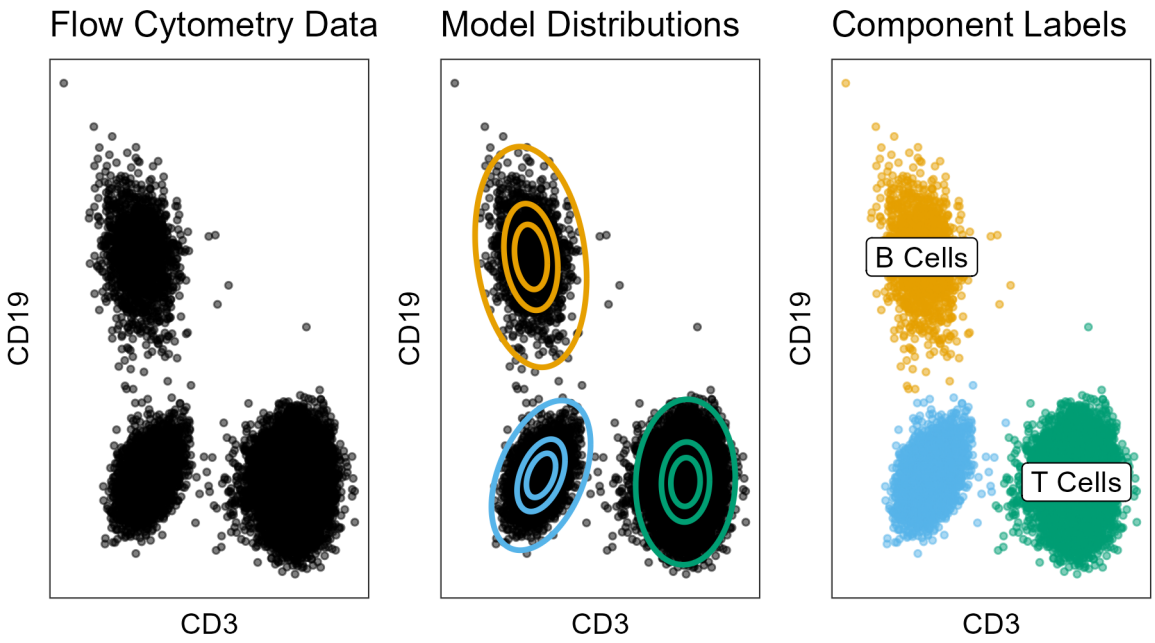


Figure 2.3: Three CD3 vs CD19 scatter plots of a flow cytometry sample from the healthyFlowData R package (Azad, 2013). The plots show how the data is clustered by a three-component, bivariate Gaussian mixture model, implemented by the mclust R package (Fraley et al., 2022). They display the two stages of fitting a mixture model: estimating each cluster's distribution and assigning each event to the cluster with the most suitable distribution. In the left scatter plot, the data is uncoloured but three distinct clusters are visible. In the middle scatter plot, three coloured, concentric ellipses are imposed over each cluster, showing the 33^{rd}, 66^{th}, and 99^{th} percentile level sets of each cluster's density according to a Gaussian distribution. In the right scatter plot, the three clusters are coloured and the CD3+CD19− and CD3−CD19+ clusters are labelled as "T Cells" and "B Cells", respectively.

2.2.2 Advantages of Model-Based Clustering

One of the main benefits of employing model-based clustering is its strong statistical framework. As well as the richer, event-specific clustering solution given by probabilistic clustering compared to hard clustering algorithms, employing a model-based clustering approach also allows us to be clear about our algorithm's assumptions and provides mathematical justification for our decisions. When choosing a clustering algorithm and interpreting any clustering solution it produces, it is important to understand the type of clusters that it tends to construct. Model-based clustering explicitly assumes the probability distribution of each component, which provides transparency around the form of its clusters. Its mathematical foundations can also provide reassurance about the validity of model selection decisions with respect to number of components or component distributions. The lack of a true class structure in cytometry makes model-based clustering's clear assumptions and statistical framework particularly appealing.

Another benefit of model-based clustering is its adaptability, as discussed in Sections 2.3 to 2.7. Changing the probability distribution of a mixture model influences the type of clusters which the model will construct. Extending a sample-specific model to form a hierarchical multi-sample model allows information to propagate between samples. Factor analysis includes dimension reduction as a part of the clustering process, allowing each cluster to have its own projection to a lower-dimensional space. The model fitting process can also be modified to search for rare populations. These adaptations, which will be discussed in more detail in the following sections, allow mixture models to be adjusted to suit the complexities of flow and mass cytometry data, without losing the benefits of their mathematical origins.

One drawback of model-based clustering is its computational cost. Complex models require the estimation of a high number of parameters, which can be relatively slow, depending on the available computing power. As a result, fitting a mixture model and obtaining a soft, probabilistic clustering will often be more computationally demanding than obtaining a hard clustering from a more straightforward clustering algorithm, such as

k-means. However, mixtures of factor analysers and other parsimonious mixture models can be used to reduce the number of parameters to be estimated.

Another disadvantage is that the form of the clusters identified by a mixture model can be limited by the distributions chosen for its components. That is, if the shapes of the cell populations violate the assumptions underpinning the mixture model, then it may not be flexible enough to cluster the data satisfactorily. This can be partially mitigated by choosing a more flexible and robust distribution for the components, as discussed in the next section.

2.3 Non-Gaussian Populations: Alternative Models

Model-based clustering operates within a strong statistical and probabilistic framework compared to other clustering approaches. This mathematical foundation is obtained by making assumptions about the distribution of the data within each cluster. For example, the Gaussian mixture model defined by Equations 2.1 and 2.2 assumes that each cluster occupies a symmetric ellipsoidal region and that events are dispersed through this region according to a multivariate Gaussian distribution. Mixture models can still be effective when their assumptions are not fully satisfied, however there can be undesirable consequences if the assumptions are severely violated. An example of this is the tendency of Gaussian mixture models to model a single non-Gaussian cluster using multiple components (Baudry et al., 2010). One of the challenges associated with using model-based clustering for population identification is that the cell populations may not be approximately Gaussian-distributed, either in terms of their asymmetrical shape or the concentration of events away from the centre of the population. However, by choosing a more sophisticated distribution for our mixture components than the Gaussian distribution, it is possible to address these concerns.

The level sets of a t distribution's probability density are symmetric ellipsoids, similar to those of a Gaussian distribution (Andrews and McNicholas, 2012; McLachlan and Peel, 1998). However, the probability density of a t distribution decreases less rapidly away

from the mean than that of a Gaussian distribution. Hence, sampling from a t distribution is more likely to result in points far from the mean than sampling from a Gaussian distribution. As a result, such points can be accommodated more comfortably by t distributions. Skewed distributions are generalisations of symmetric distributions which allow for asymmetry. Whereas Gaussian and t mixture models assume that the shape of each population can be approximated by a symmetric ellipsoid, their skewed generalisations, skew-normal and skew-t mixture models, include a skewness parameter to explicitly account for asymmetry in the shape of a cluster. Lee and McLachlan (2022) provide a detailed review of mixtures of skewed distributions. Many non-Gaussian mixture models were not necessarily intended specifically for cytometry data but instead were developed as a generic clustering method (Franczak et al., 2014; Lee and McLachlan, 2016; O'Hagan et al., 2016; Zhu and Melnykov, 2018). Among others, Bouveyron et al. (2019) credit McLachlan and Peel (1998) for the t mixture, Lin (2009) for the skew-normal mixture, and Pyne et al. (2009) for the skew-t mixture. However, the development of the skew-normal distribution itself is attributed to Azzalini and Dalla Valle (1996). Interestingly, Pyne et al. (2009) does explicitly intend for the skew-t mixture to be used for flow cytometry data. The software implementation corresponding to this publication is named FLAME (FLow analysis with Automated Multivariate Estimation). Furthermore, the 'Non-Gaussian Model-based Clustering' chapter of Bouveyron et al. (2019) uses flow cytometry data as an example.

Several methods for flow and mass cytometry have employed these more flexible distributions in order to account for the non-Gaussian populations commonly found in this data. flowClust (Lo et al., 2008) addresses asymmetry in the data by applying a Box-Cox transformation (Box and Cox, 1964) intended to make the populations more symmetric (Lo and Gottardo, 2012), then uses a standard t mixture model as a more outlier-friendly alternative to a Gaussian mixture model. FLAME (Pyne et al., 2009), JCM (Pyne et al., 2014), and NPflow (Hejblum et al., 2019) take a more direct approach by employing the skew-t distribution. However, the symmetric Gaussian remains the most commonly used distribution. It is the basis of HDPGMM (Cron et al., 2013), ASPIRE (Dundar et al.,

2014), SWIFT (Naim et al., 2014), BayesFlow (Johnsson et al., 2016), CYBERTRACK (Minoura et al., 2019, 2021), LAMBDA (Abe et al., 2020), and Tailor (Ionita et al., 2021). Meanwhile, immunoClust (Sörensen et al., 2015) and CytoFA (Lee, 2019) utilise a t distribution and a skew-normal distribution, respectively, both choosing an intermediate option between the symmetric Gaussian and the skew-t.

2.4 Multiple Samples: Hierarchical Models

One of the most interesting challenges associated with clustering data from flow or mass cytometry is the need to cluster multiple samples in a coherent and consistent manner. Flow and mass cytometry are often employed in immunological studies involving multiple sets of samples from different control or treatment groups. Population identification methods must identify corresponding clusters of events in each sample, despite the biological differences between the subjects and any technical variation between the samples. In particular, these biological differences are often the focus of the experiment and must not be eliminated. Figure 2.4 displays scatter plots of the CD4 and CD8 protein markers for four flow cytometry samples from the healthyFlowData package (Azad, 2013), where each sample was obtained from a different subject. The four samples are clustered according to four-component Gaussian mixture models implemented by the mclust package (Fraley et al., 2022). This figure demonstrates how the shape, size, and location of corresponding cell populations can differ between different cytometry samples.

The most straightforward approach to a multi-sample clustering problem would be to combine the samples into one large, pooled sample and apply the clustering algorithm once. This is the approach taken by Tailor (Ionita et al., 2021) when dealing with multiple samples. It relies on the assumption that the shape and location of populations do not vary substantially between samples. If this assumption is violated, then distinct populations from different samples can overlap in the pooled sample despite not overlapping in any individual sample. However, pooling does allow for some sharing of information between samples. A rare population could contain too few events to be

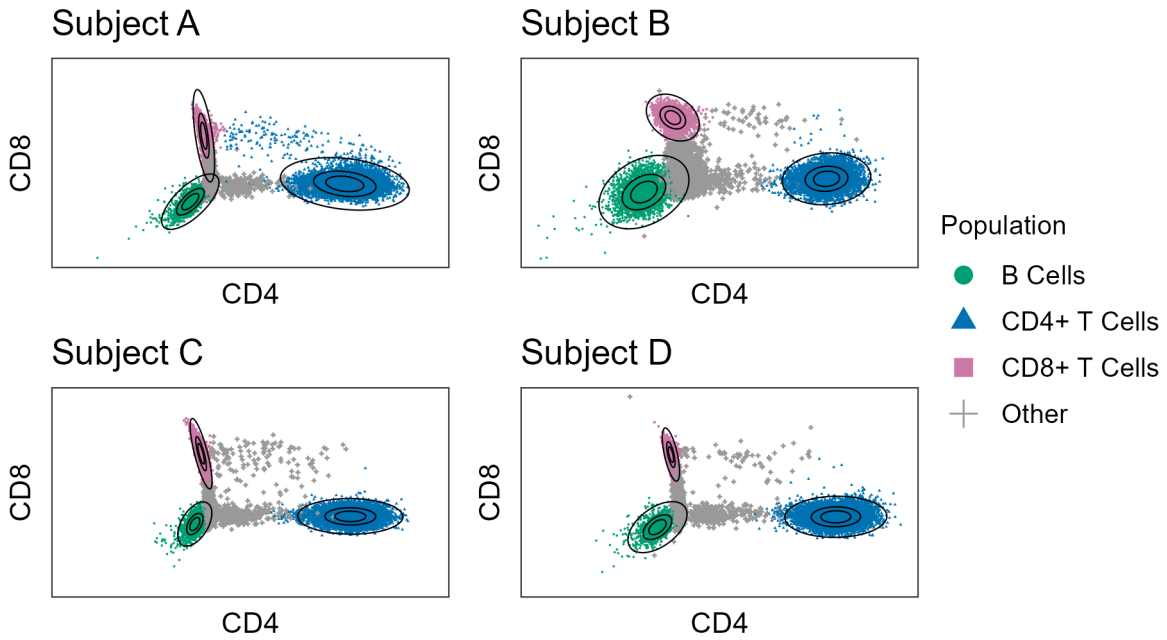


Figure 2.4: Four CD4 vs CD8 scatter plots of healthyFlowData (Azad, 2013) samples from four individuals. The first sample from each subject is shown. These plots illustrate that corresponding cell populations from different subjects in the same study can have different shapes, sizes, and locations despite being analysed in the same way. Each plot is coloured according to a four-component Gaussian mixture model implemented by the mclust R package (Fraley et al., 2022). Three concentric ellipses are imposed over three of the four clusters. These are the 33^{rd}, 66^{th}, and 99^{th} percentile level sets of the Gaussian density for the clusters identified as "B Cells", "CD4+ T Cells", and "CD8+ T Cells".

identified in some samples but become identifiable in the pooled sample when its events from all samples are considered.

Another possibility is to cluster each sample individually and apply a meta-clustering algorithm to match groups of similar clusters from different samples. FLAME (Pyne et al., 2009) and immunoClust (Sörensen et al., 2015) both include meta-clustering extensions to their original clustering algorithms. Several stand-alone meta-clustering methods have also been released which are intended to be used after any clustering algorithm has been applied to each sample individually, for example, flowMatch (Azad et al., 2012), FlowMap-FR (Hsiao et al., 2016), QFMatch (Orlova et al., 2018), and MetaCyto (Hu et al., 2018). One popular approach to meta-clustering, used by FLAME and immunoClust, is to treat the cluster centres from all samples as a new set of data points and apply a clustering algorithm to them. Another approach is to construct a bipartite graph from two sets of clusters and match the clusters based on pairwise dissimilarities, similarly to flowMatch. Figure 2.5 demonstrates how flowMatch can identify sets of corresponding clusters from different samples and construct template Gaussian components for each of these meta-clusters. It can be difficult to correctly match corresponding populations in different samples. Meta-clustering does not benefit from any sharing of information between samples when constructing the clusters.

Model-based clustering also facilitates a joint clustering approach. By defining a hierarchical model, it is possible to allow information to propagate between samples without pooling them. These models jointly cluster all samples simultaneously. They construct a model for each sample, similarly to model-based clustering with meta-clustering, but consider these sample-specific models to be variations of a cross-sample template model. As a result, information from one sample, for example, evidence of the existence of a rare population, can affect the clustering solution constructed for the other samples by influencing the form of the template model. JCM (Pyne et al., 2014) and ASPIRE (Dundar et al., 2014) implement joint clustering models based on random effects, whereas BayesFlow (Johnsson et al., 2016) jointly clusters using a hierarchical Bayesian model.

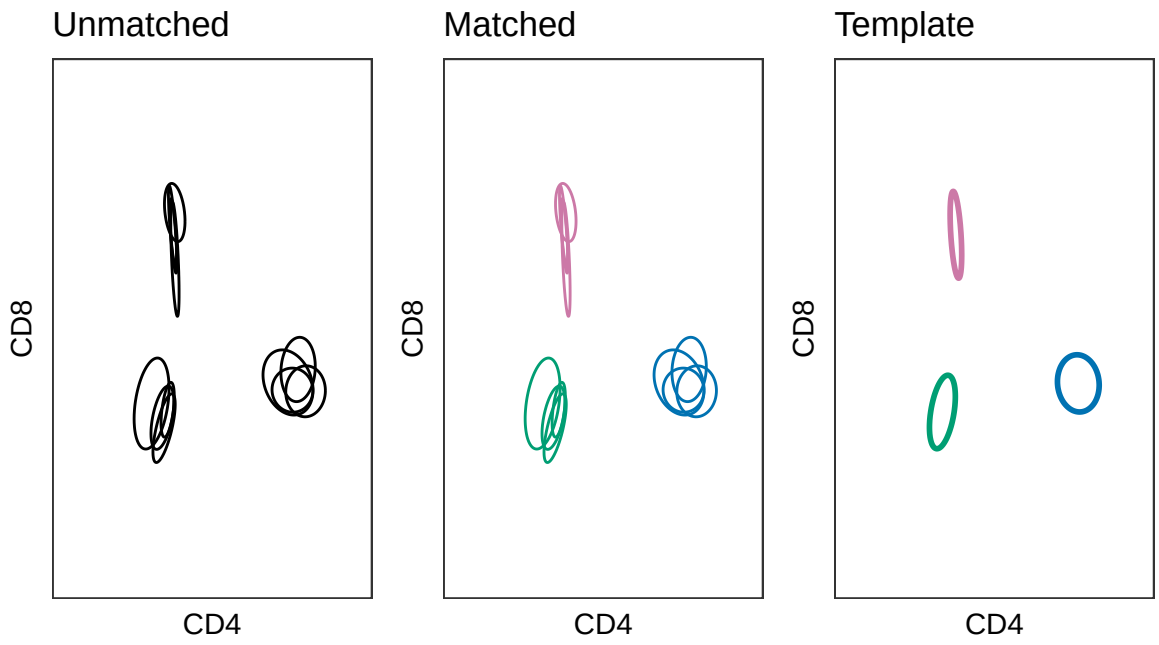


Figure 2.5: Three plots illustrating how flowMatch (Azad et al., 2012) matches clusters from different samples and constructs a template Gaussian mixture model with one component per meta-cluster, or set of clusters. A four-component Gaussian mixture model was fitted to the first sample of each of the four subjects from the healthyFlowData R package (Azad, 2013) using the mclust R package (Fraley et al., 2022). Three components from each sample's mixture model were provided to flowMatch. Left: 66^{th} percentile ellipses for twelve components from four samples are overlaid on the same plot. Middle: The same twelve ellipses coloured as three sets of four clusters according to a meta-cluster assignment by flowMatch. Right: 66^{th} percentile ellipses for the three template components constructed by flowMatch for each meta-cluster.

Another model-based approach to the multi-sample problem is sequential clustering. If a Bayesian model is applied to a single sample, then a posterior distribution for the model parameters will be obtained which can be used as a prior distribution for the next sample's model. NPflow (Hejblum et al., 2019) is an example of a model which uses this sequential Bayesian approach. This technique allows information to propagate between samples without requiring all samples to be clustered simultaneously, making it particularly useful for longitudinal samples from the same subject.

2.5 Unknown Populations: Model Selection

Due to the complexity of the population identification problem, it may be difficult to choose an appropriate number of clusters before applying a clustering algorithm. Therefore, having a data-driven procedure to inform this choice is desirable. However, many clustering algorithms do not have natural methods to select the number of clusters. In contrast, model-based clustering methods are able to take advantage of mathematically derived likelihood-based model selection criteria such as the Bayesian Information Criterion (BIC; Schwarz, 1978) or the Integrated Completed Likelihood (ICL; Biernacki et al., 2000) to select the number of components to be used in their mixture model (Fraley and Raftery, 2002). The likelihood for the model parameters increases as the number of components increases, so these criteria include model complexity penalty terms. When comparing two models with different numbers of components, the criterion will only select the model with a higher number of components if the increase in the likelihood is larger than the increase in the complexity penalty term.

Baudry et al. (2010) define the BIC to be

$$\mathsf{BIC}(G) = \sum_{i=1}^{n} \log \left(\sum_{g=1}^{G} \hat{\pi}_g f_g(\boldsymbol{x}_i | \hat{\boldsymbol{\theta}}_g) \right) - \frac{\nu_G}{2} \log(n), \tag{2.4}$$

and the ICL to be

$$\text{ICL}(G) = \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig} \log\left(\hat{\pi}_g f_g(\boldsymbol{x}_i|\hat{\boldsymbol{\theta}}_g)\right) - \frac{\nu_G}{2}\log(n), \tag{2.5}$$

where $\hat{\boldsymbol{\pi}}_g$ and $\hat{\boldsymbol{\theta}}_g$ are the estimated mixing proportion and parameters for component g, $\hat{z}_{ig}$ is the estimated probability of event i being assigned to component g, and ν_G is the number of estimated parameters required by the model with G components. They also state that the ICL is approximately equal to the BIC with an additional mean entropy penalty term. This means that the ICL more severely penalises clustering solutions with higher component assignment uncertainties. In practice, this means that the ICL is less likely to choose a model with overlapping or adjacent components than the BIC, since events near the boundary between these components could reasonably be assigned to either. As a result, the ICL tends to select a lower number of components than the BIC.

Another model selection approach is to use the Bayes factor (Kass and Raftery, 1995), although we are not aware of this being used by any cytometry-specific methods. flowClust (Lo et al., 2008), JCM (Pyne et al., 2014), and LAMBDA (Abe et al., 2020) utilise the BIC, whereas immunoClust (Sörensen et al., 2015) employs a modified version of the ICL which is intended to serve as a compromise between the two criteria. The main drawback of using these model selection criteria is that models with different numbers of components can only be compared after being fitted to the data, and the process of fitting a range of models and only using one can be considered time-consuming and wasteful.

Bayesian non-parametric mixture models offer an alternative approach (Ghahramani, 2013). They incorporate selecting the number of components into the model fitting process so that it is carried out as the parameters of the model are being estimated. This allows them to fit a single model to the data without pre-specifying the number of components to be used. However, these methods often utilise Markov Chain Monte Carlo (MCMC) sampling schemes, which can be computationally demanding. Summarising the

output of Bayesian non-parametric models can also require additional consideration, as they sample from the posterior distribution over the space of partitions instead of converging to a single clustering solution. Label-switching issues can be resolved using loss-based relabelling approaches (Jasra et al., 2005; Stephens, 2000), while more recently Wade and Ghahramani (2018) outlined a general approach for identifying an optimal partition of the data using MCMC output. Lin and Hejblum (2021) provide a discussion of Bayesian non-parametric mixture modelling. Several methods for flow and mass cytometry have taken this approach using Dirichlet process priors (Ferguson, 1973; Neal, 2000), for example, HDPGMM (Cron et al., 2013), ASPIRE (Dundar et al., 2014), and NPflow (Hejblum et al., 2019).

Another approach for determining the number of cell populations is to fit a relatively high number of mixture components to the data and apply a merging procedure to combine multiple components for each cell population. It is well-established in model-based clustering that if the shape of a cluster is not compatible with the mixture components' probability distribution, then the model fitting process may compensate by modelling a single cluster with more than one component (Fraley and Raftery, 2002; Pyne et al., 2009). A variety of approaches have been proposed for merging mixture components (Hennig, 2010). flowMerge (Finak et al., 2009) applies the entropy-based merging approach of Baudry et al. (2010) to mixture components fitted by flowClust (Lo et al., 2008). Another approach is to associate each component with a local density maximum or mode and then merge any components that share the same mode. The Hidden Markov Model on Variable Blocks (HMM-VB) method (Lin and Li, 2017) employs a modal clustering approach as part of the model fitting process. BayesFlow (Johnsson et al., 2016) and SWIFT (Naim et al., 2014) both merge unimodal components, while SWIFT also splits multimodal components. immunoClust (Sörensen et al., 2015) and HDPGMM (Cron et al., 2013) carry out merging in the context of dealing with multiple samples. Tailor (Ionita et al., 2021) annotates each component based on their expression levels for every marker and merges components with the same annotation. Clustering methods that involve a merging step often "over-cluster" initially, that is, they construct more clusters

than is necessary for the data. Over-clustering with a mixture model and then merging some of the resulting mixture components is appealingly flexible since a union of components can form a wider variety of cluster shapes than a single component. However, proceeding in a fully principled manner when merging components is challenging, because each merging approach must be motivated by subjective choices about what constitutes a cluster (Hennig, 2010). Another drawback of this approach is that the flexibility obtained by representing a cluster as a union of components comes at the expense of interpretability as the cluster will not be described by the model parameters of a single component and its construction process will be more convoluted.

Despite these model-based approaches, selecting the correct number of clusters remains a difficult challenge in analysing flow or mass cytometry data. There is no objectively correct number of populations in a cytometry sample since hierarchical relationships exist between many cell populations. As a result, distinct solutions with different levels of cluster resolution may be suitable for particular problems. In other words, experts may require a large number of small clusters or a small number of large clusters, depending on the context of their research. Although the intermediate solutions through which an iterative merging procedure transitions could be interpreted as representing a hierarchy of clustering solutions, it is also common for these methods to only present the final outcome. Neither likelihood-based criteria nor non-parametric models address the dual issues of cluster resolution and cell hierarchy as they both result in a single number of components being chosen. While mathematical justification exists for this choice, it may not suit the specific needs of the researcher.

2.6 Rare Populations: Novel Model Fitting

Cytometry data sets can contain both rare and abundant populations. Some model-based clustering methods for cytometry data have developed alternative model fitting approaches designed to better identify these rare populations.

SWIFT (Naim et al., 2014) uses a novel procedure called Weighted Iterative Sampling

which is designed to prevent large populations from dominating the model fitting process. SWIFT starts by fitting a mixture model to a uniform subsample from the data. The p most populous components are included in all subsequent models with their original parameter values fixed. Another subsample is computed from the data with events sampled according to their probability of not being assigned to any of the previously fixed components. Then, a new model is fitted to this subsample, which includes the previously fixed components, and the p most populous components from this model are also fixed. This process of constructing a weighted subsample which avoids events from previous components and fitting a new model to obtain the next p components is repeated until all components' parameter values have been fixed.

immunoClust (Sörensen et al., 2015) adopts a hierarchical divisive clustering approach. It fits models with a range of component numbers to a weighted subsample from the full data set, including a single component model, and uses a modified version of the ICL criterion defined by Equation 2.5 to select one of these models. This model is then used to partition the dataset and a new range of models is fitted to each of the resulting subsets. This process is repeated until the modified ICL criterion selects the single component model for each remaining subset.

HMM-VB (Lin and Li, 2017) is a model-based clustering approach that is designed to identify rare clusters in large data sets, and is partly motivated by cytometry data. This model involves arranging the variables into a sequence of disjoint sets and fitting a Gaussian mixture model to each variable block with a sequential dependence between these blocks controlled by a Hidden Markov Model. The model fitting procedure is carried out using a modal version of the Baum–Welch algorithm, which is related to the Expectation-Maximisation (EM) algorithm. This modification involves merging mixture components with the same mode.

Initialisation is an important aspect of model fitting, particularly for mixture models that are fitted to data via an EM algorithm (Baudry and Celeux, 2015). Model-based hierarchical clustering can be used to obtain an initial partition of the data (Fraley, 1998;

Fraley and Raftery, 2002; Scrucca and Raftery, 2015). This is the initialisation technique used by immunoClust, flowClust, and flowMerge. Another approach is to test multiple initialisations and retain the one that achieved the highest likelihood after a small number of EM iterations (O'Hagan et al., 2012). FLAME, HMM-VB, and SWIFT all utilise multiple initial partitions. FLAME partitions the data by applying the k-means clustering algorithm, while SWIFT generates random partitions, and HMM-VB uses a combination of both types of partition. Tailor's binning approach, which is discussed in the next section, is in part an initialisation technique as the representatives of the largest bins are used as initial component means. An intriguing and potentially promising idea for cytometry data sets would be to obtain an initial partition by clustering a reduced dimension representation of the data obtained from a non-linear dimension reduction technique, such as t-SNE and UMAP. However, we have not come across any methods which utilise this approach. Researchers may have been put off by the possibility of irregularly shaped initial clusters as a result of the non-linear nature of the dimension reduction, or it could simply be an under-explored area of research.

2.7 Large Data Sets: Binning and Factor Analysis

The size of flow and mass cytometry data sets can cause computational challenges both in relation to dimensionality and sample size, that is, the number of variables and the number of events. CytoFA (Lee, 2019) applies a factor analysis approach as a dimension reduction technique while Tailor (Ionita et al., 2021) employs binning to construct a smaller representative data set used for model fitting.

CytoFA adopts a factor analysis approach for a skew–normal mixture model. Factor analysis is a popular model-based method for clustering high-dimensional data which incorporates dimension reduction into the model fitting process (Ghahramani et al., 1996; McNicholas and Murphy, 2008; Murphy et al., 2020). It reduces the number of parameter values that need to be estimated to reduce the computational complexity of the model. This approach assumes that the variation present in the given data can be modelled by a

simpler covariance structure than that of a traditional, high-dimensional model. Instead of modelling each cluster as a set of data points sampled from a high-dimensional random variable, factor analysis models each cluster as a high-dimensional projection of a set of data points sampled from a low-dimensional random variable.

Whereas CytoFA uses a dimension reduction approach to address the high number of variables, Tailor instead tries to reduce the sample size by binning the events and replacing each bin with a representative data point. Each variable is partitioned univariately using one-, two-, or three-component Gaussian mixture models to create an orthogonal partitioning of the data space. By default, the optimal number of components with respect to BIC is selected for each variable, however, an alternative number can be chosen by the user. Highly populous regions are further divided into bins using k-means, sparsely populated regions are discarded, and moderately populated regions are used as bins directly. A single weighted representative data point is obtained from each of the bins. A weighted EM algorithm is used to fit a Gaussian mixture model to this representative data set. Other methods, including SWIFT (Naim et al., 2014) and immunoClust (Sörensen et al., 2015), attempt to reduce the number of events by subsampling, where a subset of the original events are selected, rather than constructing new data points as representatives.

2.8 Other Challenges Related to Manual Gating

To assess how well a clustering algorithm performs when applied to a flow cytometry or mass cytometry data set, its clustering solution will often be compared to a manual gating of the same data set. The F_1 measure (Rijsbergen, 1979) is commonly used to quantify the similarity between the set of clusters produced by the algorithm and a set of gates produced by an expert. Evaluating the performance of population identification algorithms by comparing their clustering solutions to manual gating using external validation indices, such as the F_1 measure, assumes that the manual gating represents the true class structure of the data. However, if a researcher is focused on a particular cell type, they

may not be interested in distinguishing between every subpopulation of another cell type. Hence, the manual gating may only represent one of the valid class structures for a particular data set. If an algorithm discovers a structure in the data that was not identified in the manual gating, then this will reduce its F_1 score because its clusters will be less similar to the gates.

It may be beneficial to allow the expert to provide information to the clustering algorithm about which cell populations they are trying to identify. This could increase the practicality of an algorithm in a research context and improve the performance of the algorithm with respect to external validation indices in demonstrations and applied comparisons. The ACDC algorithm developed by Lee et al. (2017) utilises expert information in the form of a table with one row per cell type and one column per protein marker, whose entries indicate whether the cell type is positive, negative, or neutral for that marker. Ji et al. (2018) inform their Bayesian tree algorithm using the same table, and the GateMeClass method (Caligola et al., 2024) also employs a similar table.

Another challenge relating to replicating manual gating is the practice of leaving events unassigned. As a result of the sequential subsetting nature of manual gating, many events are not assigned to any of the final gates. This allows the manual gating process to naturally perform outlier detection, as well as excluding borderline events that could belong to more than one population. Traditional mixture models are designed to assign every event to one of their mixture components. However, methods have been developed for outlier identification in model-based clustering. Mixture models can be modified to include a single uniform component across the region occupied by the data to which outliers should be assigned (Banfield and Raftery, 1993). Another approach, called trimming, involves identifying and excluding outliers during the model fitting procedure (Gallegos and Ritter, 2005; García-Escudero et al., 2008). In this case, events are usually classified as outliers if they do not fit the model well. In other words, if the value of the mixture model's density function at that event is low. Chan et al. (2008) discuss labelling as unassigned any events that lie outside a pre-specified confidence region for their assigned component or whose posterior probability falls below a pre-specified threshold.

flowClust (Lo et al., 2008) classifies outliers based on the Mahalanobis distance from an event to the mean of each component. CytoFA (Lee, 2019) trims the events with the lowest contribution to the mixture model. We are not aware of any other model-based clustering algorithms that attempt to address the prevalence of unassigned events in cytometry data.

2.9 Conclusion

Model-based clustering is a statistical approach to data clustering that can be used for population identification in flow and mass cytometry. This approach's appeal lies in its clear assumptions and mathematical foundations. Population identification is a particularly challenging problem. Among the difficulties associated with this task are the non-Gaussianity of the populations, the multi-sample nature of cytometry studies, the absence of a known number of populations, the imbalance between the sizes of abundant and rare populations, and the high dimensionality and large sample sizes of the data sets. We have discussed a number of ways in which mixture models have been modified to address these issues, which are displayed in Figure 2.6. These include the adoption of more robust and flexible distributions, the design of hierarchical, multi-sample models, the application of model selection criteria and non-parametric mixtures, the development of novel model fitting procedures, and the use of data reduction techniques, such as binning and factor analysis. The range and potential of these adaptations demonstrate the capability of model-based clustering to evolve to overcome the challenges posed by population identification.

This chapter has not discussed some of the inherent issues with cytometry data that are not specific to clustering (Herzenberg et al., 2006; Liechti et al., 2021). Autofluorescence refers to the innate fluorescence of cells that is not due to the excitation of a fluorescent antibody attached to their protein markers. As a result of autofluorescence, we cannot directly attribute all of an event's measured fluorescent intensity to its expression of a protein marker. Fluorescence spillover is when the emission spectra of fluorochromes

Figure 2.6: A graphical summary of the challenges and adaptations of using model-based clustering for flow cytometry. Five challenges (green) are centred around "Model-Based Clustering for Cytometry" (pink). Two adaptations (purple) arise from each of these challenges.

overlap and contribute to the fluorescence intensity measured by each others' detectors. This must be corrected via a process called compensation which linearly transforms the detected fluorescent intensities with respect to the inverse of the spillover coefficient matrix (Roederer, 2002). These spillover coefficients are computed based on Fluorescence Minus One (FMO) control samples. Another issue related to spectral overlap, spillover spreading error, arises from increased errors in photon counting and can reduce the sensitivity of the analysis to low intensity signals (Liechti et al., 2021; Nguyen et al., 2013). It is standard practice to apply a parametrised, non-linear transformation to the data, such as the biexponential, logicle (Parks et al., 2006), and inverse hyperbolic sine (arcsinh) transformations. Choosing the correct parameter setting for these transformations is critical to properly visualising the data and accurately identifying the cell populations (Finak et al., 2010; Liechti et al., 2021). Inter-sample variability in cytometry can arise as a result of technical and biological variation. Collecting samples on different days can cause a higher degree of technical variability between the batches of samples collected under unintentionally different conditions. Normalisation approaches have been developed to try to correct these batch effects (Van Gassen et al., 2020), however, minimising inter-sample variability is preferable to trying to remove it. Finally, automated methods for cell population identification are usually applied to data that has been manually pre-gated. For example, the dead cells, debris, and doublets may be removed using manual gating on the scatter channels and a viability or live/dead marker. The manual pre-gating may also isolate the broad cell type of interest, such as lymphocytes, before the automated method identifies cell populations within that cell type. Therefore, any issues with this manual pre-gating can affect the performance of the automated method. However, developing an algorithm to conduct this pre-gating would be challenging due to the different types of variables involved and the heavy reliance of this process on biological knowledge. The above issues with analysing cytometry data contribute to the difficulty of automating cell population identification.

In Section 2.8, we highlighted that the practice of treating manual gating as the ground truth for external validation could penalise algorithms for identifying a different but

potentially valid class structure in the data. We believe that the development of population identification algorithms that allow the expert to inform the model is an area that has been under-explored by the model-based clustering community. As well as improving the quantitative performance of the algorithm in comparisons, this also has the potential for real-world benefits by allowing immunological researchers to tailor the algorithm to their specific needs. We will revisit this topic in Chapter 4, where we review some methods for user-informed cell population identification, and in Chapters 5 and 6, where we introduce a pair of novel user-informed methods.

This chapter has discussed the challenges of population identification for flow and mass cytometry data and highlighted a number of adaptations that have been developed to aid model-based clustering in addressing these challenges. We believe that the statistical foundations and adaptability of model-based clustering make it a suitable choice for the problem of population identification.

3 Non-Model-Based Methods for Flow Cytometry

3.1 Introduction

In Chapter 2, we discussed a variety of model-based methods which have been adapted to suit cytometry data. However, many of the leading cell population identification methods are not model-based (The FlowCAP Consortium et al., 2013; Weber and Robinson, 2016; Liu et al., 2019). We have selected seven cell population identification algorithms to discuss in this section.

Table 3.1: The non-model-based methods discussed in Chapter 3.

Method	Reference
FLOCK	Qian et al. (2010)
flowMeans	Aghaeepour et al. (2011)
SPADE	Qiu et al. (2011); Qiu (2017)
FlowSOM	Van Gassen et al. (2015)
PhenoGraph	Levine et al. (2015)
X-Shift	Samusik et al. (2016)
cytometree	Commenges et al. (2018)

These unsupervised methods implement a variety of approaches, often combining centroid-based, non-parametric density-based, and hierarchical clustering techniques to overcome the challenges associated with cytometry data.

The k-means algorithm iteratively alternates between assigning data points to their nearest cluster centroid and re-computing the centroids based on the updated cluster

assignment. We will refer to this, and the related self-organising map algorithm described in Section 3.2.4, as centroid-based algorithms because they assign data points to clusters based solely on which cluster's centroid is closest. Non-parametric density-based methods estimate the local density of each data point and construct clusters based on high-density and low-density regions. They differ from model-based clustering methods because they do not construct a mixture model and they rely on a more local notion of density. Hierarchical agglomerative and divisive methods initially consider the data set to consist of many clusters each containing a single data point or a single cluster containing all of the data points, respectively. Agglomerative methods then proceed to iteratively merge clusters, while divisive methods iteratively split clusters. They may continue until the clusters cannot be merged or split any further, that is, until they have reached the other's initial solution, or they may have a premature stopping criterion. Both hierarchical approaches construct a range of intermediate cluster solutions between their starting point and their stopping point. Each of these three broad families of non-model-based clustering methods is intuitive in its own way, and each has different advantages and disadvantages. In Section 3.2, we will discuss how these approaches have been adapted and combined for cytometry data.

3.2 Methods

3.2.1 FLOCK

Qian et al. (2010) introduced a non-parametric density-based clustering procedure called FLOCK (FLOw Clustering without K). FLOCK begins by partitioning each dimension of the data into t equally sized bins. For a d-dimensional data set, this results in a hypergrid consisting of $t \times d$ hyper-rectangles. Two data-driven methods can be used to select the number of bins, t. A threshold must also be chosen, which is used to identify which bins are "dense". A hyper-rectangle is classified as dense if the number of events that it contains is greater than this threshold. Three data-driven methods are implemented for determining possible values of this threshold and FLOCK chooses the values that results

in the clustering solution with the best silhouette value (Rousseeuw, 1987). Adjacent dense hyper-rectangles are merged to form dense hyper-regions, and the centroids of these dense hyper-regions are computed. Each event is assigned to its nearest dense hyper-region centroid to form clusters, then the centroids of these clusters are computed, and each event is assigned to its nearest cluster centroid. These cluster assignment and centroid computation steps are effectively k-means iterations, initialised using FLOCK's dense hyper-region centroids. The main advantages of FLOCK compared to the k-means algorithm itself are that it does not require the number of centroids to be pre-specified and its initial centroids are determined using a density-based approach. While FLOCK does not require the number of centroids to be pre-specified, it is still dependent on the choices for the number of bins, t, and the denseness threshold, both of which are automatically determined by FLOCK using data-driven procedures.

3.2.2 flowMeans

flowMeans (Aghaeepour et al., 2011) was one of the top performing methods in the FlowCAP I challenge (The FlowCAP Consortium et al., 2013). flowMeans consists of a k-means algorithm followed by cluster merging based on Mahalanobis distances. The initial number of centroids used in the k-means algorithm is deliberately chosen to be high because many of the k-means clusters will later be merged. This initial number of clusters is chosen via a mode-finding approach based on the number of changes in the signs of the kernel density estimate gradients for each projection of the data onto one of its eigenvectors. flowMeans then applies a hierarchical agglomerative clustering algorithm to the final k-means centroids. The dissimilarity between each pair of centroids used in this agglomerative clustering was the minimum of the Mahalanobis distances between them with respect to each cluster's covariance matrix. The final number of merged clusters is given by the breakpoint of a two-segment linear regression model fitted to the dissimilarities between each pair of merged clusters.

3.2.3 SPADE

Qiu et al. (2011) introduced the SPADE (Spanning-tree Progression Analysis of Density-normalised events) method. The original algorithm was stochastic, however, Qiu (2017) proposed a deterministic version of the SPADE method. Both approaches have three main steps. SPADE down-samples the data to select a subset of the data without very low density events and with fewer events from high-density regions. Then, it clusters the down-sampled events, and finally it classifies the unsampled events as belonging to the same cluster as their nearest neighbour in the down-sampled subset. It also visualises its clustering solution using a minimum spanning tree.

SPADE requires a target density (TD) and an outlier density (OD), then it computes the local density (LD_i) of each event i, which is defined to be the number of events in a fixed-radius L_1 neighbourhood around event i. The radius of this neighbourhood is given by α times the median nearest-neighbour distance across all events. It computes a weight, w_i, for each event, which can be characterised as $w_i = \mathbb{I}(\mathsf{LD}_i > \mathsf{OD}) \times \min\left(1, \frac{\mathsf{TD}}{\mathsf{LD}_i}\right)$. In the stochastic SPADE algorithm (Qiu et al., 2011), the user specifies TD and the number of events in the down-sample is stochastic, with each event having probability w_i of being included. However, in the deterministic version of SPADE, the user specifies their desired target number of events, $T_N \leq N$, then TD is chosen such that $\sum_{i=1}^{N} w_i = T_N$, and a deterministic sampling procedure is implemented which relies on the weights, w_i. Qiu et al. (2011) then applied a stochastic agglomerative clustering procedure, whereas Qiu (2017) applies the k-means algorithm, initialised deterministically using a divisive hierarchical clustering approach. The outlier density, OD, is specified by the user in terms of a percentile of the local densities, LD_i, for both stochastic and deterministic SPADE, and so is the target density, TD, for stochastic SPADE. Both variants of SPADE require the user to specify the desired final number of clusters.

3.2.4 FlowSOM

FlowSOM (Van Gassen et al., 2015) performed well in a review and applied comparison by Weber and Robinson (2016). It implements a self-organising map (SOM; Kohonen, 1990) and then 'meta-clusters' the nodes of the SOM via a consensus hierarchical clustering approach. There is an elbow method for automatically determining the number of meta-clusters, but it is advised for the user to manually specify this based on domain knowledge.

Self-organising maps can be viewed as an extension of the k-means algorithm which allows information to be shared between cluster centroids. They can also be formulated as a neural network and are an example of 'competitive learning'. They involve establishing a rectangular grid with each node assigned a location in the data space. The algorithm iterates across all events, determining each event's nearest node and moving the node towards the event. However, unlike in k-means, the grid neighbours of that node also move towards the selected event, though to a lesser extent.

3.2.5 PhenoGraph

PhenoGraph (Levine et al., 2015) converts the data into a graph where the edge weight between two events is given by the number of k-nearest neighbours that they share. This quantity is the Jaccard similarity coefficient with respect to the events' k-nearest neighbour sets. Then, it applies the Louvain method (Blondel et al., 2008), a modularity-optimising community detection algorithm, to identify clusters where edges between events in the same cluster tend to have higher weights than edges between events in different clusters. PhenoGraph is a non-parametric density-based clustering method, since it attempts to capture the local density of each event via its k-nearest neighbours and the Jaccard similarity. The number of neighbours, k, is the main parameter which must be specified by the user. The value of k is inversely proportional to the final number of clusters. That is, increasing k tends to decrease the number of clusters. The default value of k is 30.

3.2.6 X-Shift

The X-Shift method (Samusik et al., 2016) is a non-parametric density-based clustering algorithm. It begins by computing a k-nearest neighbours density estimate for each event using its own 'fast KNN search' algorithm. Then, X-Shift iterates over the events, identifying the Z-nearest neighbours of each event and if any of these neighbours have a higher density than the selected event, it connects the event to its higher density neighbour with a connected edge. Otherwise, if the selected event has a higher density estimate value than all of its Z-nearest neighbours, then it is designated as a candidate centroid. If two candidate centroids are Gabriel neighbours, that is, no other centroids lie in the sphere with diameter given by the line between these two centroids, and there is no density minimum between them, then X-Shift will connect the centroid with lower density to the other. The lower density centroid is no longer considered a candidate centroid. Each event should either be a candidate centroid or have a series of directed edges leading from it to a candidate centroid. X-Shift assigns each event to a cluster corresponding to the centroid to which it is connected. As a final step, X-Shift performs hierarchical agglomerative clustering on the set of clusters with respect to a Mahalanobis distance until this distance is greater than two for each pair of clusters. The values of Z and K are both automatically determined from the data, through a straightforward calculation $\left(Z = -\frac{\log_2(p)}{n}\right)$ and a 'line-plus-exponent' regression of cluster number versus K, respectively. However, the user can specify K directly and can control Z via the p-value involved in its calculation.

3.2.7 cytometree

cytometree (Commenges et al., 2018) is a binary tree which recursively partitions the data with respect to one variable at a time. Each of these variables is split at the decision boundary of a two-component univariate Gaussian mixture model (GMM). The mixture components of the GMM are assumed to have equal variance terms. In order to decide which variable, if any, should be partitioned next, cytometree evaluates each potential

split's normalised AIC difference. They define the normalised AIC difference as $D = \frac{\text{AIC}_1 - \text{AIC}_2}{n_j}$ where n_j is the number of events at node j prior to the split, and AIC_1 and AIC_2 refer to the Akaike Information Criterion (Akaike et al., 1973) value of one-component and two-component GMMs fitted to these events. The AIC of a fitted model is given by $2\kappa - 2\ln(\hat{\mathcal{L}})$ where κ is its number of estimated parameters and $\hat{\mathcal{L}}$ is its likelihood. A variable cannot be partitioned more than once for the same node. In other words, if during the construction of a node one of its parent nodes was partitioned with respect to a certain variable, then that node cannot be partitioned with respect to that variable. This does not preclude a variable from being used several times on different branches of the same tree. For a node of the tree to be split into two new nodes, there must exist a variable which was not involved in the construction of that node and which has a normalised AIC difference of greater than t, where t is a user-defined threshold. If multiple variables satisfying those conditions are available, then the one with the highest normalised AIC difference will be chosen. The parameter t plays a role in controlling the height / depth of the tree. Decreasing t tends to increase the number of splits made, which increases the height of the tree and the number of leaf nodes / final clusters. cytometree also has a procedure for annotating the expression level of its final clusters as '$-$', '$+$' or '$++$' for all markers, including those that were not used in the construction of the tree. This involves clustering the univariate cluster means into one, two or three groups, as chosen by the user or based on a comparison of univariate Gaussian mixture models with one, two, or three components.

3.3 Conclusion

The methods outlined in Section 3.2 are among the leading cell population identification algorithms. They have shown strong performance when applied to cytometry data both in comparative reviews and in their respective papers. Moreover, they demonstrate some of the most prominent non-model-based approaches, including centroid-based methods involving k-means or self-organising maps, non-parametric density-based methods involving k-nearest neighbours density estimates, and hierarchical methods, both

agglomerative and divisive. One advantage of non-model-based methods is their capacity to heuristically mix and match elements of different clustering algorithms, and many of the methods in this section have innovatively combined two or more of the aforementioned approaches to leverage their respective strengths.

There are many overlaps and similarities between these methods. FLOCK, flowMeans, and deterministic SPADE all implement some form of the k-means algorithm, while FlowSOM constructs a self-organising map, which can be considered to be an evolution of k-means. X-Shift is a clear example of a non-parametric density-based clustering algorithm, however, several other methods at least partially follow the same philosophy. SPADE's density-dependent down-sampling, FLOCK's grid-based dense hyper-regions, and PhenoGraph's kNN-Jaccard similarity graph all utilise non-parametric density estimation. flowMeans, FlowSOM, and X-Shift all apply some variant of agglomerative hierarchical clustering to merge their initial clusters, while stochastic SPADE applies it directly to its down-sampled events. Meanwhile, cytometree's binary tree is a divisive hierarchical approach and deterministic SPADE initialises its k-means implementation using a divisive hierarchical clustering algorithm.

Although the k-means algorithm tends to construct spherical clusters, most of the methods in this section that are based on this approach have the ability to construct clusters with a variety of shapes. The final step of the FLOCK algorithm consists of a few k-means iterations, but the shape of its clusters will be more strongly determined by its dense hyper-region approach, which should be quite flexible in terms of cluster shape. flowMeans is more heavily based on the k-means algorithm. The degree to which k-means' spherical cluster tendency will affect flowMeans will depend on the relationship between its initial number of centroids and its final number of clusters after merging. At one extreme, if the final number of clusters was chosen to be identical to the initial number of clusters, then no merging will take place and flowMeans simplifies to a k-means implementation. On the other hand, if the number of k-means centroids is much higher than the number of merged clusters, then flowMeans will be very flexible in the possible shapes of its clusters, but the k-means stage will effectively be reduced to a data

Table 3.2: The parameters required by the non-model-based methods, both data-driven and user-specified.

Method	Parameter	Data-Driven	User-Specified
FLOCK	number of bins, t	✓	x
	denseness threshold	✓	x
flowMeans	initial number of centroids	✓	x
	final number of clusters	✓	x
SPADE	neighbourhood radius coefficient, α	x	✓(default)
	outlier density percentile	x	✓
(stochastic)	target density percentile	x	✓
(deterministic)	target number of events, T_N	x	✓
	final number of clusters	x	✓
FlowSOM	initial grid dimensions	x	✓(default)
	final number of clusters	✓	✓
PhenoGraph	number of neighbours, k	x	✓(default)
X-Shift	number of neighbours, K	✓	✓(default)
	p-value for number of neighbours, Z	x	✓
cytometree	normalised AIC difference threshold, t	x	✓(default)

reduction step. FlowSOM uses a self-organising map instead of standard k-means, which may provide it with greater flexibility, but it will have a similar relationship between its initial number of grid nodes and its final number of meta-clusters. Deterministic SPADE applies a standard k-means algorithm as its final step, as a result, it may inherit k-means' tendency to identify spherical clusters. Stochastic SPADE's hierarchical agglomerative clustering procedure would not suffer from this limitation, and neither would the non-parametric density-based methods, PhenoGraph and X-Shift. cytometree's binary tree approach imposes greater restrictions on the shape of its clusters, since the univariate splits that it implements will partition the data space into unions of orthants. However, by sacrificing the ability to form fully multivariate clusters, cytometree achieves a highly interpretable and actionable population identification solution.

We have seen in Chapter 2 that model-based methods can vary significantly depending on which model assumptions are chosen. Having to make assumptions about the distribution of the data can be off-putting, and this can lead to model-based methods being seen as less flexible than alternative approaches. Non-model-based algorithms, on the other hand, can sometimes give the impression that they offer an off-the-shelf solution. However, each

method discussed in this chapter has at least one parameter which can strongly influence its clustering solution. Therefore, non-model-based approaches also require choices to be made, even if they are made by the method's creator when choosing a default value or a heuristic procedure for determining a data-driven value for the parameter. Table 3.2 lists the parameters required by each of these methods and whether they are computed using a data-driven procedure or specified by the user. These parameters increase the adaptability of their methods and allow researchers to adjust the algorithms to suit their requirements, however, post-hoc parameter tuning can also reduce the objectivity of unsupervised clustering algorithms and affect the reproducibility of an analysis.

Although both model-based and non-model-based methods display variety and adaptability, there are differences in how they incorporate these adaptations. Model-based approaches lend themselves to modification, as discussed in Chapter 2, but only up to a point. The mixture model framework imposes some limitations because the clustering algorithm can only be modified in one of two ways: by changing the model assumptions, which results in a different mixture density function, or by changing the model fitting process, which should remain optimal with respect to the mixture density.
Non-model-based methods do not face such restrictions and can freely combine various clustering techniques. However, this freedom comes with a trade-off in terms of credibility and justifiability. The mixture model framework may limit clustering algorithms but in doing so it ensures that they maintain a principled and statistically valid approach. On the other hand, combining too many techniques can make a non-model-based method hard to interpret, as the contributions of its subcomponents can be difficult to distinguish. Despite this, since some of the basic non-model-based techniques are intuitive and straightforward, combining two or more of them may be easier to understand than a model-based clustering method, particularly for those more familiar with these approaches.

4 User-Informed Methods for Flow Cytometry

4.1 Introduction

Classification and clustering both involve assigning data points to groups. In classification, the group structure is learned from training data which have been assigned to their true classes or 'labelled'. A trained classifier then applies the rules that it developed from the labelled data to assign new data to its learned classes. In clustering, the group structure is learned directly from the unlabelled data. This means that it does not require training data, and that the groups to which it assigns data are developed based solely on the distribution of the data themselves and the assumptions of the clustering algorithm. Classification and clustering are examples of supervised and unsupervised learning, respectively. The term semi-supervised learning can sometimes be used to refer to methods that learn from both labelled and unlabelled data, combining clustering and classification.

In this chapter, we will examine four cell population identification methods which operate in a slightly different paradigm. In particular, as listed in Table 4.1, we will discuss flowDensity (Malek et al., 2015), ACDC (Lee et al., 2017), the Mondrian tree method by Ji et al. (2018), and GateMeClass (Caligola et al., 2024). These methods exploit additional information provided by the user about the cell populations that they are targeting. Hence, they are not fully unsupervised in the traditional sense, but they do not

learn from labelled training data like supervised or semi-supervised clustering methods. (GateMeClass can learn its table from a labelled training sample, but for the purposes of this chapter, we will focus on its user-informed setting.) For example, flowDensity (Malek et al., 2015) requires a manual gating strategy, while ACDC (Lee et al., 2017), Ji et al. (2018), and GateMeClass (Caligola et al., 2024) are all informed by a table of cell population descriptions, which we denote by $\mathcal{T}$.

Table 4.1: The user-informed methods discussed in Chapter 4.

Method	Reference	Information
flowDensity	Malek et al. (2015)	Gating Strategy
ACDC	Lee et al. (2017)	Table
Mondrian trees	Ji et al. (2018)	Table
GateMeClass	Caligola et al. (2024)	Table

flowDensity recreates a manual gating strategy which has been provided by the user, while recursively implementing univariate splits whose locations are determined based on the distribution of the data. Therefore, flowDensity requires the split order to be user-provided and determines data-driven split locations. ACDC, the Mondrian tree method, and GateMeClass all employ tables of cell population descriptions of the format introduced by Lee et al. (2017), but utilise this information very differently. Table 4.2 is an example of this format. ACDC and GateMeClass both simultaneously partition all of the markers before grouping the resulting subdivisions into regions matching the characteristics described in the table, whereas the Mondrian tree method recursively partitions the data. Moreover, rather than use these regions as the final partition, both ACDC and GateMeClass perform further steps to refine their cell populations, with ACDC applying the PhenoGraph algorithm and random walk classification and GateMeClass implementing a k-nearest neighbour classification. The Mondrian tree method, on the other hand, constructs cell populations directly from the recursive partitions in its trees, resulting in a more unified and cohesive approach. However, many researchers in this field may be more familiar with PhenoGraph and k-nearest neighbour methods than they would be with Bayesian modelling.

Table 4.2: A basic example of a 'cell type-marker' table of the format introduced by Lee et al. (2017). Here, we are stating that the B-cell population has a positive expression level for the CD19 marker but a negative expression level for the CD3 marker. We have left the B-cell population's expression levels for the CD4 and CD8 markers undefined. We denote such a table by $\mathcal{T}$.

	CD3	CD19	CD4	CD8
B Cells	-1	$+1$	0	0
CD4+ T Cells	$+1$	-1	$+1$	-1
CD8+ T Cells	$+1$	-1	-1	$+1$

4.2 Methods

4.2.1 flowDensity

flowDensity (Malek et al., 2015) is designed to closely replicate a given manual gating strategy. It implements a series of univariate splits to partition a sample using the same variables in the same order. This involves recursively partitioning the data with respect to one variable at a time, in the same way as a binary tree. Although flowDensity does not choose the order in which it partitions the variables, it does choose the locations at which each variable is split. Its primary approach when choosing a split location for a particular variable is to estimate the variable's distribution by applying a cubic smoothing spline to a kernel density estimate, and then to select a pair of adjacent density peaks and split the variable at the minimum point of the smoothed density curve between the two peaks. If flowDensity identifies more than two peaks, then it computes a score for each peak which is given by the formula $\frac{d}{h}$, where h is the peak's height and d is the distance from the peak to its next adjacent peak, that is, the subsequent peak to the right of the peak being evaluated. The peak with the highest score, $\frac{d}{h}$, and its next adjacent peak are chosen. This score prefers pairs of adjacent peaks that are well-separated horizontally, relative to the vertical height of the left peak of the pair. If the estimated distribution is unimodal, that is, if only a single peak is identified, then flowDensity will split the variable at an estimated changepoint in the slope of the density curve, before or after the peak, as specified by the user. flowDensity can also implement alternative methods to determine

the split location based on additional information provided by its user. These alternative methods include placing the split at a percentile of the data or a certain distance from the peak, in terms of standard deviations, where both the choice of percentile and the number of standard deviations are user-specified.

4.2.2 ACDC

The ACDC method (Automated Cell-type Discovery and Classification; Lee et al., 2017) utilises a cell-type marker table provided by the user. In this table, each column represents a marker / variable, m_k, and each row represents a cell type / population, c_j. The entries of the table, $s(c_j, m_k)$, are $+1$, -1, or 0, corresponding to positive expression, negative expression, and undefined. Lee et al. begin by using score functions based on univariate two-component Gaussian mixture models (GMMs) to partition the data set using orthogonal hyperplanes such that each resulting region corresponds to one of the cell types in the table or a single 'unknown' type. Then they apply PhenoGraph to the data and designate its cluster centres as landmark points for each cell type, before classifying the events to these cell types based on random walks over a nearest neighbour graph.

Univariate two-component GMMs are fitted to each marker, m_k, without an equal variance constraint. For any event, $x_i, = (x_{i1}, \ldots, x_{iK})$, the posterior probability of assigning x_{ik} to the positive component of the two-component GMM for marker k, m_k, is denoted by $P_k(1|x_{ik})$, and the probability for the negative component is denoted by $P_k(0|x_{ik}) = 1 - P_k(1|x_{ik})$. Since ACDC does not impose an equal variance constraint, P_k is not monotonic and the two-component GMM will have two decision boundaries. a_k is defined to be a decision boundary of the two-component GMM for marker m_k, where $P_k(0|a_k) = P_k(1|a_k)$, and b_k is defined to be the slope of P_k at this point. Lee et al. (2017) do not specify which boundary is selected as a_k. The posterior probability P_k is approximated by the monotonic function,

$$\tilde{P}_k(1|x) = \frac{\exp((x - a_k) \times b_k)}{1 + \exp((x - a_k) \times b_k)}.$$

Lee et al. define a cell-type assignment score, $f(x_i, c_j) = \min_{k:s(c_j,m_k)\neq 0} \tilde{P}_k(s(c_j, m_k)|x_{ik})$, and an unknown-type score, $f(x_i, \text{unknown}) = 1 - \max_{c_j} \tilde{P}_k(s(c_j, m_k)|x_{ik})$. These score functions are used to partition the data set into regions, S_j, corresponding to each cell type, c_j, which are defined by $S_j = \left\{x_j | f(x_i, c_j) > \frac{1}{2}\right\}$.

PhenoGraph is applied to the full data set and its cluster centres are treated as landmark points for whichever cell type is associated with the region, S_j, that they lie inside. For each event and cell type, the probability that the first landmark point reached by a random walk starting at that event belongs to that cell type is computed based on random walks over a 10 nearest neighbour graph. This probability is used to assign each event to a cell type.

4.2.3 Bayesian Trees with Mondrian Process Priors

Ji et al. (2018) developed a Bayesian tree algorithm with a Mondrian Process prior informed by a table of the format introduced by the ACDC method. A node of the tree will be partitioned if the table, $\mathcal{T}$, at that node has more than one row, $|\mathcal{T}| > 1$, and a sample from an exponential distribution, $\hat{t} \sim \text{Exp}(\sum_d \gamma_{(d)}|\mathcal{X}_d|)$, is greater than a pre-specified lifetime parameter, λ_0. Here, $|\mathcal{X}_d|$ is the range of the data, $\mathcal{X}$, for variable d, and the hyperparameter, $\gamma_{(d)}$, will take one of three user-specified values, $\gamma_{(d)} \in \{\gamma_0, \gamma_1, \gamma_2\}$, based on which entries are present in column d of the node's table. For each node of the tree, the next variable to be split, $\hat{d}$, is sampled from a categorical distribution with probabilities $p_d \propto \gamma_{(d)}|\mathcal{X}_d|$. The unscaled cut point, $\hat{r}$, for the chosen variable, $\hat{d}$, is sampled from a beta distribution with parameters $\alpha_{\hat{d}}$ and $\beta_{\hat{d}}$, which will each take one of three user-specified values, $\alpha_{\hat{d}}, \beta_{\hat{d}} \in \{\phi_0, \phi_1, \phi_2\}$, based on which entries are present in column $\hat{d}$ of the node's table. Since the support of a beta distribution is the unit interval, the cut point or split location is given by $\hat{c} = (1 - \hat{r}) \min\{\mathcal{X}_{\hat{d}}\} + \hat{r} \max\{\mathcal{X}_{\hat{d}}\}$. A likelihood is obtained for the data by modelling the events at each leaf node of the tree using a multivariate Gaussian distribution. Ji et al. perform approximate inference by sampling tree structures from the prior rather than conditioning on the data. Then, given the sampled tree structures, they use MCMC to sample the cut points / split locations.

Table 4.3: The values of the three hyperparameters (right) for different combinations of signs which are present for a given marker (left).

Marker Signs			Hyperparameters		
$+1$	-1	0	$\gamma_{(d)}$	α_d	β_d
✓	✓	✓	γ_0	ϕ_0	ϕ_0
✓	✓	x	γ_1	ϕ_0	ϕ_0
✓	x	✓	γ_2	ϕ_0	ϕ_1
x	✓	✓	γ_2	ϕ_1	ϕ_0

This results in multiple tree structures and multiple MCMC samples per tree structure. For every sampled tree, Ji et al. associate each cell population described in the table with the leaf nodes that match their described characteristics. They then conduct a majority vote over the ensemble of sampled trees to obtain a single classification solution.

The relationship between the marker signs present in column d of the table and the values of the hyperparameters $\gamma_{(d)}$, α_d, and β_d is described by Table 4.3. If for marker d, there are both cell types defined as positive, $+1$, and cell types defined as negative, -1, then $\alpha_d = \beta_d = \phi_0$. In this case, if there are no cell types defined as neutral, 0, then $\gamma_{(d)} = \gamma_0$, and otherwise, $\gamma_{(d)} = \gamma_1$. If there are both neutral and positive cell types or both neutral and negative cell types, but not both positive and negative cell types, then $\gamma_{(d)} = \gamma_2$. In the first case, neutral and positive, $\alpha_d = \phi_0$ and $\beta_d = \phi_1$, while in the second case, neutral and negative, $\alpha_d = \phi_1$ and $\beta_d = \phi_0$.

Ji et al. state that the options, γ_0, γ_1, γ_2, for the hyperparameter, $\gamma_{(d)}$, are in decreasing order in terms of magnitude. That is, $\gamma_0 \gg \gamma_1 \gg \gamma_2$. This means that markers with $\gamma_{(d)} = \gamma_0$ are more likely to be selected to be split next than markers with $\gamma_{(d)} = \gamma_1$ or $\gamma_{(d)} = \gamma_2$, and markers with $\gamma_{(d)} = \gamma_1$ are more likely to be selected than markers with $\gamma_{(d)} = \gamma_2$. However, if the range of a marker is relatively small, this will decrease the likelihood of selecting that marker, since the probabilities of the categorical distribution used to sample which variable is split next are proportional to the hyperparameter, $\gamma_{(d)}$, multiplied by the current range of the marker, $|\mathcal{X}_d|$. This is likely based on the intuition that a marker which contains events with both positive and negative expression levels will have a wider range of values, and may prevent redundant splits of markers which have not

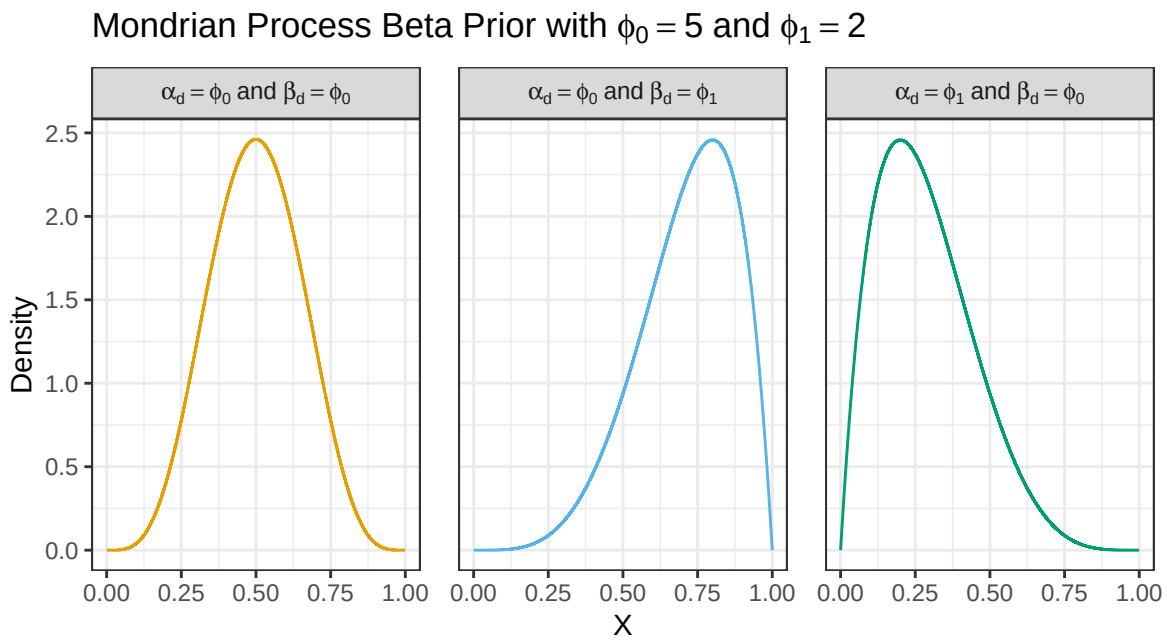


Figure 4.1: Plots of the Mondrian tree method's beta prior distributions in the different settings outlined in Table 4.3, using the values of ϕ_0 and ϕ_1 mentioned in Ji et al. (2018).

yet been split but whose negatively or positively expressed events have been removed by a previous split with respect to another marker. When $\alpha_d = \beta_d$, the beta distribution from which the unscaled cut point is sampled will be symmetric with a mean of $\frac{1}{2}$, which, after min-max rescaling, will correspond to the midpoint of the marker's range. When $\alpha_d \neq \beta_d$, the beta prior distribution will be skewed, as shown in Figure 4.1. If $\alpha_d > \beta_d$, then the beta prior will be negatively skewed, encouraging splits at higher marker values. Likewise, if $\alpha_d < \beta_d$, then the beta prior will be positively skewed, encouraging splits at lower marker values. In Ji et al. (2018), the values of ϕ_0 and ϕ_1 used were 5 and 2. With these values, Table 4.3 states that for a marker with signs $\{+1, 0\}$, the hyperparameters would be $\alpha_d = \phi_0 = 5$ and $\beta_d = \phi_1 = 2$, therefore, the beta prior would be negatively skewed, encouraging a split further to the right, as shown in the second panel of Figure 4.1. Similarly, a marker with signs $\{-1, 0\}$ would have a positively skewed prior distribution, encouraging splits further to the left.

The hyperparameters are updated recursively as the tree is constructed. After splitting variable $\hat{d}$, two smaller tables, $\mathcal{T}_{\hat{d}-}$ and $\mathcal{T}_{\hat{d}+}$, are constructed from the table, $\mathcal{T}$. $\mathcal{T}_{\hat{d}-}$ consists of the rows corresponding to cell types that were negatively or neutrally defined

for marker $\hat{d}$, while $\mathcal{T}_{\hat{d}+}$ contains those that were positively or neutrally defined. Since $\gamma_{(d)}$, $\alpha_{\hat{d}}$, and $\beta_{\hat{d}}$ are defined based on the table, their values will update during the tree's recursive construction. Meanwhile, the data, $\mathcal{X}$, is partitioned into two parts, $X_{<\hat{c}}$ and $X_{>\hat{c}}$, consisting of the events with values for marker $\hat{d}$ which are less than or greater than the cut point, $\hat{c}$. Therefore, $|\mathcal{X}_d|$, the range of the data, will also be updated during the recursion.

Although the Mondrian tree method implements univariate splits, that is, partitions which are orthogonal to the axes, it evaluates these splits through their effect on a multivariate model. Ji et al. model the events using a multivariate Gaussian mixture model with a hard component assignment based on the tree's leaf nodes. This hard assignment means that the Gaussian mixture's components will be truncated at the tree's cut points. The distribution of the events will only influence the model's likelihood through this Gaussian mixture model, therefore the splits are evaluated based on how they affect the final multivariate model, rather than based on the univariate distribution of the marker being split.

Ji et al. also discuss a random effects model for extending this approach to multiple samples. They assume that there exists a global Mondrian tree, $\mathcal{M} = \{\boldsymbol{d}, \boldsymbol{c}\}$, where $\mathcal{M} \sim MP(\mathcal{X}, \mathcal{T}_{C\times D})$, and sample-specific Mondrian trees, $\mathcal{M}_i = \{\boldsymbol{d}, \boldsymbol{c} + \boldsymbol{\xi}_i\}$ where $\boldsymbol{\xi}_i \sim \mathcal{N}_D(\mathbf{0}, \boldsymbol{\Sigma})$. That is, they assume that each sample has the same tree structure in terms of the order in which variables are split, $\boldsymbol{d}$, but there is sample-specific variation in the cut points / split locations, $\boldsymbol{c}$, which can be modelled using a zero-mean multivariate Gaussian distribution.

4.2.4 GateMeClass

GateMeClass (Caligola et al., 2024) also uses the table format from Lee et al. (2017). However, as well as being able to utilise a user-provided table, like ACDC and the Mondrian tree method, it can also construct its own table from a labelled sample. In the latter case, since it requires a labelled training sample, GateMeClass would be operating in

the same setting as methods such as flowLearn (Li et al., 2017), DeepCyTOF (Lux et al., 2018), and CytOpT (Freulon et al., 2023) which learn from a manually gated sample. For the purposes of this chapter, we will focus on the situation where GateMeClass is provided with a table by the user. GateMeClass's table differs from those used by ACDC and the Mondrian tree method because Caligola et al. allow the user to define a population-marker pair's expression level to be positive/hi ('$+$'), medium/mid ('m'), negative/lo ('$-$'), or any ('$*$').

GateMeClass selects a simple random subsample of events, then uses two-component or three-component univariate Gaussian mixture models to categorise each event in the subsample as positive, medium, or negative for every marker. This results in a ternary vector for each event in the subsample, which Caligola et al. refer to as its "signature". The subsampled events are then classified as belonging to one of the cell populations described in the table based on their signatures, or left unclassified if their signature does not match any of the descriptions. GateMeClass constructs a training set consisting of the classified events and unclassified events from the subsample that it is more confident about based on the classes of their mutual k-nearest neighbours. All of the events that are not included in this training set, including those that were not included in the original subsample, are then assigned to one of the cell populations or the "Unclassified" class via a k-nearest neighbour classification.

GateMeClass simultaneously partitions each marker of the data set according to univariate Gaussian mixture models (GMMs). The number of components fitted for each marker is dependent on the corresponding column of the population description table. If any of the populations are defined to have "medium/mid" expression levels for a particular marker, then a three-component GMM will be fitted, otherwise a two-component GMM will be fitted to the marker. For each marker, the user must specify whether its mixture's components are constrained to have equal variance terms. In the case of a two-component GMM, imposing an equal variance constraint ensures that there will be a single decision boundary between the two component means and all events on the same side of this boundary will have the same *maximum a posteriori* component assignment.

For a three-component GMM, there will be two decision boundaries, dividing the marker into three intervals, each corresponding to a component and containing only events assigned to that component. Without an equal variance assumption, the events assigned to a particular component may have values less than and greater than the mean of another component. This means that the events assigned to a component may not all be contained within a single interval containing no events from other components.

Before fitting the univariate GMMs, GateMeClass performs a modified version of ranked set sampling (Wolfe, 2012) to mitigate univariate skewness. This sampling procedure is marker-specific and samples events from the subsample obtained at the start of the GateMeClass algorithm. It involves repeatedly sampling pairs of events and, if a marker is positively skewed, keeping the event with a larger value for this marker. In other words, if the marker is skewed to the right, then it selects the event which is further to the right. If the marker is not positively skewed, then the event with a lower value is retained. This procedure up-samples the tail of a skewed marker or the smaller mode of a bimodal marker.

After fitting a univariate GMM to each marker, GateMeClass will have a 'signature' for each event. That is, each event will be classified as positive/hi ('$+$'), medium/mid ('m'), or negative/lo ('$-$') for every marker. If an equal variance constraint has not been imposed, then the *maximum a posteriori* classification will be modified using a series of rules based around the component means. For a three-component model, these rules involve partitioning the marker into four intervals using the three component means. Any event left of the smallest component mean or right of the largest component mean is reassigned to that component. Any event from the component with the highest mean or from the component with the lowest mean which is in the interval between the means of the other two components is reassigned to the middle component. Figure 4.2 illustrates this reassignment procedure.

If an event's signature matches one of the populations described in the table, then GateMeClass classifies it as belonging to that cell population. Otherwise, it leaves the

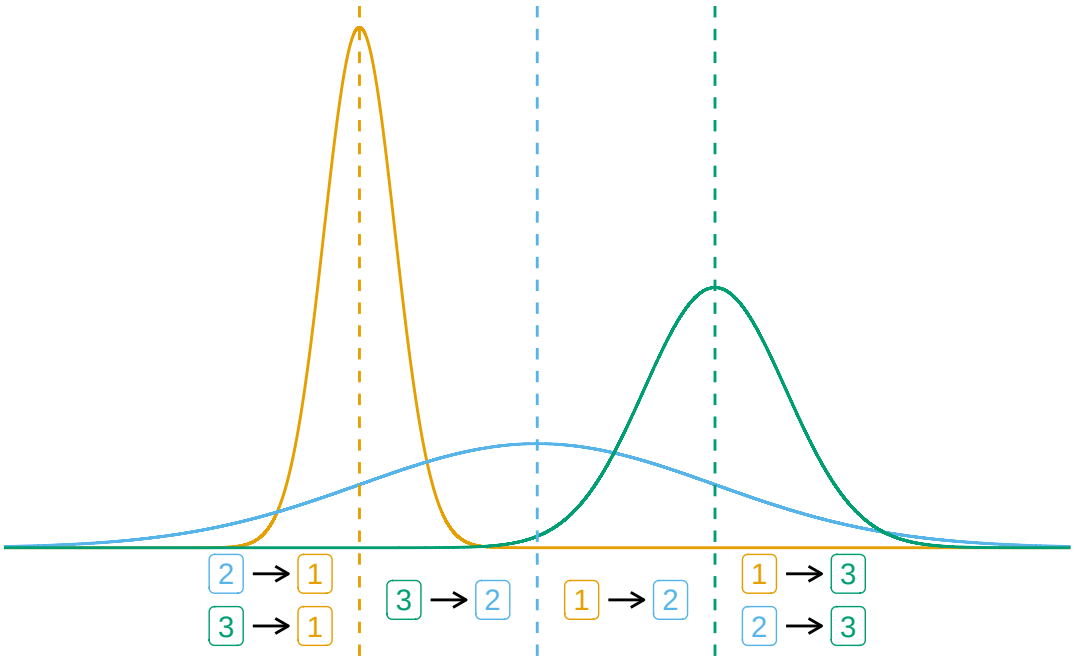


Figure 4.2: A plot of the probability density functions of three hypothetical univariate Gaussian distributions to illustrate GateMeClass' cluster reassignment of events. The vertical dashed lines mark the component means and divide the x-axis into four intervals. Each interval has one or two reassignment rules, shown below the x-axis. For example, in the left-hand interval, $(-\infty, \hat{\mu}_1)$, events assigned to components 2 or 3 of the Gaussian mixture model are reassigned to cluster 1.

event unclassified. Caligola et al. (2024) denote the sets of classified and unclassified events as A and U, respectively. GateMeClass identifies the k-nearest neighbours of every event and defines a pair of events to be mutual nearest neighbours if they are both in each other's k-nearest neighbourhood. Therefore, an event's mutual nearest neighbours form a subset of its k-nearest neighbourhood. Caligola et al. construct subsets U' and A' of U and A, respectively, consisting of those events whose classification they are confident about based on the classification of their mutual nearest neighbours. $U' \subset U$ consists of unclassified events whose mutual nearest neighbours are all unclassified events and $A' \subset A$ consists of classified events with a low proportion of unclassified mutual nearest neighbours. We will denote the full sample by X and the subsample by S. The subsample, S, is the union, $S = A \cup U$, of the classified set, A, and the unclassified set, U. Together, A' and U' form the training set, $A' \cup U'$, which is used for a k-nearest neighbour classification of the events that were not subsampled, $X \setminus S$, the unconfident classified events, $A \setminus A'$, and the unconfident unclassified events, $U \setminus U'$. That is, the

union, $(X \setminus S) \cup (U \setminus U') \cup (A \setminus A')$.

The initial subsampling and the mutual nearest neighbours refinement of the training data are both optional. Without the former, we would have $X = S$, and without the latter, we would have $A' = A$ and $U = U'$. This means that if neither were implemented, then all of the events would be classified based on their signatures and the table.

4.3 Conclusion

There is an obvious appeal in taking advantage of the wealth of biological knowledge possessed by the immunological researchers who carry out flow cytometry experiments. Particularly since the cells being analysed do not have a true 'ground truth' class and the cell populations that researchers hope to identify depend on the context of the experiment and their research. All four of the methods discussed in this chapter require the user to be able to describe their target populations as positive, negative, or undefined for each marker in the data. flowDensity further requires them to specify the order in which the markers should be partitioned, whereas ACDC and GateMeClass do not recursively partition the data, and the Mondrian tree method samples this order based on the prior hyperparameters derived from the table.

5 A User-Informed Tree Algorithm for Flow Cytometry

5.1 Introduction

Population identification has traditionally been carried out using a manual approach called gating which we have discussed in Sections 1.1 and 2.1. Despite this practice of manually identifying cell populations, most of the methods which have been developed for automated cell population identification have been unsupervised algorithms, with no information provided to the algorithm about the cell populations. However, unsupervised algorithms often require an iterative approach involving parameter tuning and post-processing to refine their initial solution and construct a set of populations that coincide with the researcher's biological knowledge and suit the purposes of their experiment (Pedersen and Olsen, 2020). This reduces the objectivity and convenience of employing an automated, unsupervised algorithm and suggests that it would be beneficial to incorporate the user's domain knowledge into the population identification process in a more principled and transparent manner.

There has been less research on developing methods which utilise biologically meaningful information from the researcher that would not be available in a typical clustering setting. The flowDensity algorithm (Malek et al., 2015) follows a pre-defined gating strategy to implement recursive univariate splits, similar to a decision tree, but only chooses the location of the splits, not the order in which the variables are partitioned. flowDensity

differs from unsupervised binary tree algorithms, such as cytometree (Commenges et al., 2018), which choose both the order in which they partition the variables, and the locations at which they are split. The ACDC method (Lee et al., 2017) proposed an approach in which a table of population descriptions is elicited from the user and its algorithm constructs cell populations with the described characteristics. The same table format was later utilised to inform Bayesian decision trees with Mondrian Process priors (Ji et al., 2018) and the GateMeClass method (Caligola et al., 2024). In this chapter, we introduce gateTree, a new method for user-informed, automated cell population identification based on a constrained decision tree algorithm. gateTree is informed by user-provided descriptions of the cell populations that it is targeting. This information is formatted according to the table introduced by the ACDC method. gateTree recursively partitions the data using univariate splits implemented by a pair of mechanisms. One of these involves finding the deepest valley in a kernel density estimate and the other consists of identifying the decision boundary of a two-component Gaussian mixture model. The latter is almost identical to the mechanism used by cytometree, while the former is similar to the mechanism used by flowDensity. gateTree is implemented in an open-source R package which can be installed from `github.com/UltanPDoherty/gateTree2`.

Decision trees recursively partition data based on whether they have high or low values for branching sequences of chosen variables. As a result of the similarities between this approach and the manual gating process, tree-based algorithms seem well-suited for cell population identification. gateTree allows the user to describe the cell populations that they are trying to identify, and this information is then used to constrain the clustering process. These constraints limit which variables gateTree can choose from when selecting the next variable on which to partition the data. In doing so, they guide the algorithm towards identifying clusters of events whose properties match those of the cell populations being targeted.

The following sections of this chapter consist of the gateTree method, and applications of gateTree to simulated and real data sets. The first four subsections of Section 5.2 discuss the splitting mechanisms that gateTree employs to recursively partition the data, the

user-informed tree constraints that direct the algorithm towards identifying the targeted populations, the multi-sample information propagation procedures which gateTree uses to ensure that all samples in an experiment are gated according to the same tree structure, and the cut-off thresholds which allow it to focus on a meaningful range of marker values. The final two subsections of Section 5.2 discuss gateTree's visualisation of individual splits and compare gateTree to some of its most closely related methods. In Section 5.3, we provide details on the main gateTree algorithm and the procedures used when implementing valley or boundary splits. Finally, in Section 5.4, we apply gateTree to simulated data, publicly available benchmark data, and data from two novel experimental data sets. For the first three data sets, we evaluated gateTree's performance by comparing its solution to manual gating, and in the final data set we compared the results of downstream statistical analyses carried out with the manual gating solution and the gateTree solution.

5.2 Methodology

5.2.1 Recursive Univariate Splitting

Decision trees partition a data set into two subsets with respect to a single variable, then attempt to further subdivide each of these subsets, proceeding in a recursive manner. Before a decision tree can partition a data set into two subsets, it must choose which variable it will use to perform the partition and the value of that variable at which the data set will be divided. This means that it must find the optimal split for each variable and then quantify the goodness of each of these splits in order to choose the best one. gateTree has two univariate splitting mechanisms: a kernel density valley mechanism and a Gaussian mixture model (GMM) boundary mechanism. The user specifies a minimum depth and a minimum difference which the density valleys and GMM boundaries are required to exceed in order to be implemented. When applied to a single sample, gateTree will begin by finding the deepest kernel density valley, but if no valley is found which meets the minimum requirements, then it will compute a GMM boundary instead. When applied

to multiple samples, the process of deciding which splitting mechanism is implemented for each sample is dependent on gateTree's multi-sample information propagation procedures which are discussed in Section 5.2.3 and further details are provided in Section 5.3.

gateTree's primary approach is to find the deepest kernel density valley. This involves obtaining a kernel density estimate for the given variable and identifying the peaks of this density estimate. In other words, the locations at which the density is higher than the surrounding region. To facilitate comparisons between variables, the density estimate for each variable is rescaled so that its maximum has a value of one. When plotted, the region between the highest peak and any other peak usually resembles a valley. gateTree calculates the difference in density between the lower of these two peaks and the lowest density point of the valley between them. gateTree computes this valley depth for every peak other than the highest one, then identifies the deepest valley. The lowest density point of the deepest valley is the split location used by gateTree. The minimum depth which valleys must exceed is specified by the user (default $\texttt{min\_depth} = 0.05$, or 5% of the maximum density). There are similarities between gateTree's valley depth and the dip calculated in Hartigan's dip test for unimodality, which computes the maximum difference between an empirical distribution function fitted to univariate data and a uniform distribution (Hartigan and Hartigan, 1985). flowDensity (Malek et al., 2015) also implements a univariate splitting mechanism based on a kernel density estimate.

gateTree's secondary splitting mechanism is to fit a two-component univariate Gaussian mixture model (GMM) with equal variance terms and use the border between the two components as its splitting location. In other words, the split location is the point at which an event would be equally likely, under the fitted model, to have arisen from either of the two components. This is the approach taken by cytometree (Commenges et al., 2018) when recursively partitioning the data to construct their binary tree. To evaluate a split arising from a GMM boundary, gateTree also fits a one-component GMM, then computes the difference between the BIC (Bayesian Information Criterion; Schwarz, 1978) scores of the two models and divides it by twice the number of data points $\left(\frac{\text{BIC}_2 - \text{BIC}_1}{2n}\right)$. Our scaled BIC difference is inspired by the normalised AIC (Akaike Information Criterion;

Akaike et al., 1973) difference used by cytometree $\left(\frac{\mathsf{AIC}_2 - \mathsf{AIC}_1}{n}\right)$. We chose to use BIC instead of AIC because of its stronger penalty on the number of estimated parameters, ν, which in this case would mean that BIC may favour the one-component model ($\nu_1 = 2$) in some situations where AIC would favour the two-component model ($\nu_2 = 4$). Therefore, BIC represents a more conservative choice than AIC, although whether or not gateTree and cytometree implement a variable split is primarily governed by the minimum threshold parameters that the computed criterion differences must exceed (default `min_diff` $= 0$ and default $t = 0.01$, respectively). If gateTree's minimum difference is set equal to zero, then a GMM boundary split exceeding this threshold is equivalent to the two-component model having a better BIC than the one-component model.

The scaled BIC difference is defined as

$$\begin{aligned}
\frac{\mathsf{BIC}_2 - \mathsf{BIC}_1}{2n} &= \frac{\left[2\sum_{i=1}^{n} \ln \hat{f}_2(x_i) - \nu_2 \ln n\right] - \left[2\sum_{i=1}^{n} \ln \hat{f}_1(x_i) - \nu_1 \ln n\right]}{2n} \\
&= \frac{1}{n}\sum_{i=1}^{n} \ln\left(\frac{\hat{f}_2(x_i)}{\hat{f}_1(x_i)}\right) - \frac{\ln n}{2n}\left(\nu_2 - \nu_1\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} \ln\left(\frac{\hat{f}_2(x_i)}{\hat{f}_1(x_i)}\right) - \frac{\ln n}{n},
\end{aligned}$$

while the normalised AIC difference has the following definition,

$$\begin{aligned}
\frac{\mathsf{AIC}_2 - \mathsf{AIC}_1}{n} &= \frac{\left[2\sum_{i=1}^{n} \ln \hat{f}_2(x_i) - 2\nu_2\right] - \left[2\sum_{i=1}^{n} \ln \hat{f}_1(x_i) - 2\nu_1\right]}{n} \\
&= \frac{2}{n}\sum_{i=1}^{n} \ln\left(\frac{\hat{f}_2(x_i)}{\hat{f}_1(x_i)}\right) - \frac{2}{n}\left(\nu_2 - \nu_1\right) \\
&= \frac{2}{n}\sum_{i=1}^{n} \ln\left(\frac{\hat{f}_2(x_i)}{\hat{f}_1(x_i)}\right) - \frac{4}{n}.
\end{aligned}$$

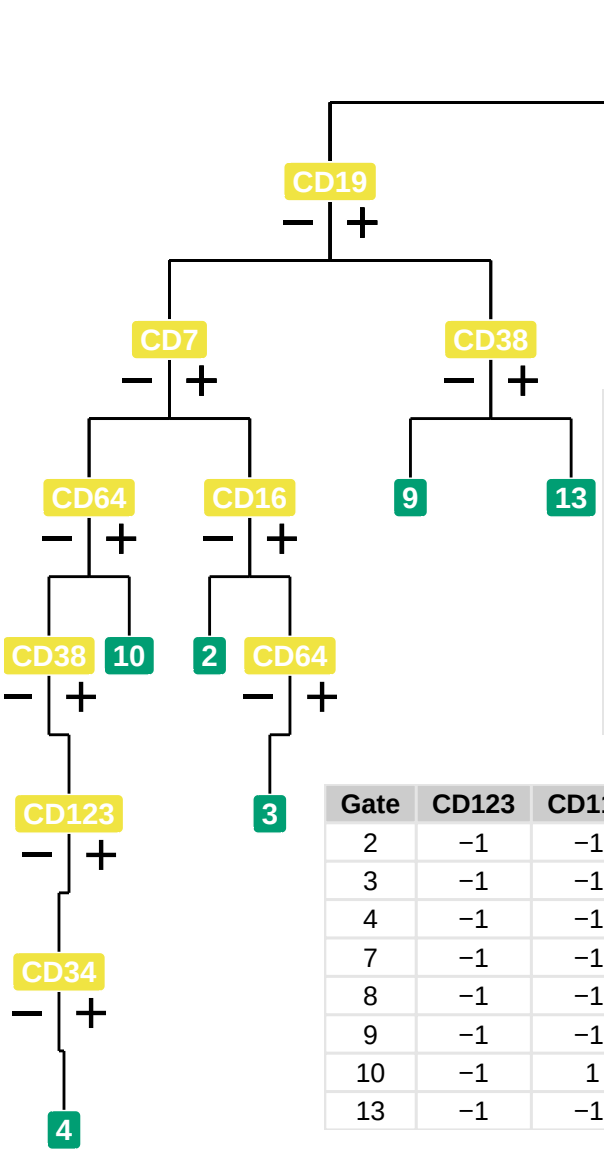
5.2.2 User-Informed Tree Constraints

An unconstrained decision tree would be free to choose from any of the variables in a data set when deciding which variable to use for its next partition. However, gateTree constrains the options available when choosing a variable to split. gateTree requires the

user to describe each of the populations they are trying to identify as positive, negative, or undefined for every variable in the data set. These descriptions must be provided in the form of a table with one row per population, one column per variable, and entries taking the values '$+1$', '-1', or '0'. This is the format proposed by Lee et al. (2017) for their ACDC method, and later used by Ji et al. (2018) for their Bayesian trees and Caligola et al. (2024) for their GateMeClass method. If a population has a positive expression level for a particular variable, then the table entry relating to that population-variable pair should be '$+1$'. Likewise, an entry of '-1' denotes a negatively expressed population-variable pair, while an entry of '0' indicates that the user has not defined the expression level of that population for that variable. Figure 5.1 shows the population description table that was used to inform our analysis of the Levine32 data in Section 5.4.2. This figure also displays the tree that was constructed by gateTree for this data based on that table.

Prior to the first split, the tree consists of a single branch and all of the target populations described in the table are subsets of the full data set which lies on that branch. After the first split, the tree will consist of two branches which we will define to be positive or negative for the variable used for that split. If a target population is defined by the user to be positive or negative for the partitioned variable, then we assume that this population lies on the corresponding new branch of the tree. That is, the events belonging to that population are mostly contained in the subset of events that have been assigned to that branch. However, if a target population was undefined for the variable being split, then it would not be reasonable to assume that its events lie on one branch or the other. gateTree's constraints ensure that this situation cannot happen.

For a variable to be a valid option for gateTree's first split, all of the target populations must be defined as positive or negative for that variable. This is because initially the tree consists of a single branch and all of the target populations are assigned to that branch. To be considered a valid option for any subsequent split, only those populations which are assigned to the branch being split must all have positive or negative descriptions for that variable. If there is no variable for which every population assigned to the branch is not



Gate	CD19	CD8	CD34	CD45	CD7
2	−1	0	−1	0	1
3	−1	0	−1	1	1
4	−1	−1	1	−1	−1
7	−1	−1	−1	1	0
8	−1	1	−1	1	0
9	1	−1	−1	1	−1
10	−1	−1	−1	0	−1
13	1	−1	−1	0	−1

Gate	CD123	CD11c	CD16	CD38	CD3	HLA.DR	CD64
2	−1	−1	−1	0	−1	−1	−1
3	−1	−1	1	0	−1	−1	−1
4	−1	−1	−1	1	−1	0	−1
7	−1	−1	−1	−1	1	−1	−1
8	−1	−1	−1	−1	1	−1	−1
9	−1	−1	−1	−1	−1	1	0
10	−1	1	−1	0	−1	0	1
13	−1	−1	−1	1	−1	0	−1

Figure 5.1: The tree constructed by gateTree for the Levine32 data and the population description table that it was informed by.

undefined, then that branch cannot be split further.

It is important to note that these constraints do not predetermine the outcome of applying gateTree. As long as there is more than one valid variable satisfying the constraints at some point in the tree, then gateTree will make a choice about which of those variables to split next. Moreover, the algorithm will always estimate the optimal location for each univariate split and it will always have a decision to make over whether to implement a split, even if it is valid according to the constraints, based on whether the split satisfies minimum quality control checks.

5.2.3 Multi-Sample Information Propagation

While gateTree can be applied to each sample from a flow cytometry experiment separately, this may result in a different tree structure for each sample, depending on the order in which the algorithm chooses to split the variables and which splits passed the checks required to be implemented. If two samples are partitioned according to different tree structures, then there will not be a direct correspondence between their clustering solutions. In other words, the cell populations that have been identified for each sample may have different characteristics.

There are two ways to guarantee that every sample in an experiment is clustered by gateTree according to the same tree structure. The first is to pool the different samples and then apply gateTree to the resulting large data set. The second is to provide gateTree with all of the samples as a single structured collection without pooling them. This means that we apply gateTree to a list of sublists of expression matrices, where the list represents the experiment, the sublists represent batches of samples, and each matrix corresponds to a single sample. In this case, gateTree will make decisions based on all of the samples simultaneously. If gateTree chooses to split on a variable for which some of the samples do not have a satisfactory split, then we replace the missing split using splits from other samples. When allowing split information to propagate between samples, we take into account the experimental structure by distinguishing between samples from

different batches and samples from the same batch. We envision each batch representing a group of samples that were collected together, however, in practice, the user should choose to group the samples based on the degree of information sharing that they believe is appropriate.

The order of preference for how gateTree identifies a sample's split is as follows: (1) a sample-specific density valley, (2) a weighted median of other valleys from the same batch, (3) a sample-specific GMM boundary, (4) a weighted median of other boundaries from the same batch, (5) a weighted median of valleys from other batches, (6) a weighted median of boundaries from other batches. If the algorithm implements splits of type (5) or (6), then an informative warning message is outputted. There is also an option in the software implementation of gateTree to prevent splits of type (5) and (6) from being implemented.

5.2.4 Event Trimming

Events with abnormally high or low marker expression values are often excluded by immunologists when they are performing manual gating. To replicate this behaviour, gateTree allows its user to specify minimum and maximum event cut-off thresholds for each marker. Events with values for a given marker that are outside that marker's user-specified event cut-offs are excluded from gateTree's populations. However, this event trimming only occurs if and when that marker is used by gateTree to perform a split.

During manual gating, immunologists can ignore some abnormalities in the data and focus on meaningful differences between events. For example, the build-up of events at one end of the cytometer's detection range would not be treated as a distinct cell population. However, since this build-up of events would cause a peak in a marker's kernel density estimate, gateTree needs to be able to ignore peaks at the extreme ends of a marker's range. To prevent erroneous splits and ensure that gateTree's split location is differentiating between the negative and positive populations, we allow the user to specify

minimum and maximum split cut-off thresholds for each marker. If an event is below the minimum split cut-off but above the minimum event cut-off, then it will be excluded from the split construction process but once a split is identified, it will be returned to the negative population. Likewise, an event above the maximum split cut-off but below the maximum event cut-off will be temporarily excluded while a split is found, then it will be assigned to the positive population. In other words, the event cut-offs permanently exclude events from the target population, whereas the split cut-offs only temporarily exclude events while a split is being found.

These two pairs of cut-off thresholds enable gateTree to explicitly replicate actions that are implicitly made by immunologists when performing manual gating. Both pairs of cut-offs are fully optional and will only be employed if specified by the user.

5.2.5 Visualisation

The software implementation of the gateTree method in R provides functions to easily visualise the splitting process and verify whether the splits being implemented by gateTree are reasonable. gateTree's `plot_split` function allows the user to view a single split plot, such as the one shown in Figure 5.2, by specifying the name or number of the split's batch, sample, variable, and population. Figure 5.2 displays the first split carried out for the fifth Rechallenged sample, which was for the CD19 marker, when constructing a population consisting of both the Vγ4 and Vγ6 $\gamma\delta$ T cells, and considering all samples to belong to the same batch. If the user does not provide either the population, variable, sample, or sample and batch, then the selection of plots that match the provided details will be returned. When producing Figure 5.2, the batch and sample were left out, so twenty plots were produced, corresponding to the twenty samples, of which this was the last one, as shown by the text "(20/20)" in the top-left corner. For this experiment, we implemented minimum and maximum event cut-offs at 0 and 255, and minimum and maximum split cut-offs at 50 and 255, as shown by the dashed red lines at 50 and 255. The text below the plot conveys that only 94,623 of the 97,511 events on this branch were used to construct the kernel density estimate that is being displayed. The remaining

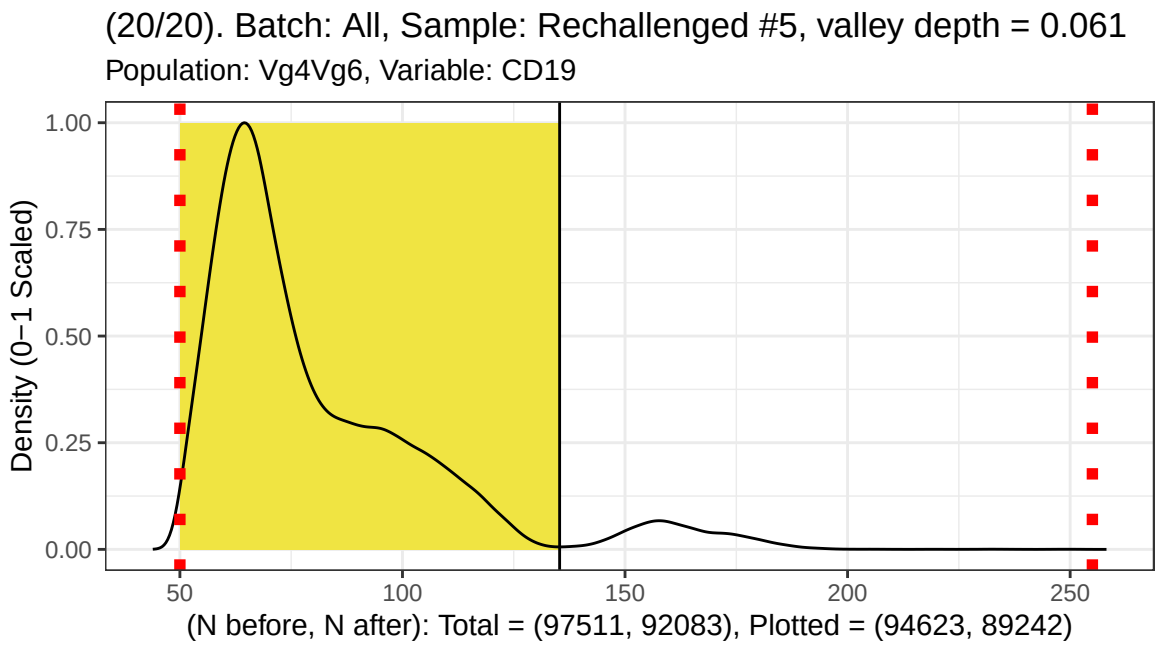


Figure 5.2: Plot of the gateTree valley split and kernel density estimate of the CD19 marker for the live lymphocytes of the fifth Rechallenged sample in the *S. aureus* experiment discussed in Section 5.5.

2,888 events were excluded by the cut-offs. This text also shows that the number of events on this branch after the plotted split will be 92,083. In other words, this split has refined the current branch by removing 5,428 events that were CD19 positive (above our chosen valley split) or had CD19 values that were below the minimum event cut-off for CD19, which we had set to be 0.

The title of Figure 5.2, as well as giving details of which split is being displayed, states that the valley depth of this split is 0.061. That is, the depth of the valley is 6.1% of the maximum density. Since, the y-axis has been rescaled so that the maximum density has a value of one, the valley depth of 0.061 is the rescaled density of the peak at $x \approx 158$ ($y \approx 0.067$) minus the rescaled density of the trough at $x = 135.31$ ($y \approx 0.006$). Based on the noisiness or smoothness of the kernel density estimates for their samples, the user may choose to increase or decrease the minimum valley depth from its default value of 0.05. For example, if the density estimate curves are particularly erratic, such that several of the splits appear to be based on noise rather than a meaningful bimodal distribution, then the user may need to increase the minimum depth parameter. On the other hand, if the density estimate curves are quite smooth and the user believes that they can discern a

bimodal distribution but its valley has a depth of less than 0.05, then they may decrease the minimum depth parameter. For GMM boundary splits, the scaled BIC difference would be stated instead of the valley depth and the background colour would be light blue instead of yellow. Splits which are propagated from other samples have an orange background colour and do not provide a depth or difference value but do state how they were propagated.

The tree plot shown in Figure 5.1 was also produced using a function in the software implementation of gateTree, namely the `plot_tree` function.

5.2.6 Related Methods

In developing gateTree, we have built on the research of three previous methods, in particular. These are the ACDC method (Lee et al., 2017), the flowDensity algorithm (Malek et al., 2015), and cytometree (Commenges et al., 2018). Another closely related method is the Bayesian tree method proposed by Ji et al. (2018).

Lee et al. (2017) introduced the 'cell-type marker table' which gateTree uses to format the cell population descriptions provided by its user. However, their ACDC method differs significantly from gateTree. Whereas gateTree recursively partitions the data according to a binary tree, the ACDC method simultaneously divides each marker to subdivide the data space into orthants, then merges these orthants to form regions with characteristics matching those described in the table. Rather than using these regions as its final clustering solution, the ACDC method applies the PhenoGraph algorithm (Levine et al., 2015) to find cluster centres and associates each region with any of these 'landmark points' that lie inside it, then assigns events to clusters based on a random walk classification using a 10-nearest neighbour graph and the landmark points generated by PhenoGraph. We believe that by constructing a single binary tree, gateTree offers a more intuitive and less heuristic method of using the cell population descriptions from ACDC's table.

Another approach that utilises this table format is the Bayesian tree method developed by

Ji et al. (2018). This method uses a Mondrian Process prior for which the hyperparameters of its 'dimension prior', a categorical distribution which is used to sample which variable to split next, and its 'cut prior', a beta distribution which is used to sample the location of the next split, are chosen based on the table's values. The leaf nodes of the Mondrian trees are modelled by disjoint multivariate Gaussian distributions.

By recursively partitioning the data in a similar manner to manual gating, gateTree and the Mondrian tree method will be able to uncover differences between cell populations that are not visible prior to implementing any splits. This is why we chose to design a tree-based method rather than simultaneously partitioning all of the markers like ACDC and GateMeClass. However, in contrast to the Mondrian tree method's Bayesian model, gateTree's more modular design allows it to flexibly incorporate univariate splitting mechanisms inspired by other methods, as well as more readily facilitating future improvements to its subcomponents, such as its splitting mechanisms or its multi-sample information propagation. Considering each split independently and univariately also provides us with a more granular view of the tree's construction and the distribution of each marker. Moreover, by sampling its tree structures from the prior without conditioning on the data, the Mondrian tree method does not allow the distribution of the events to dictate the structure of its trees, unlike gateTree.

gateTree's kernel density technique is similar to the approach used by flowDensity (Malek et al., 2015) and our GMM boundary splitting mechanism mirrors that of cytometree (Commenges et al., 2018). As well as combining these two mechanisms and extending to multi-sample applications, gateTree also offers a compromise between cytometree's unsupervised approach and flowDensity's pre-determined gating strategy.

5.3 gateTree Algorithms

5.3.1 Setup

The main gateTree function constructs one target population at a time. Each target population corresponds to a single row in the population description table. However, to determine which variables gateTree can choose from when selecting the next variable to split while satisfying the user-informed constraints, each population's construction does require the full population description table. All samples are involved in the construction of each target population. Section 5.3.2 describes how gateTree chooses which variable to split next (Algorithm 1), while Section 5.3.3 describes how gateTree determines the location of its next split (Algorithms 2 and 3).

5.3.2 Split Variable Choice

At any point in the target population construction process, when determining the next split, we begin by estimating each sample's optimal density valley split for every valid variable. For each sample-variable pair, we obtain a valley depth. If this valley depth is less than the single-sample minimum valley depth, $\gamma_{\text{min}}^{(\text{single})}$, then we set it equal to zero. We compute the mean valley depth for each variable across samples. Denote the mean valley depth of variable v by $\bar{\gamma}_v$. We select the variable with the highest mean valley depth. If the selected variable's mean valley depth is greater than or equal to the multi-sample minimum valley depth, $\gamma_{\text{min}}^{(\text{multiple})}$, then this variable will be chosen for the next split. Otherwise, we will search for a GMM boundary split. In this case, we repeat the above process with GMM boundaries instead of density valleys and scaled BIC differences instead of valley depths. As above, if a boundary's scaled BIC difference is less than the single-sample minimum scaled BIC difference, $\Delta_{\text{min}}^{(\text{single})}$, then we set it equal to zero. If the selected variable, v^*, has a mean scaled BIC difference, $\bar{\Delta}_{v^*}$, which is greater than or equal to the multi-sample minimum scaled BIC difference, $\Delta_{\text{min}}^{(\text{multiple})}$, then this variable will be chosen for the next split. Otherwise, this branch of the tree will not be

split, meaning that this target population will not be refined further.

5.3.3 Split Location Estimation

Given a chosen variable, gateTree must implement a split of that variable for every sample. However, the estimated splits for some samples may have had split scores below the minimum single-sample split score thresholds. Therefore, gateTree requires procedures for replacing these unsuitable sample-specific splits.

If the variable v^* was chosen based on density valley splits and a sample's valley split is not suitable, then it may be replaced with a weighted median of suitable valley locations from other samples in the same batch. This in-batch median valley is denoted by $\tilde{\gamma}_{v^*}^{(b)}$ where b is the batch index. If none of the samples from its batch have suitable valley splits, then gateTree searches for GMM boundaries for every sample in that batch. If the sample's GMM boundary split is not suitable, then it may be replaced with a weighted median of suitable GMM boundary locations from other samples in the same batch. If none of the samples from its batch have suitable boundary splits, then it will be replaced with a median of suitable valley splits from all samples. This all-batch median valley is denoted by $\tilde{\gamma}_{v^*}$.

If the variable v^* was chosen based on GMM boundaries and a sample's boundary split is not suitable, then it may be replaced with a weighted median of suitable GMM boundary locations from other samples in the same batch. This in-batch median boundary is denoted by $\tilde{\Delta}_{v^*}^{(b)}$ where b is the batch index. If none of the samples from its batch have suitable boundary splits, then it will be replaced with a weighted median of suitable boundary splits from all samples. This all-batch median boundary is denoted by $\tilde{\Delta}_{v^*}$.

Algorithm 1 Choosing which variable to split for a population p.

$\mathcal{P} :=$ cell populations.
$\mathcal{V} :=$ variables.
$T(q, v) \in \{-1, 0, +1\} :=$ table value for population q and variable v.
$\mathcal{P}_p :=$ populations on the same branch of the tree as population p.
$\mathcal{U}_p :=$ variables that have been split previously on this branch.
$\mathcal{V}_p \leftarrow \{v \in \mathcal{V} : T(q, v) \neq 0 \ \forall \ q \in \mathcal{P}_p\} \setminus \mathcal{U}_p :=$ variables that can be used to split this branch.

Choosing a variable using kernel density valleys:
Find a density valley for every variable, v, in $\mathcal{V}_p$ of each sample, s, in every batch, b.
Denote each valley's location by $y_v^{(b,s)}$ and its depth by $\gamma_v^{(b,s)}$.
If a valley's depth, $\gamma_v^{(b,s)}$, is less than $\gamma_{\min}^{(\text{single})}$, then set its depth equal to zero.
Compute the mean, $\bar{\gamma}_v$, of the valley depths for each variable.
If any variables have $\bar{\gamma}_v \geq \gamma_{\min}^{(\text{multiple})}$, then choose the variable, v^*, with the highest mean valley depth, $\bar{\gamma}_{v^*}$.

Choosing a variable using two-component GMM boundaries:
If all variables have $\bar{\gamma}_v < \gamma_{\min}^{(\text{multiple})}$, then repeat the above steps for GMM boundaries instead of density valleys, as follows.
Find a GMM boundary for every variable, v, in $\mathcal{V}_p$ of each sample, s, in every batch, b.
Denote the boundary's location by $z_v^{(b,s)}$ and its scaled BIC difference by $\Delta_v^{(b,s)}$.
If a boundary's difference, $\Delta_v^{(b,s)}$, is less than $\Delta_{\min}^{(\text{single})}$, then set its difference equal to zero.
Compute the mean, $\bar{\Delta}_v$, of the boundary differences for each variable.
If any variables have $\bar{\Delta}_v \geq \Delta_{\min}^{(\text{multiple})}$, then choose the variable, v^*, with the highest mean boundary depth, $\bar{\Delta}_{v^*}$.

Choosing not to split the branch further:
If all variables have $\bar{\gamma}_v < \gamma_{\min}^{(\text{multiple})}$ and $\bar{\Delta}_v < \Delta_{\min}^{(\text{multiple})}$, then no further splits will be carried out on this branch of the tree.

Algorithm 2 Selecting a split location for a population p and a variable v^* chosen using kernel density valleys.

for b in 1 to B **do**
  for s in 1 to S_b **do**
    Denote the valley split location for sample s by $y_{v^*}^{(b,s)}$ and its depth by $\gamma_{v^*}^{(b,s)}$.
    If $\gamma_{v^*}^{(b,s)}$ is less than $\gamma_{\min}^{(\text{single})}$, set $\gamma_{v^*}^{(b,s)}$ equal to zero.
  end for
  Compute the in-batch weighted median, $\tilde{y}_{v^*}^{(b)}$, of the valley locations, $y_{v^*}^{(b,s)}$, with respect to their updated depths, $\gamma_{v^*}^{(b,s)}$.
end for
Compute the all-batch weighted median, $\tilde{y}_{v^*}$, of the valley locations, $y_{v^*}^{(b,s)}$, with respect to their updated depths, $\gamma_{v^*}^{(b,s)}$.

for b in 1 to B **do**
  if $\gamma_{v^*}^{(b,s')} \geq \gamma_{\min}^{(\text{single})}$ for any sample s' in batch b **then**
    for s in 1 to S_b **do**
      If $\gamma_{v^*}^{(b,s)}$ is greater than or equal to $\gamma_{\min}^{(\text{single})}$, then select $y_{v^*}^{(b,s)}$ as this sample's split location.
      Otherwise, assign the in-batch weighted median valley location, $\tilde{y}_{v^*}^{(b)}$.
    end for
  else
    for s in 1 to S_b **do**
      Find a two-component GMM boundary for v^* of sample s in batch b.
      Denote the boundary location by $z_{v^*}^{(b,s)}$ and its scaled BIC difference by $\Delta_{v^*}^{(b,s)}$.
      If $\Delta_{v^*}^{(b,s)}$ is less than $\Delta_{\min}^{(\text{single})}$, set $\Delta_{v^*}^{(b,s)}$ equal to zero.
    end for
    Compute the in-batch weighted median, $\tilde{z}_{v^*}^{(b)}$, of the boundary locations, $z_{v^*}^{(b,s)}$, with respect to their updated differences, $\Delta_{v^*}^{(b,s)}$.
    if $\Delta_{v^*}^{(b,s')} \geq \Delta_{\min}^{(\text{single})}$ for any sample s' in batch b **then**
      for s in 1 to S_b **do**
        If $\Delta_{v^*}^{(b,s)}$ is greater than or equal to $\Delta_{\min}^{(\text{single})}$, then select $z_{v^*}^{(b,s)}$ as this sample's split location.
        Otherwise, assign the in-batch weighted median boundary location, $\tilde{z}_{v^*}^{(b)}$.
      end for
    else
      for s in 1 to S_b **do**
        Use the all-batch weighted median valley location, $\tilde{y}_{v^*}$.
      end for
    end if
  end if
end for

Algorithm 3 Selecting a split location for a population p and a variable v^* chosen using two-component GMM boundaries.

for b in 1 to B **do**
  for s in 1 to S_b **do**
    Denote the boundary split location for sample s by $z_{v^*}^{(b,s)}$ and its scaled BIC difference by $\Delta_{v^*}^{(b,s)}$.
    If $\Delta_{v^*}^{(b,s)}$ is less than $\Delta_{\min}^{(\text{single})}$, set $\Delta_{v^*}^{(b,s)}$ equal to zero.
  end for
  Compute the in-batch weighted median, $\tilde{z}_{v^*}^{(b)}$, of the boundary locations, $z_{v^*}^{(b,s)}$, with respect to their updated differences, $\Delta_{v^*}^{(b,s)}$.
end for
Compute the all-batch weighted median, $\tilde{z}_{v^*}$, of the boundary locations, $z_{v^*}^{(b,s)}$, with respect to their updated differences, $\Delta_{v^*}^{(b,s)}$.

for b in 1 to B **do**
  if $\Delta_{v^*}^{(b,s')} \geq \Delta_{\min}^{(\text{single})}$ for any sample s' in batch b **then**
    for s in 1 to S_b **do**
      If $\Delta_{v^*}^{(b,s)}$ is greater than or equal to $\Delta_{\min}^{(\text{single})}$, then select $z_{v^*}^{(b,s)}$ as this sample's split location.
      Otherwise, assign the in-batch weighted median boundary location, $\tilde{z}_{v^*}^{(b)}$.
    end for
  else
    for s in 1 to S_b **do**
      Use the all-batch weighted median boundary location, $\tilde{z}_{v^*}$.
    end for
  end if
end for

5.4 Application and Results

5.4.1 Simulated Data

In this simulation study, we sought to replicate some aspects of flow cytometry data by generating data sets consisting of populations with positive or negative expression for each variable. The values of the data points for each variable are simulated from univariate Gaussian distributions with mean, μ, and standard deviation, σ, equal to one of the following three pairs: $(\mu = -1, \sigma = 0.5)$, $(\mu = +1, \sigma = 0.5)$, or $(\mu = 0, \sigma = 0.75)$. We had four simulation scenarios with different numbers of variables, d, and populations, G. The options for these two characteristics were $(d = 3, G = 4)$, $(d = 4, G = 5)$, $(d = 5, G = 6)$, and $(d = 6, G = 7)$. We designed the data sets to have a 'telescoping' structure so that the true populations could be approximately recovered with a sequence of univariate splits. Each data set's marginal distribution with respect to the first variable is bimodal with half of the points arising from a single population with mean -1, and the other half arising from multiple populations with mean $+1$. Similarly, if we exclude the population which is negative for the first variable, then half of the remaining points arise from a single population with mean -1 for the second variable, and the other half arise from multiple populations with mean $+1$. Table 5.1 provides the variable means and sizes for the three-dimensional and six-dimensional data sets. We randomly simulated 30 samples per simulation scenario. Figure 5.3 displays scatter plots for one of the five-dimensional data sets. Each scatter plot shows a more refined subset of the data, with the negative population from the previous plot being removed to highlight the bimodality of the remaining data.

We applied gateTree and cytometree, which are both tree-based algorithms, to the simulated data sets, as well as FlowSOM (Van Gassen et al., 2015) and mclust (Scrucca et al., 2016), which are multivariate clustering algorithms implementing a self-organising map and a Gaussian mixture model, respectively. We fitted gateTree with its default minimum valley depth of 0.05 and we also tried 0.01, although these produced identical

Table 5.1: The population means and sizes for the three-dimensional (left) and six-dimensional (right) simulated data sets.

g	X1	X2	X3	Size
1	-1	0	0	400
2	$+1$	-1	0	200
3	$+1$	$+1$	-1	100
4	$+1$	$+1$	$+1$	100

g	X1	X2	X3	X4	X5	X6	Size
1	-1	0	0	0	0	0	3200
2	$+1$	-1	0	0	0	0	1600
3	$+1$	$+1$	-1	0	0	0	800
4	$+1$	$+1$	$+1$	-1	0	0	400
5	$+1$	$+1$	$+1$	$+1$	-1	0	200
6	$+1$	$+1$	$+1$	$+1$	$+1$	-1	100
7	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	100

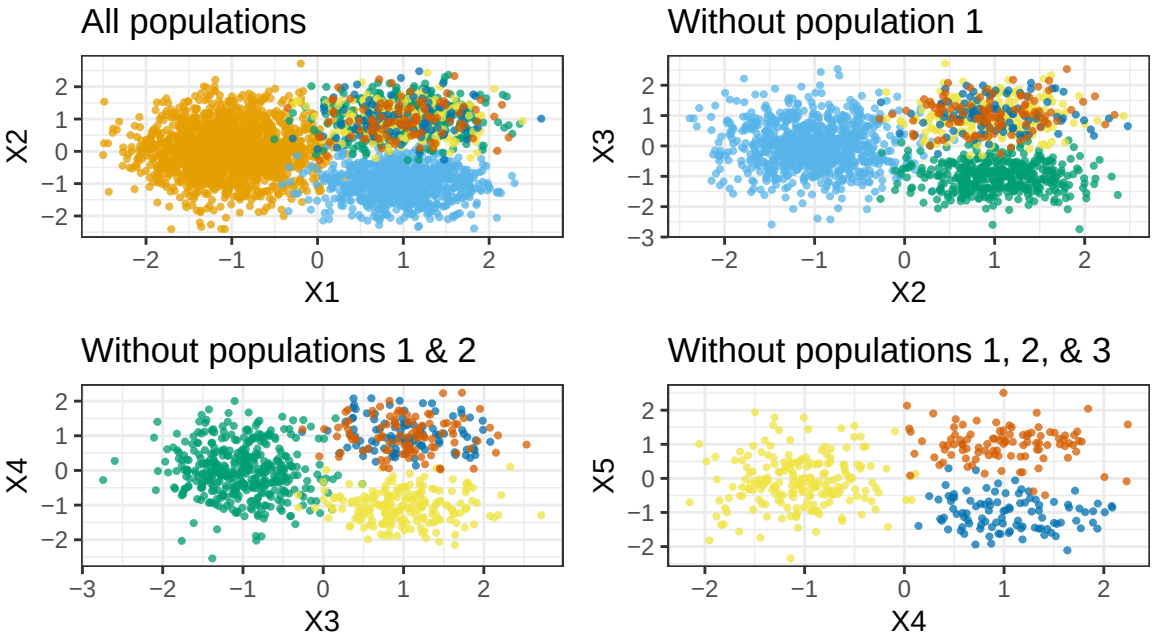


Figure 5.3: Scatter plots of a five-dimensional sample from the simulation study showing the bimodal and nested or telescoping nature of the simulated populations.

results for every sample. We ran cytometree with two values (0.01, 0.1) for its minimum AIC difference parameter, t, and we ran FlowSOM and mclust with seven values (4-10) for their number of meta-clusters parameter, `nClus`, and number of mixture components parameter, G, respectively. For each simulation setting, we chose the parameter setting which performed best on average for each method across the samples.

We evaluated the performance of each method using the F_1 measure. Since there are multiple simulated populations in each sample and each method identified several clusters per sample, we computed the F_1 for every population-cluster pair and matched each population to the cluster with which it achieved the highest pairwise F_1 score. We chose to allow a cluster to be matched to more than one population, rather than enforce a one-to-one mapping. We have implemented a similar procedure for matching clusters and populations in Sections 5.4.2 and 5.4.3. We computed a sample mean F_1 value by taking the unweighted average of the F_1 scores for each population in a given sample. For every simulation scenario and population identification method, we also computed the mean of the unweighted sample mean F_1 values to evaluate how well the methods performed in each scenario. These values are plotted in Figure 5.4.

Figure 5.4 demonstrates that the clustering tree algorithms, gateTree and cytometree, which are designed to recursively perform univariate splits, are better able to accurately recover the simulated populations than other clustering and cell population identification algorithms. The performance of the multivariate algorithms declined more than that of the tree-based methods as the number of variables increased. Despite the populations being simulated from a Gaussian mixture model, the nested nature of the populations allowed gateTree and cytometree to outperform a Gaussian mixture modelling algorithm. Since the populations are distributed according to Gaussian distributions with diagonal covariance matrices, and each pair of populations is well-separated in at least one variable, the primary challenge of clustering these data sets is the telescoping layout of the populations. In each sample, the two smallest populations only differ in terms of distribution with respect to a single variable. This is a tricky scenario for multivariate clustering algorithms because all but one of the variables are providing evidence that these

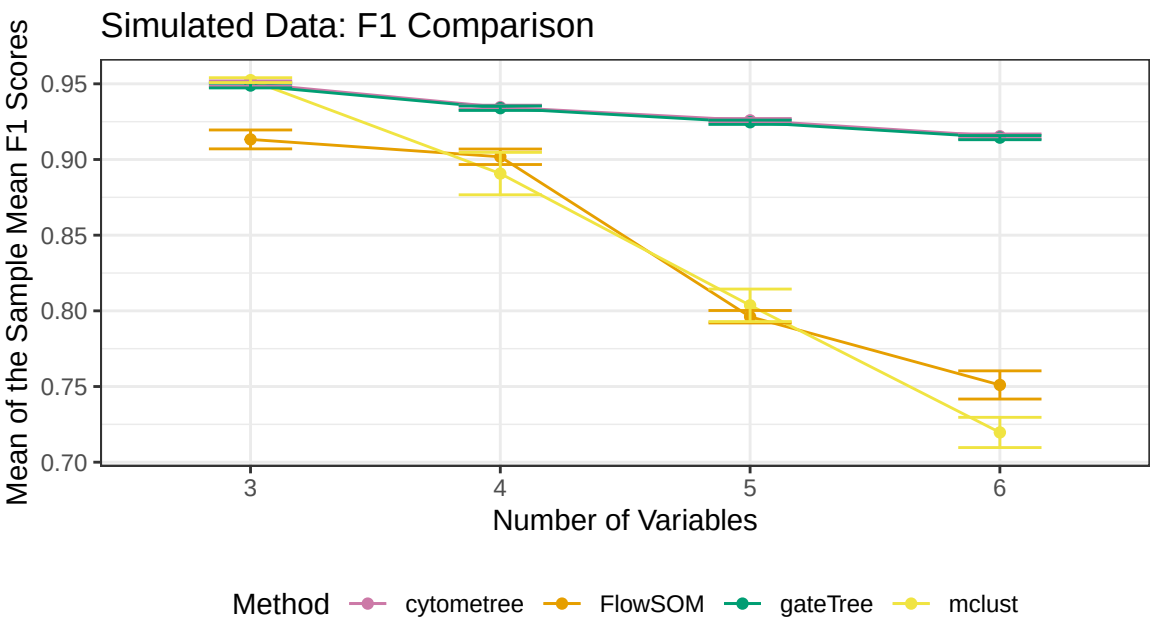


Figure 5.4: A line plot with error bars illustrating the cross-sample means of the unweighted sample mean F_1 values for every simulation scenario and method.

populations belong to the same cluster. An example of such a situation can be seen in Section 5.5, where the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations in the *S. aureus* data sets are equivalent for nine out of ten markers used by the manual gating. The strong relative performance of tree-based methods in this simulation study demonstrates that algorithms such as gateTree and cytometree may be better-suited for trying to differentiate between cell populations in cytometry data that demonstrate this deeply nested property than multivariate approaches.

5.4.2 Benchmark Data: Levine32

This mass cytometry data set featured in the PhenoGraph paper by Levine et al. (2015), and a review by Weber and Robinson (2016). It consists of two samples of human bone marrow cells analysed with respect to 32 main markers, as well as two DNA channels and a viability marker. We obtained this data set from the publicly available HDCytoData R package (Weber and Soneson, 2019), although we removed a large number of duplicated events before using these data. After removing these duplicates, there were 118,888 events and 42,555 events in the two samples. Each event has been assigned to one of

fourteen gates or left unassigned. Of these fourteen gates, four each contain more than 10% of all events and a further four each contain more than 1% of all events. We focused our analysis on attempting to identify these eight cell populations.

We applied gateTree to both samples using our structured multi-sample approach and again using a pooling approach. We ran gateTree with and without the GMM boundaries splitting mechanism. The tree constructed by gateTree for the Levine32 data, as well as the population description table used to constrain the tree construction process are shown in Figure 5.1. This table's values were determined based on the expression levels of each marker for the manually gated populations, using their percentiles and the degree of univariate overlap between them. We also applied FlowSOM and cytometree using the pooling approach and selecting only the twelve variables which were available to gateTree from its population description table. Both methods performed better with this variable selection than on the full data set with 32 markers. We ran FlowSOM with its number of meta-clusters parameter, `nClus`, ranging over nine values from eight to forty at intervals of four and we ran cytometree with its minimum AIC difference parameter, t, ranging over nine values from 0.1 to 0.9 at intervals of 0.1. For every clustering solution and sample, we matched each of that sample's eight main gates to the cluster with which it achieved the highest F_1. Then, we computed an unweighted mean of these eight F_1 scores in each sample and averaged across the two samples. This allowed us to identify which parameter setting performed best for each method. FlowSOM achieved its highest mean F_1 for $\texttt{nClus} = 20$, while cytometree did so for $t = 0.4$. gateTree's structured approach performed better without GMM boundaries and its pooled approach performed better with them. We have only reported the results for the best parameter setting of each method.

Table 5.2 shows each method's unweighted mean F_1 scores for the two samples, while Figure 5.5 displays their F_1 scores for each gate. gateTree was the best performing method across both samples with a mean F_1 score of 0.816 when using its structured approach and 0.778 when using the pooled approach. FlowSOM and cytometree achieved mean F_1 scores of 0.772 and 0.759, respectively.

Table 5.2: Each method's unweighted mean F_1 scores for the two Levine32 samples.

	Best Parameters	Sample 1	Sample 2	Mean
gateTree (structured)	Without GMMs	0.778	0.855	0.816
gateTree (pooled)	With GMMs	0.745	0.811	0.778
FlowSOM	`nClus` $= 20$	0.733	0.812	0.772
cytometree	$t = 0.4$	0.721	0.798	0.759

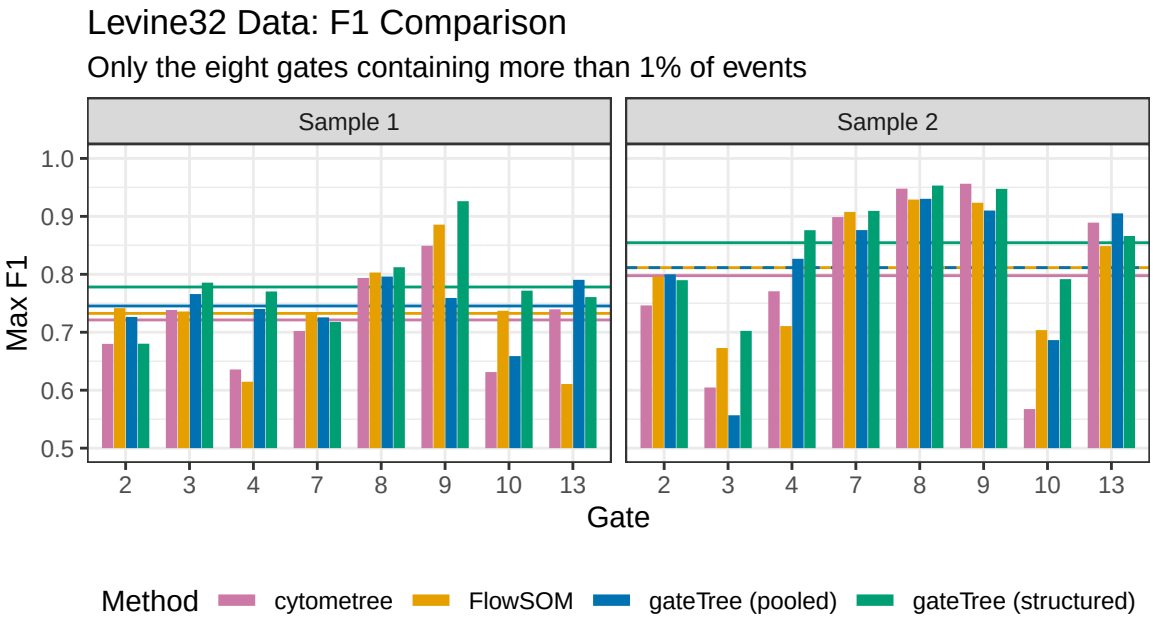


Figure 5.5: Column plots of each method's F_1 values per gate for the two Levine32 samples. For both samples, each method's unweighted mean F_1 value across populations is displayed as a horizontal line.

5.4.3 Novel Experimental Data: Haemodialysis

These data consist of twenty-four samples of human peripheral blood mononuclear cells from five experimental batches in a haemodialysis study. The data from this study are currently unpublished and were provided to us by Tracey J. Claxton. These samples were manually gated by an expert along five different gating pathways: Monocytes (Classical, Intermediate, and Non-Classical), Gamma-Delta Cells ($\gamma\delta 1$ and $\gamma\delta 2$), Natural Killer Cells (NK and NK T), B Cells, and CD3+CD8+ Cells. gateTree was applied separately for each pathway using the population descriptions in Table 5.3. We also applied FlowSOM and cytometree to a pooled collection of all samples. These methods were applied separately to each subset of variables featured in one of the population description tables. For comparison purposes, we applied gateTree once to the pooled data set and once to the full collection of samples in experiment structure.

We applied each of these methods for a range of parameter settings. For each pathway, we ran FlowSOM eight times with its number of meta-clusters parameter, `nClus`, ranging from three to ten, we ran cytometree with four values of its minimum normalised AIC difference, t, ($t = 0.001$, $t = 0.01$, $t = 0.1$, and $t = 1$), and we ran gateTree with its minimum valley depth parameter equal to 0.05 and 0.01. For each pathway and method, we are reporting the results from using the parameter that resulted in the best average score for that pathway across all samples.

We assessed how accurately the clusters constructed by these methods corresponded to the manual gates identified previously in these data. We evaluated this using the F_1 measure. The unweighted mean F_1 values for each method and population across samples are shown in Table 5.4. Computing the unweighted mean of these values gives us overall average F_1 scores of 0.887 and 0.872 for the structured and pooled applications of gateTree, as well as 0.843 for FlowSOM and 0.672 for cytometree. Although, if we exclude the Gamma-Delta pathway, then cytometree achieves an average F_1 of 0.855, which is comparable to FlowSOM (0.827), and both the structured and pooled applications of gateTree (0.886 and 0.872).

Table 5.3: The five population description tables used by gateTree for each pathway of the haemodialysis data.

Monocytes	CD 14	CD 16	CD 8	CD 56
Classical	$+1$	-1	0	0
Intermediate	$+1$	$+1$	0	0
Non-Classical	-1	$+1$	-1	-1

Natural Killer Cells	CD 56	CD 3
NK	$+1$	-1
NK T	$+1$	$+1$

Gamma-Delta Cells	CD 3	Vδ 1	Vδ 2
$\gamma\delta 1$	$+1$	$+1$	-1
$\gamma\delta 2$	$+1$	-1	$+1$

B Cells	CD 3	CD 19
B Cells	-1	$+1$

CD3+CD8+ Cells	CD 3	CD 8
CD3+CD8+ Cells	$+1$	$+1$

Table 5.4: The unweighted mean F_1 scores for each method and population across all twenty-four samples from the haemodialysis data.

Pathways	Monocytes			Natural Killer Cells	
Populations	Classical	Intermediate	Non-Classical	NK	NK T
gateTree (structured)	0.98	0.63	0.83	0.94	0.87
gateTree (pooled)	0.99	0.68	0.85	0.92	0.74
cytometree	0.94	0.63	0.84	0.92	0.74
FlowSOM	0.98	0.60	0.74	0.83	0.71

Pathways	Gamma-Delta Cells		B Cells	CD3+ CD8+ Cells
Populations	$\gamma\delta 1$	$\gamma\delta 2$		
gateTree (structured)	0.79	0.99	0.97	0.98
gateTree (pooled)	0.77	0.98	0.94	0.99
cytometree	0.04	0.03	0.94	0.97
FlowSOM	0.83	0.96	0.97	0.96

Table 5.5: The experimental conditions for each of the four groups in the *S. aureus* data.

	Day 0 Exposure	+ 3 Hours Collection	Day 7 Exposure	Day 14 Exposure	Day 35 Exposure	+ 3 Hours Collection
Naïve	x	✓	-	-	-	-
Acute	✓	✓	-	-	-	-
Convalescent	✓	x	✓	✓	x	✓
Rechallenged	✓	x	✓	✓	✓	✓

5.5 *Staphylococcus aureus In Vivo* Infection Experiment

5.5.1 Experiment Details

These data consist of twenty samples of peritoneal exudate cells (PEC) from mice in four experimental groups of five. This experiment was carried out by Dr. Charlotte M. Leane and the data from it are currently unpublished. The treatment programmes for these groups are described in Table 5.5, and we will refer to them as Naïve, Acute, Convalescent, and Rechallenged. The Acute and Rechallenged groups were both exposed to *Staphylococcus aureus* (2.5×10^8 CFU) three hours prior to sample collection, whereas the Naïve and Convalescent groups did not experience a recent infection prior to sample collection. The Convalescent and Rechallenged groups were exposed to *S. aureus* on Days 0, 7, and 14, followed by a convalescence period before sample collection on Day 35, whereas the Acute and Naïve groups underwent sample collection on Day 0. This experiment sought to identify the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations and compare their properties across the four groups. In particular, we were interested in any differences in the size of these populations, as a proportion of all lymphocytes, and any differences in the mean IL-17A expression levels of these populations. Each sample was manually pre-gated using the scatter channels and a viability marker to identify the live lymphocytes. The manual gating then proceeded to use ten of the remaining sixteen markers to identify the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations.

Table 5.6: The population descriptions used by gateTree for the *S. aureus* data.

	CD45	CD19	CD11b	CD4	CD8	$\gamma\delta$-TCR	CD3	CD27	Vγ1	Vγ4
Vγ4	$+1$	-1	-1	-1	-1	$+1$	$+1$	-1	-1	$+1$
Vγ6	$+1$	-1	-1	-1	-1	$+1$	$+1$	-1	-1	-1

5.5.2 gateTree Application

gateTree was applied to the live lymphocytes with the population descriptions given in Table 5.6. A two-stage approach was employed for gateTree in this application. gateTree was initially provided with a table consisting of the first nine columns from Table 5.6, then it was applied to the subset of events identified by the first stage with a table consisting only of the tenth column. This allowed gateTree to first identify a subset containing both the Vγ4 and Vγ6 $\gamma\delta$ T cells, and then to differentiate between these two populations using the tenth marker, Vγ4. This two-stage approach was necessary because of the high number of desired splits and the small size of the V$\gamma 4$ $\gamma\delta$ T-cell population. When we provided gateTree with all ten variables, it chose to split Vγ4 quite early, resulting in a Vγ4$+$ branch with a relatively small number of events. The small sample size on the Vγ4$+$ branch prevented gateTree from implementing all of the remaining splits. As a result, the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations would have differed in terms of more than just their Vγ4 expression levels, making biological interpretation of their differences challenging.

5.5.3 gateTree vs Manual Gating Comparison

Although our two-stage application of gateTree to the *S. aureus* data ensured that Vγ4 would be the final variable, gateTree was free to split the first nine variables in any order. It also could have refused to split some of these variables, if it was unable to find splits that satisfied its minimum requirements. Therefore, it is worth noting how similar the order in which gateTree split the variables was to the order in which they were used in the manual gating. This can be seen in Figure 5.6 and Table 5.7. Only one variable's rank changed significantly between the two variable orders. CD45 was the first variable used by

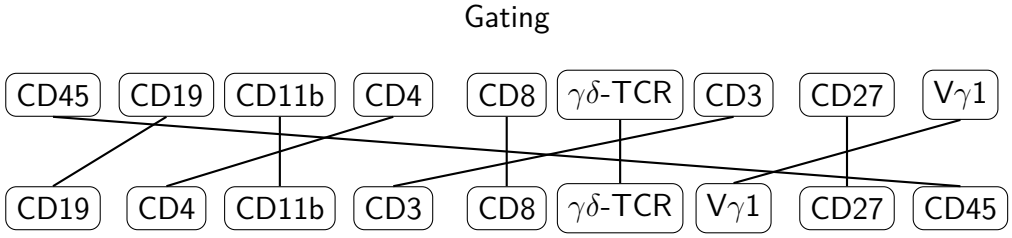


Figure 5.6: Visual comparison of the order in which the first nine channels were used by manual gating versus by gateTree for the *S. aureus* data.

Table 5.7: Order in which channels were used by manual gating versus by gateTree for the *S. aureus* data.

	Gating	gateTree	Difference
CD45	1	9	-8
CD19	2	1	$+1$
CD11b	3	3	0
CD4	4	2	$+2$
CD8	5	5	0
$\gamma\delta$-TCR	6	6	0
CD3	7	4	$+3$
CD27	8	8	0
Vγ1	9	7	$+2$
Vγ4	10	10	–

the manual gating and the ninth variable used by gateTree. However, four of the eight other variables had the same rank in both orders, and four had rank differences of between 1 and 3.

As well as using the variables in a similar order, Table 5.8 demonstrates that the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified by gateTree correspond well to those constructed via manual gating in terms of precision and recall. However, we can see in Table 5.8 that gateTree's Vγ4 $\gamma\delta$ T-cell populations for the Convalescent and Rechallenged groups contained fewer events on average than the manual gates, which caused low mean recall values. Figure 5.7 displays IL-17A vs Vγ4 scatter plots of the two populations identified by gateTree (odd rows) and manual gating (even rows) for each sample. As well as illustrating how similar the populations identified by gateTree were to the manually gated populations, Figure 5.7 also provides an insight into why gateTree's Vγ4 $\gamma\delta$ T cell recall

Table 5.8: Mean size, precision, and recall values for gateTree across samples within each experimental group from the *S. aureus* data.

Vγ4	Size (Gate Size)	Precision	Recall
Naïve	8 (7)	0.64	0.87
Acute	12 (7)	0.62	0.86
Convalescent	22 (47)	0.82	0.46
Rechallenged	38 (109)	0.89	0.34

Vγ6	Size (Gate Size)	Precision	Recall
Naïve	68 (73)	0.97	0.92
Acute	152 (155)	0.94	0.92
Convalescent	16381 (16167)	0.98	0.99
Rechallenged	22128 (22333)	0.96	0.96

values were slightly lower for the Convalescent and Rechallenged groups. For these two groups, we can see that the manual gating classified events with moderate Vγ4 expression levels as Vγ4 $\gamma\delta$ T cells, whereas gateTree classified some of these as Vγ6 $\gamma\delta$ T cells. Many of the events that were manually gated as Vγ4 $\gamma\delta$ T cells appear to be located in the upper tail of the Vγ6 $\gamma\delta$ T-cell population. Therefore, while these low recall values do highlight disagreement between gateTree's solution and the manual gating, we can defend gateTree's solution in this case. This example is a reminder that evaluating cell population identification algorithms solely by comparing their solutions to manual gates can give misleading results because the manual gating is itself subjective.

As an alternative demonstration of gateTree's performance for this data, we carried out an analysis of the differences between the experimental groups for the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations using the subsets of events identified via manual gating and those identified by gateTree to assess whether both population identification methods would result in a downstream statistical analysis reaching the same conclusions.

5.5.4 Differential Analysis

Our statistical analysis compared four pairs of experimental groups: Naïve vs Acute, Naïve vs Convalescent, Convalescent vs Rechallenged, and Acute vs Rechallenged. These pairs were chosen by the immunologists who carried out this experiment and they are the four

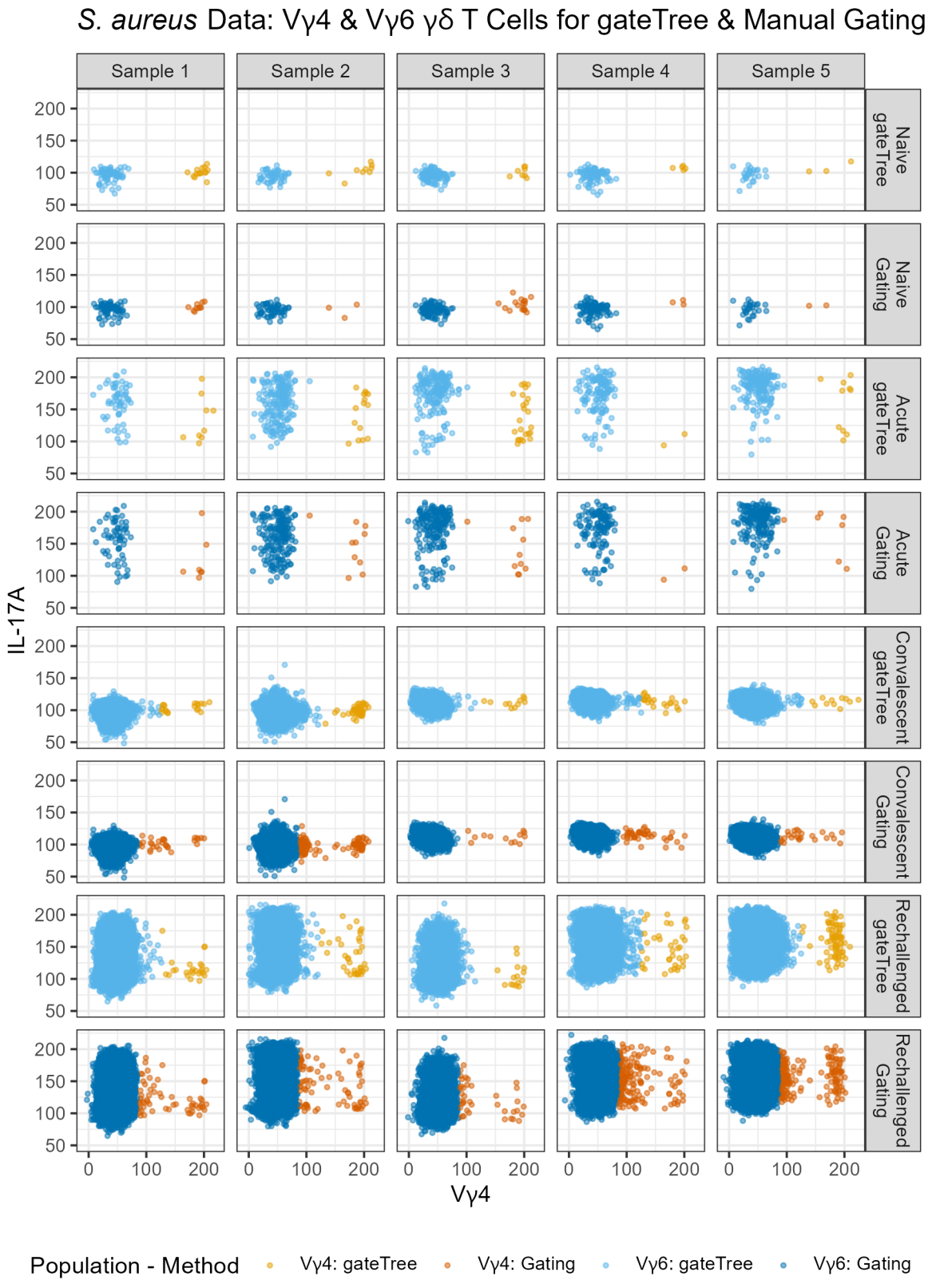


Figure 5.7: IL-17A vs Vγ4 scatter plots of the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified by gateTree and manual gating. It can be observed, particularly for the Convalescent and Rechallenged groups, that some cells with mid Vγ4 expression levels have been assigned to the Vγ4 $\gamma\delta$ T-cell population by the manual gating and to the Vγ6 $\gamma\delta$ T-cell population by gateTree.

pairs out of a possible six that differed in terms of recent exposure or convalescence period but not both. The excluded pairwise comparisons were Naïve vs Rechallenged and Acute vs Convalescent. For each pair of groups, we compared two quantities of interest, sample mean IL-17A value and sample proportion, for each of the two populations, Vγ4 and Vγ6 $\gamma\delta$ T cells. We conducted sixteen Wilcoxon rank-sum tests, otherwise known as Mann-Whitney U tests, to compare the distribution of the quantities of interest for each population across the chosen pairs of groups. The sixteen p-values were adjusted for multiple comparisons using the Benjamini-Hochberg method. This testing procedure was carried out once using the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified by gateTree, and once with the corresponding populations identified by manual gating.

Figure 5.8 displays each sample's population proportions and mean IL-17A expression levels for the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified by gateTree and manual gating. From this figure, we can observe strong differences between the sample mean IL-17A values for the Vγ6 $\gamma\delta$ T-cell population in each of the four experimental groups. In particular, the two groups that underwent a recent exposure prior to sample collection (Acute and Rechallenged) had higher sample mean IL-17A levels than the two groups without a recent exposure (Naïve and Convalescent). There is also a strong difference in Vγ6 $\gamma\delta$ T-cell proportion between the two groups that did not undergo a convalescence period (Naïve and Acute) and those that did (Convalescent and Rechallenged). For the Vγ4 $\gamma\delta$ T-cell comparisons, the group differences are less clear-cut. The small size of the Vγ4 $\gamma\delta$ T-cell population results in vastly different y-axis scales for the proportion values in this plot. The smaller size of the Vγ4 $\gamma\delta$ T-cell populations identified by gateTree compared to those constructed by the manual gating can also be seen in this plot. As a result of the small size of the Vγ4 $\gamma\delta$ T-cell populations relative to the Vγ6 $\gamma\delta$ T-cell populations, any group differences that do exist may also be less consequential. For example, most of the production of the IL-17A cytokine will be driven by the Vγ6 $\gamma\delta$ T-cell populations due to their greater abundance.

Figure 5.9 displays the thirty-two adjusted p-values arising from the two populations (Vγ4 and Vγ6 $\gamma\delta$ T cells), the two population identification methods (manual gating and

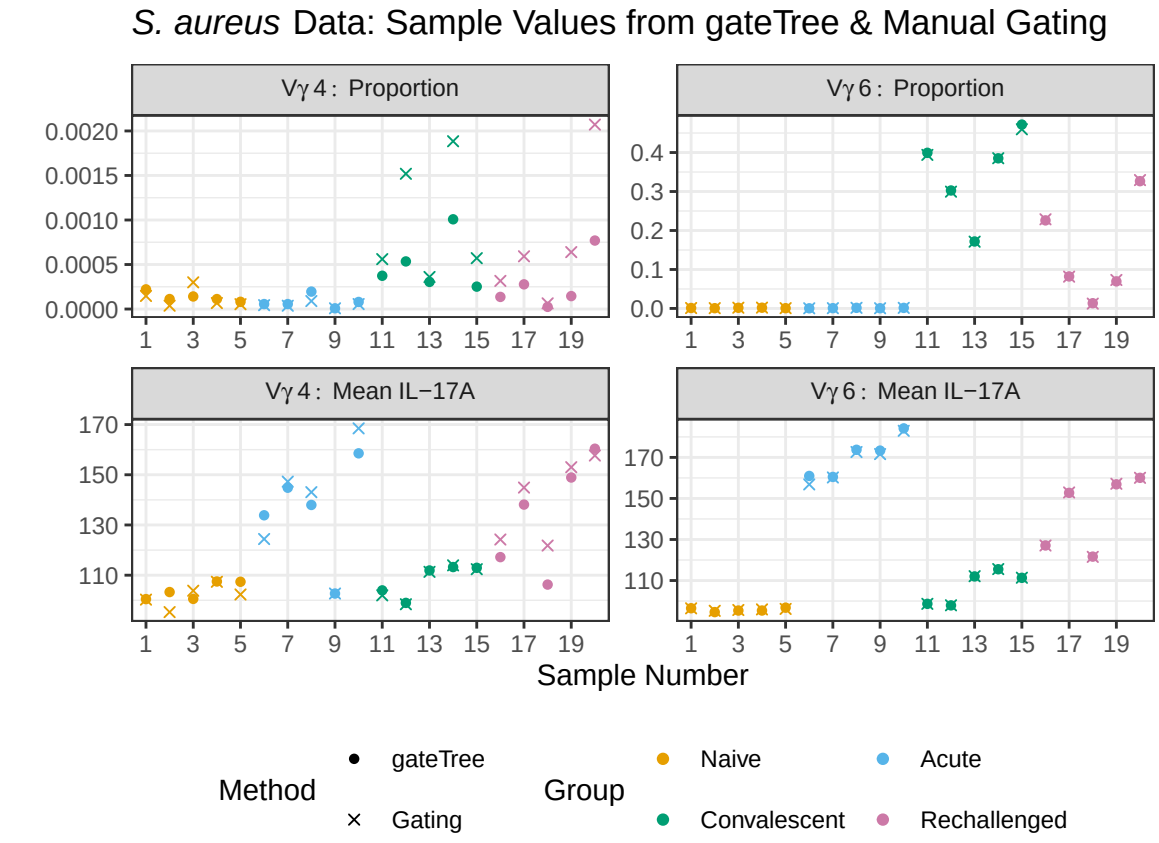


Figure 5.8: Scatter plots of the sample population proportions and sample mean IL-17A values for the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified via manual gating and by gateTree.

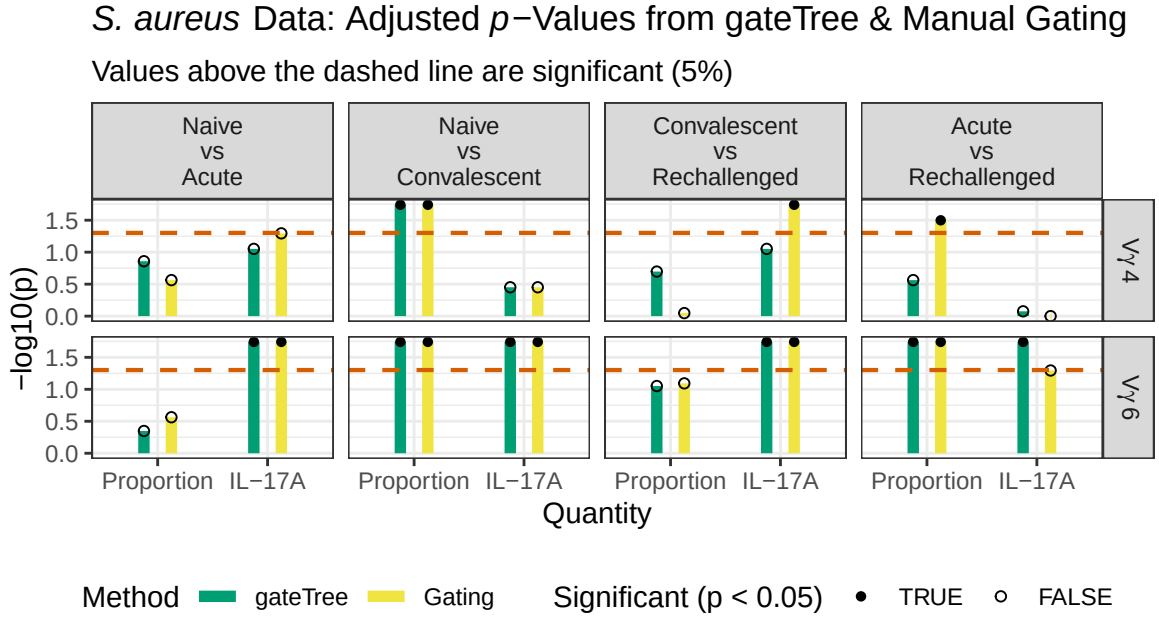


Figure 5.9: Column plots of the adjusted p-values obtained from pairwise Wilcoxon rank-sum tests using the Vγ4 and Vγ6 $\gamma\delta$ T-cell populations identified via manual gating and by gateTree. The negative base-10 logarithm of each adjusted p-value is plotted.

gateTree), the two quantities of interest (Mean IL-17A and Proportion), and the four pairwise comparisons. This figure illustrates that the p-values resulting from these pairwise Wilcoxon tests were often similar for the gateTree populations and the gating populations. Only three out of sixteen pairwise comparisons differed in terms of significance at the 5% level based on whether gateTree or manual gating were used to identify the populations.

The gateTree and gating populations led to the same conclusions when comparing the proportion of Vγ6 $\gamma\delta$ T cells in each sample. The two approaches found significant differences between the distributions of the Vγ6 $\gamma\delta$ T-cell proportions for both the Naïve vs Convalescent and Acute vs Rechallenged comparisons. Likewise, both approaches found no significant differences for the Naïve vs Acute and Convalescent vs Rechallenged proportion comparisons. When comparing the distributions of the sample mean IL-17A values, both the gateTree solution and the manual gating solution led us to conclude that the Vγ6 $\gamma\delta$ T-cell population differed significantly between each of the first three pairs of experimental groups being compared. For the Acute vs Rechallenged IL-17A comparison, the Vγ6 $\gamma\delta$ T-cell populations identified by gateTree did differ significantly at a 5% significance level in terms of the distribution of their sample mean IL-17A values. However, the manual gating solution did not find a significant difference for this comparison after adjusting for multiple comparisons.

Using the Vγ4 $\gamma\delta$ T-cell populations identified by gateTree, we did not find a significant difference between the sample mean IL-17A distributions for any of the four experimental group comparisons. However, the manual gating solution did find a significant difference when comparing the Convalescent group to the Rechallenged group. When we compared the distributions of the sample Vγ4 $\gamma\delta$ T-cell proportions, we found a significant difference between the Naïve and Convalescent groups using either solution. Additionally, the manual gating solution found a significant difference between the Acute and Rechallenged groups.

In summary, gateTree was able to successfully identify the targeted populations in these

data sets. A downstream statistical analysis using the gateTree identified populations aligned well with an identical analysis carried out using manually gated populations. Regardless of which population identification method was used, the two key takeaways of this analysis were unchanged. Firstly, the groups that underwent a convalescence period were statistically different in Vγ6 $\gamma\delta$ T-cell proportion to those that did not (Naïve vs Convalescent, Acute vs Rechallenged). Secondly, the groups that experienced a recent exposure prior to sample collection differed significantly in Vγ6 $\gamma\delta$ T-cell sample mean IL-17A levels from those that did not (Naïve vs Acute, Convalescent vs Rechallenged).

5.6 Conclusion

5.6.1 Discussion

gateTree offers biologists and immunologists the opportunity to exercise control over a population identification algorithm in a transparent and upfront manner, rather than tuning the parameters of an unsupervised clustering algorithm or post-processing its clustering solution. Moreover, it takes into account the user's biological knowledge in a straightforward manner that does not require them to manually gate the samples themselves. gateTree's recursive, univariate splitting approach results in an explainable clustering solution with clear biological interpretations regarding the positive and negative expression levels of the populations that it identifies. It also partitions every sample in an experiment according to the same tree structure, facilitating downstream comparison of the identified populations.

We have applied gateTree to simulated data, publicly available benchmark data, and to novel experimental data. In our first three applications, we carried out F_1 comparisons in which gateTree outperformed leading population identification algorithms. In our final application, we conducted identical experimental analyses based on manual gating and gateTree, and obtained similar outcomes from both sets of populations. In this example, gateTree successfully identified one rare and one extremely rare population across twenty

experimental samples. These applications demonstrate that gateTree can be an excellent alternative to manual gating, while offering practical advantages over leading unsupervised clustering algorithms.

5.6.2 Extensions

If a marker's distribution is trimodal rather than bimodal, then gateTree's density valley splitting mechanism will implement a single split at the deeper of the two valleys. That means it will classify events with intermediate expression levels as either positive or negative. Further work is required to automatically detect trimodal markers or allow the user to inform the algorithm that a marker is trimodal, and then to develop an approach to consistently distinguish between those three modes. The GMM boundary splitting mechanism could be easily extended by fitting a three-component model, whereas the density valley mechanism would need to be applied once to find the deepest valley, and then applied to the data on either side of that split to find two more valleys and identify which of those two is the second deepest valley overall.

Since gateTree only identifies the targeted populations which are described by the user, a potential area for future extensions of this approach would be to explore combining it with an exploratory analysis of the events which are left unassigned by gateTree. Currently, gateTree does not retain any of the branches that do not correspond to one of the targeted populations. Keeping these non-target branches would immediately provide a basic partition of the events which are currently left unassigned by gateTree. Furthermore, gateTree's splitting mechanisms could be applied to refine these branches in an unsupervised manner. This would be preferable to removing the gateTree populations from the data and applying an unsupervised clustering algorithm to the remaining events, which would result in a disjointed analysis and could cause unusual behaviour around the fringes of the removed populations.

Another possible extension of gateTree which we have been exploring is the implementation of two-dimensional splits. For bivariate straight-line splits, this would

involve searching over a discrete set of slopes by projecting the data onto a single dimension corresponding to each slope, finding the optimal univariate split for each projection, and choosing the slope-intercept pair that results in the best split.

6 Constrained Gaussian Mixtures for Flow Cytometry

6.1 Introduction

In Chapter 2, we examined the many adaptations that have been developed for model-based clustering to suit flow cytometry data and in Chapter 4 we discussed a handful of clustering methods that utilise information about the cell populations being targeted by their user. We also presented our own user-informed cell population method, gateTree, in Chapter 5. In this chapter, we explore a novel approach for integrating user information about cell populations into a mixture model via positive constraints (Melnykov et al., 2016; Shental et al., 2003). We apply our gateTree method to identify clusters matching the researcher's description of target populations, then use a constrained mixture model to keep the events at the core of each cluster together. While the Mondrian tree method (Ji et al., 2018) does implement a Bayesian model, our constrained mixture approach is designed to utilise user information while fitting a multivariate model via a modified Expectation-Maximisation (EM) algorithm and obtaining a soft, probabilistic component assignment for the unconstrained events.

We had two aims when designing this model. These were to improve a mixture model's ability to identify populations by utilising user-provided descriptions, and to extend a gateTree clustering solution to include additional clusters identified by a Gaussian mixture model. The latter is possible because the constrained mixture model can be fitted with a

higher number of mixture components than its number of constrained sets, leaving the extra mixture components free to explore the unconstrained events. We also hoped that through the multivariate modelling and probabilistic assignment of this approach, we could improve the clusters identified by gateTree.

In Section 6.3, we carried out two applications of constrained mixture models to the first sample of the Levine32 data mentioned in Sections 2.1.2 and 5.4.2. First, we evaluated their performance when identifying target populations for which constrained sets were provided. Then, we assessed their ability to identify an additional population without a constrained set, given constraints for other populations. In Section 6.4, we applied a constrained mixture model to a flow cytometry sample from the healthyFlowData R package and qualitatively compared its solution to those of gateTree and an unconstrained model.

6.2 Methodology

A positive constraint between a pair of data points ensures that they are assigned to the same cluster, while a negative constraint would instead ensure that they are assigned to different clusters (Shental et al., 2003). These are discussed in Bouveyron et al. (2019), where they refer to them as must-link and cannot-link constraints, respectively. Positive or must-link constraints can easily be extended to sets of events which are to be assigned to the same cluster.

We implemented a must-link constrained Gaussian mixture model with positively constrained sets of events arising from gateTree. Let $\mathcal{C}$ denote the collection of all constrained sets and let $\mathcal{C}_c$ denote constrained set c. The index of the constrained set containing event i, $\boldsymbol{x}_i$, is denoted by $c(i)$. It follows that $\mathcal{C}_{c(i)}$ denotes the constrained set containing event i. Following Melnykov et al. (2016), we defined the estimated posterior

probability, $\tilde{z}_{ig}^{(t)}$, at iteration t of assigning event i to component g, as follows,

$$\begin{aligned}\tilde{z}_{ig}^{(t)} :&= \mathbb{P}\left(z_{ig} = 1 \middle| \hat{\boldsymbol{\Psi}}^{(t-1)}, \boldsymbol{x}\right) \\ &= \frac{p\left(\{\boldsymbol{x}_i : i \in \mathcal{C}_{c(i)}\}, z_{ig} = 1 \middle| \hat{\boldsymbol{\Psi}}^{(t-1)}\right)}{p\left(\{\boldsymbol{x}_i : i \in \mathcal{C}_{c(i)}\} \middle| \hat{\boldsymbol{\Psi}}^{(t-1)}\right)} \\ &= \frac{\prod_{i \in \mathcal{C}_{c(i)}} \hat{\pi}_g^{(t-1)} f\left(\boldsymbol{x}_i \middle| \hat{\boldsymbol{\mu}}_g^{(t-1)}, \hat{\boldsymbol{\Sigma}}_g^{(t-1)}\right)}{\sum_{h=1}^{G} \prod_{i \in \mathcal{C}_{c(i)}} \hat{\pi}_h^{(t-1)} f\left(\boldsymbol{x}_i \middle| \hat{\boldsymbol{\mu}}_h^{(t-1)}, \hat{\boldsymbol{\Sigma}}_h^{(t-1)}\right)}.\end{aligned} \tag{6.1}$$

For an unconstrained model, the equivalent quantity, which we will denote by $\hat{z}_{ig}^{(t)}$, is defined as follows,

$$\hat{z}_{ig}^{(t)} = \frac{\hat{\pi}_g^{(t-1)} f\left(\boldsymbol{x}_i \middle| \hat{\boldsymbol{\mu}}_g^{(t-1)}, \hat{\boldsymbol{\Sigma}}_g^{(t-1)}\right)}{\sum_{h=1}^{G} \hat{\pi}_h^{(t-1)} f\left(\boldsymbol{x}_i \middle| \hat{\boldsymbol{\mu}}_h^{(t-1)}, \hat{\boldsymbol{\Sigma}}_h^{(t-1)}\right)}. \tag{6.2}$$

Notably, if event i is the only event in the constrained set $\mathcal{C}_{c(i)}$, then $\tilde{z}_{ig}^{(t)}$ is equal to $\hat{z}_{ig}^{(t)}$. Therefore, the must-link constrained Gaussian mixture model is a generalisation of a typical, unconstrained Gaussian mixture model, where the latter arises as a special case of the former when every event is in its own constrained set of size one. Moreover, any event j in the same constrained set as event i will have the same posterior assignment probability for every component. That is, if $c(i)$ is equal to $c(j)$, then $\tilde{z}_{ig}^{(t)}$ will be equal to $\tilde{z}_{jg}^{(t)}$ for every component g at each iteration t. This ensures that these events will have the same *maximum a posteriori* component assignment. The maximum likelihood estimators (MLEs) for the mean vectors, covariance matrices, and mixture proportions are identical to those for an unconstrained Gaussian mixture model, except they are calculated with respect to the modified $\tilde{z}$ values. That is, the MLE for the mean vector of component g, $\hat{\boldsymbol{\mu}}_g$, is

$$\hat{\boldsymbol{\mu}}_g = \frac{\sum_{i=1}^{n} \tilde{z}_{ig} \boldsymbol{x}_i}{\sum_{i=1}^{n} \tilde{z}_{ig}}, \tag{6.3}$$

the MLE for the covariance matrix of component g, $\hat{\boldsymbol{\Sigma}}_g$, is

$$\hat{\boldsymbol{\Sigma}}_g = \frac{\sum_{i=1}^{n} \tilde{z}_{ig} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_g)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_g)^T}{\sum_{i=1}^{n} \tilde{z}_{ig}}, \tag{6.4}$$

and the MLE for the mixture proportion of component g, $\hat{\pi}_g$, is

$$\hat{\pi}_g = \frac{\sum_{i=1}^{n} \tilde{z}_{ig}}{n}. \tag{6.5}$$

Our method begins by eliciting a population description table, then using gateTree to identify these target populations. For each cluster identified by gateTree, we compute the density of its events with respect to a multivariate Gaussian distribution, then select the subset of events whose Gaussian density exceeds a pre-specified percentile of the Gaussian densities. The selected events represent the core of the population identified by gateTree. We define must-link / positive constraints within these core subsets of events and fit a constrained mixture model to the data set. The result is a Gaussian mixture model where the core of each gateTree population cannot be split across multiple components.

An alternative approach for utilising gateTree labels to constrain a mixture model would be to fix the labels of these events prior to fitting a mixture model. In this case, the z_{ig} values of the constrained events are known and fixed *a priori* to be 0 or 1, instead of being estimated at each iteration. The mixture package in R facilitates this approach.

The main difference between these two approaches is whether multiple constrained sets are allowed to be assigned to the same mixture component. Must-link or positive constraints only ensure that the events within each constrained set are kept together, not that the events in different constrained sets are kept apart. To ensure that a maximum of one constrained set is assigned to each mixture component would require negative pairwise constraints, which are more computationally challenging to implement than their positive counterparts. The pre-labelling approach offers a simpler alternative which closely resembles utilising every possible pairwise positive constraint within each constrained set and every possible pairwise negative constraint between different constrained sets. Our initial work in this area employed must-link constraints, however, the pre-labelling approach may be more suitable for cell population identification, assuming that gateTree has correctly identified its target populations.

Initialising these models requires a non-standard approach. We have implemented a constrained version of the k-means algorithm. When fitting G mixture components given K constrained sets, we compute the means of the K constrained sets and keep these fixed while iteratively re-estimating another $G - K$ cluster centres. In other words, this initialisation algorithm has $G - K$ free clusters whose centres must be estimated and K constrained clusters whose centres are fixed. Only the unconstrained events are supplied to this algorithm. The algorithm alternates between assigning every unconstrained event to its nearest of the G cluster centres and computing the means of the unconstrained events assigned to one of the $G - K$ free clusters. These means are the new cluster centres of the $G - K$ free clusters, while the centres of the K constrained clusters are unchanged. After iteratively estimating the cluster assignment of the unconstrained events, the constrained events will be assigned to the cluster associated with their constrained set.

6.3 Levine32 Data Application

6.3.1 Identifying Target Populations

The first question we sought to answer in relation to this approach is 'Can constraints improve the ability of mixture models to identify target populations with known characteristics?'. In addition to this, given the strong performance of the gateTree algorithm, it is worth asking whether mixture models constrained using gateTree labels perform better than gateTree itself.

To investigate the performance of constrained mixture models, we applied gateTree to the first sample of the Levine32 data set and targeted the eight cell populations corresponding to manual gates containing more than 1% of events. There are four large gates containing more than 10% of the events and a further four medium-sized gates containing more than 1% of the events. We used the gateTree labels to constrain Gaussian mixture models both via must-link constraints and via pre-fixed labels. We applied both constrained models and an unconstrained model. We varied the number of mixture

components (G = 8, 9, 10, 12, 16, 22, 30, 40) fitted by each model. For the two constrained models, we also varied the proportion of the events identified by gateTree that we included in each constrained set (33%, 66%, and 99%).

Table 6.1 reveals that the performance of the two constrained models was almost indistinguishable across varying component numbers and constrained set sizes. While the constrained models did outperform the unconstrained model, we can observe that their mean F_1 increases as the constrained set size and the number of components increase. As the number of mixture components increases, the need for the components containing the constrained sets to grow and accommodate additional events diminishes. In other words, increasing the number of mixture components tends to remove more of gateTree's unassigned events from the mixture components containing the constrained sets. Meanwhile, increasing the constrained set size directly increases the number of events that were assigned by gateTree which are included in these components. Therefore, as the constrained set size and number of components increase, the components containing the constrained sets become more similar to the gateTree clusters. The highest mean F_1 score achieved by the constrained models was 0.76 for 40 mixture components and a constrained set size of 99%, which matches gateTree's mean F_1 score. These results suggest that if gateTree can be used to identify a target population, then the gateTree solution is unlikely to be improved further via the use of a constrained mixture model.

In conclusion, the use of constraints derived from a gateTree solution can improve the ability of a mixture model to identify target populations with known characteristics, however, in this situation, the gateTree solution itself is likely to perform as well or better than a constrained Gaussian mixture model.

6.3.2 Identifying Unknown Populations

The primary advantage of using a gateTree-constrained mixture model over using gateTree itself is the ability of the model to fit more components than the number of constrained sets, and, in doing so, potentially discover novel populations that were not

Table 6.1: Mean F_1 values across the eight target populations for gateTree, an unconstrained Gaussian mixture model, and two constrained GMMs. The number of mixture components and the size of the constrained sets were both varied.

Model	Number of Components							
(Constrained Set Size)	8	9	10	12	16	22	30	40
gateTree	0.76	-	-	-	-	-	-	
Must-Link (33%)	0.54	0.61	0.66	0.68	0.67	0.66	0.65	0.64
Must-Link (66%)	0.63	0.67	0.73	0.72	0.73	0.74	0.75	0.75
Must-Link (99%)	0.68	0.71	0.72	0.74	0.74	0.75	0.75	0.76
Pre-Labelled (33%)	0.54	0.61	0.66	0.68	0.67	0.66	0.65	0.64
Pre-Labelled (66%)	0.63	0.67	0.73	0.72	0.73	0.74	0.75	0.75
Pre-Labelled (99%)	0.68	0.71	0.72	0.74	0.74	0.75	0.75	0.76
Unconstrained	0.56	0.58	0.59	0.56	0.52	0.47	0.43	0.43

Table 6.2: Each model's best F_1 value for the four cell populations without constrained sets and the number of components used to achieve that solution.

	Gate 2	Gate 3	Gate 4	Gate 13	Mean
Must-Link	0.59 (10)	0.76 (9)	0.49 (12)	0.70 (10)	0.63
Pre-Labelled	0.59 (10)	0.76 (9)	0.49 (12)	0.70 (10)	0.63
Unconstrained	0.58 (40)	0.58 (40)	0.54 (22)	0.70 (12)	0.60

described in the table provided to gateTree by the user. This leads us to our second question regarding this approach: 'Can the use of constraints with respect to known populations improve the ability of mixture models to identify unknown populations?'. To evaluate the ability of the constrained models to identify unknown populations, we applied them repeatedly to the first Levine32 sample, leaving out one of the eight constrained sets each time. In particular, each of the four populations corresponding to a medium-sized gate containing between 1% and 10% of the events in the data were left out of the constraints one at a time. We then evaluated the ability of the constrained models to identify the excluded population given the other seven constrained sets. The full gateTree populations were used as constrained sets. That is, 100% of the events in each gateTree population were included in the corresponding constrained set.

Table 6.2 shows that on average the two constrained approaches marginally outperformed the unconstrained mixture model. This was driven by the improved performance of the constrained models for Gate 3 compared to the unconstrained model. Figure 6.1 displays

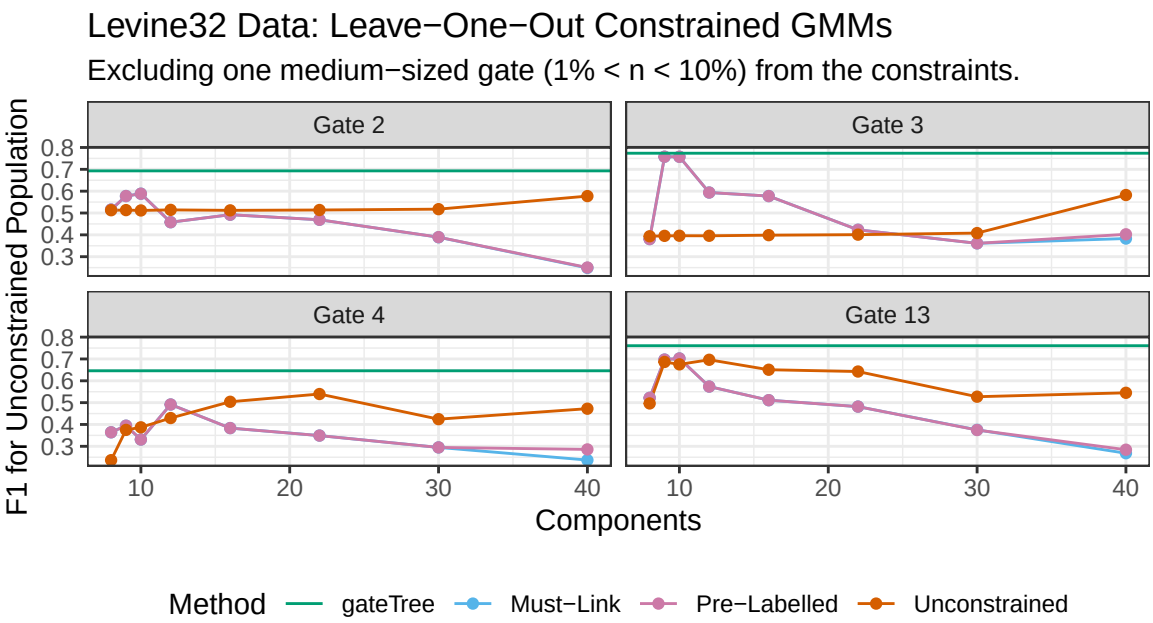


Figure 6.1: F_1 scores for constrained and unconstrained Gaussian mixture models for each medium-sized gate when the constrained set corresponding to that cell population is excluded. The green horizontal line displays the F_1 value achieved by gateTree for this population as a benchmark, however, none of the three models being evaluated were given information about the desired population, unlike gateTree.

the F_1 values for each population across a range of component numbers. It is worth highlighting that the constrained models' solutions will be consistent with the gateTree solution, whereas an unconstrained model is likely to break up the gateTree populations across multiple mixture components.

One limitation of this application is that we have evaluated each model with respect to a single manual gate, despite the fact that our constrained approach is designed to explore the events left unassigned by gateTree. In the next section, we will extend a gateTree solution using a constrained model as part of an exploratory analysis, rather than a quantitative evaluation.

6.4 healthyFlowData Application

The healthyFlowData package in R (Azad, 2013) contains twenty flow cytometry samples from four healthy human subjects. There are five samples of peripheral blood mononuclear cells from each of the four subjects, analysed with respect to the CD19,

Table 6.3: The cell population descriptions used to inform gateTree when applying it to healthyFlowData Sample A1.

	CD19	CD3	CD4	CD8
B Cells	$+1$	-1	0	0
CD4+ T Cells	0	$+1$	$+1$	-1
CD8+ T Cells	0	$+1$	-1	$+1$

CD3, CD4, and CD8 protein markers. We displayed one sample from each of the four subjects in Figures 2.3, 2.4, and 2.5 in Section 2.4. In the healthyFlowData samples, there are three prominent cell populations which can be described as CD3$-$CD19+, CD3+CD19$-$CD4+CD8$-$, and CD3+CD19$-$CD4$-$CD8+. We can associate these populations with B Cells, CD4+ T Cells, and CD8+ T Cells, respectively. There is also a visible population of CD3$-$CD19$-$ events.

The objective of this application is to visually examine and qualitatively assess the performance of a gateTree-constrained Gaussian mixture model on a flow cytometry data set, when fitted with a greater number of mixture components than constrained sets.

We applied gateTree to Sample A1, the first sample from Subject A, informed by the population descriptions in Table 6.3, then constrained a six-component Gaussian mixture model using the three gateTree populations. In this application, we have implemented the 'pre-labelling' approach, rather than positive pairwise constraints. Using more mixture components than constrained sets enables us to investigate the events left unassigned by gateTree, in a manner that is consistent with the populations constructed by gateTree. We also fitted an unconstrained six-component GMM to this data set for comparison purposes. Figures 6.2, 6.4, and 6.3 display generalised pairs plots of healthyFlowData Sample A1 for gateTree, the unconstrained GMM, and the constrained GMM, respectively. Tables 6.4 and 6.5 compare the component assignments from the unconstrained and constrained GMMs to gateTree's population identification.

In Figure 6.3, the purple component of the constrained GMM ($g = 5$) identifies a small CD3+CD4$-$CD8$-$ population that was not visible prior to clustering. It also constructs

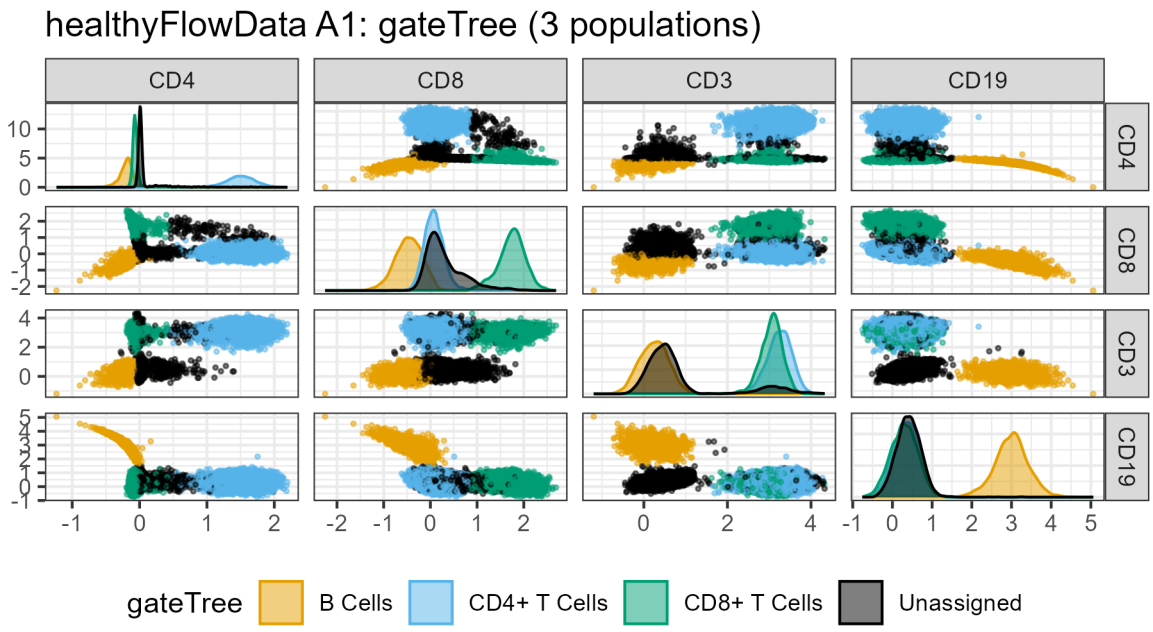


Figure 6.2: Generalised pairs plot of healthyFlowData Sample A1 coloured according to gateTree.

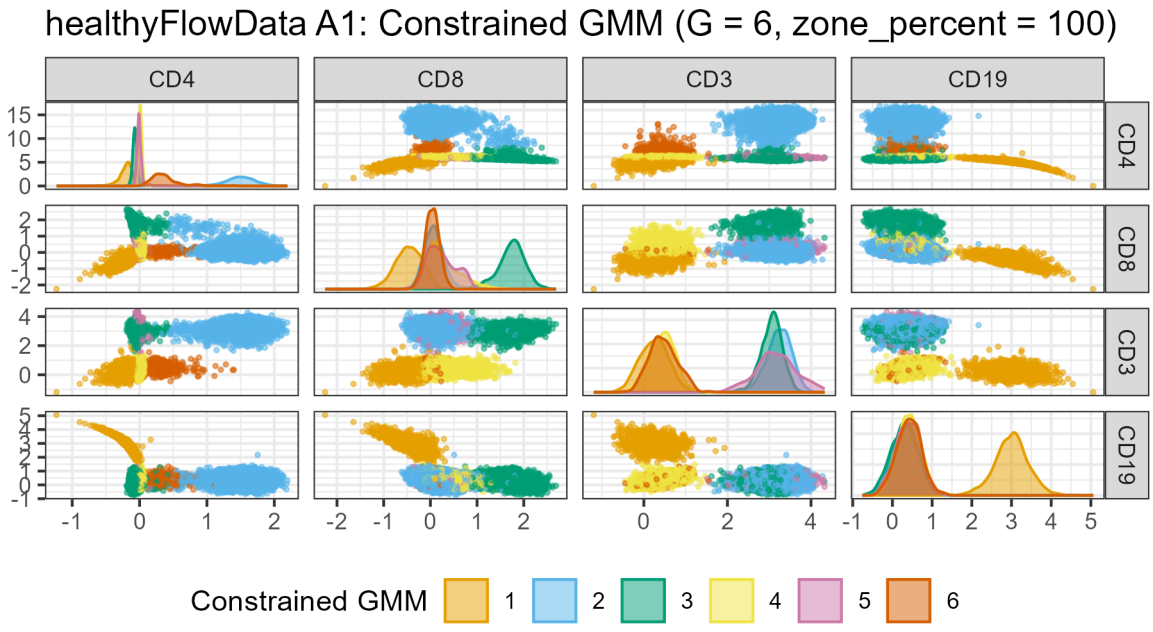


Figure 6.3: Generalised pairs plot of healthyFlowData Sample A1 coloured according to a six-component constrained GMM with three pre-labelled constrained sets from gateTree. Every event from the gateTree populations was included in the constrained sets (`zone_percent` = 100).

Table 6.4: Contingency table comparing the unconstrained six-component GMM solution to the gateTree solution.

	Unconstrained GMM						
gateTree	1	2	3	4	5	6	Total
B Cells	1658	0	0	0	0	0	1658
CD4+ T Cells	0	10,221	0	0	0	85	10,306
CD8+ T Cells	0	0	3862	0	192	99	4153
Unassigned	4	6	0	2708	329	157	3204
Total	1662	10,227	3862	2708	521	341	19,321

Table 6.5: Contingency table comparing the constrained six-component GMM solution to the gateTree solution.

	Constrained GMM						
gateTree	1	2	3	4	5	6	Total
B Cells	1658	0	0	0	0	0	1658
CD4+ T Cells	0	10,306	0	0	0	0	10,306
CD8+ T Cells	0	0	4153	0	0	0	4153
Unassigned	4	151	15	2434	318	282	3204
Total	1662	10,457	4168	2434	318	282	19,321

two CD3−CD19− clusters (yellow, $g = 4$, and dark orange, $g = 6$) which differ noticeably with respect to the CD4 marker. The density plots on the diagonal of Figure 6.3 show that the constrained GMM's three unconstrained components are reasonably unimodal and seem to consist solely of positive or negatively expressed events for each marker. Importantly, the constrained GMM integrates the target populations identified by gateTree into its solution, allowing them to expand to accommodate additional events but assigning all of their events to the same component. This can be seen in Table 6.5, which conveys that components 4 (yellow), 5 (purple), and 6 (dark orange) of the constrained GMM were not assigned any of the events from gateTree's three target populations.

There are also notable differences between the constrained and unconstrained GMM solutions. From Table 6.4, we can see that component 6 (dark orange) of the unconstrained GMM contained 85 events identified by gateTree as CD4+ T Cells as well as 99 events identified by gateTree as CD8+ T Cells. The bimodality of this mixture component ($g = 6$, dark orange) is evident in the CD8 density plot from Figure 6.4. The

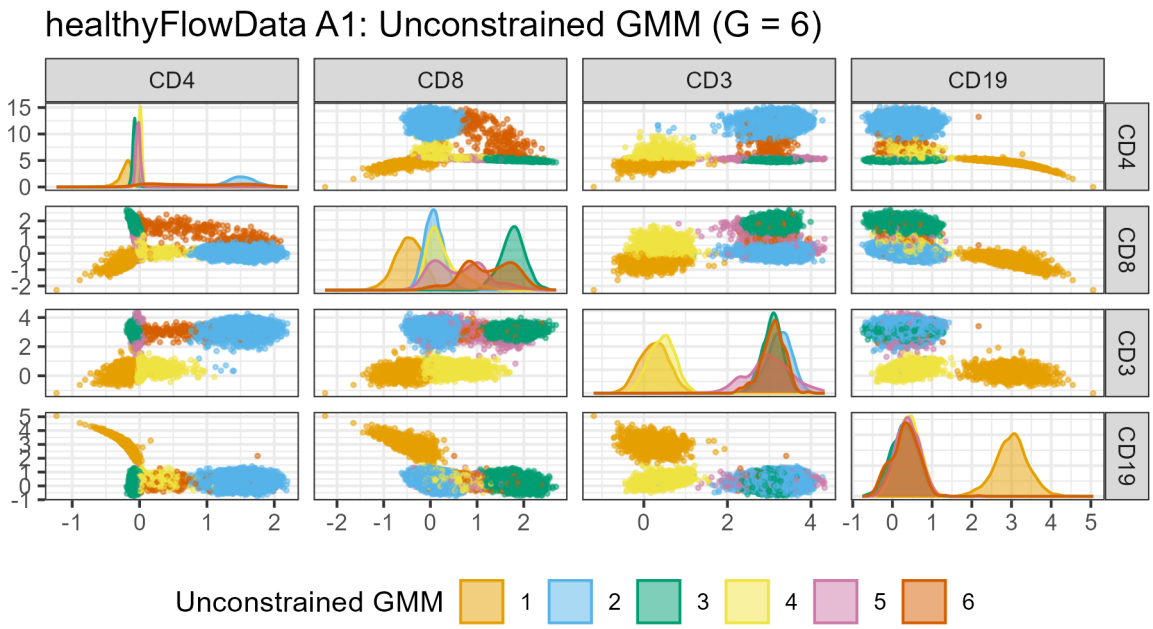


Figure 6.4: Generalised pairs plot of healthyFlowData Sample A1 coloured according to a six-component unconstrained GMM.

purple component ($g = 5$) also exhibits some bimodality with respect to CD8, and contains 192 of the events identified by gateTree as CD8+ T Cells.

These differences demonstrate that the constraints did have an effect on the clustering solution and highlight the usefulness of this approach for extending a user-informed gateTree solution to include additional unknown populations.

6.5 Conclusion

Constrained mixture models are an interesting and potentially under-explored area within model-based clustering. We demonstrated in Section 6.3.1 that they can outperform unconstrained mixture models when identifying target populations with known characteristics, although they did not outperform our gateTree method. In Section 6.3.2, the gateTree constraints did not strongly improve the mixture models' ability to identify manually gated populations that had not been targeted by gateTree. However, as part of an exploratory analysis in Section 6.4, the constrained model produced a more biologically meaningful clustering solution by maintaining the populations identified by gateTree and

constructing additional clusters that were more clearly defined with respect to the data's markers.

One limitation of our constrained mixture model application in this chapter is that it does not directly address the challenges associated with fitting mixture models to cytometry data that we discussed in Chapter 2, since it is based on a generic Gaussian mixture model. Ideally, a model-based clustering algorithm for cytometry data would combine a selection of the adaptations discussed in Chapter 2. Therefore, perhaps the constrained approaches that we have discussed in this chapter could be more beneficial if applied in conjunction with other mixture model adaptations for cytometry data. For example, in another setting, that of hyperspectral imaging, Babu et al. (2025) implemented pairwise constraints for parsimonious Gaussian mixture models. In this chapter, the main focus has been on our user-informed construction of the constraints via our gateTree algorithm. Future work could extend this approach to more sophisticated model-based clustering methods, such as skew-t mixtures.

7 Sequential Outlier Identification for Gaussian Mixture Models

7.1 Introduction

In this chapter, we will deviate from focusing exclusively on cytometry data to present a novel method which we developed for outlier detection in Gaussian mixture modelling. This work arose from a research visit to McMaster University in Ontario, Canada and was carried out in collaboration with Prof. Paul McNicholas. Although this method was not developed for population identification, outliers are a problem in cytometry data, and this was part of our motivation for researching this area.

The presence of outliers can prevent clustering algorithms from accurately determining an appropriate group structure within a data set. This can be particularly challenging in the case of model-based clustering (McNicholas, 2016; Bouveyron et al., 2019), where departures from model assumptions can unduly influence parameter estimation and the cluster allocation of observations. In such cases, small numbers of observations can dominate the cluster analysis, leading to an unsatisfactory analysis in which much of the underlying cluster structure of interest is missed.

We propose an iterative method for sequentially identifying outliers in a model-based clustering framework, where we assume that the mixture components are Gaussian distributions. outlierMBC identifies and removes outliers from a data set without assuming the probability distribution of these outliers or requiring the number of outliers to be

pre-specified. Potential outliers are removed one at a time and a dissimilarity is computed at each iteration. This dissimilarity involves the distribution of sample Mahalanobis distances. outlierMBC presents a graph of these dissimilarity values plotted against the number of removed observations. This dissimilarity curve is used to select the optimal number of outliers and allows the user to compare the minimum dissimilarity solution with our alternative 'backtrack' solution, or choose their own number of outliers based on the graph. outlierMBC also provides the Gaussian mixture model associated with the chosen solution, fitted to the remaining observations after the outliers have been removed. Removing outliers one by one allows outlierMBC to methodically search for the optimal number of outliers and provides an intuitive, interpretable sequence of solutions.

Figure 7.1 illustrates that removing outliers from a data set can improve the fit of a Gaussian mixture model. It displays the Swiss Banknote data set, which will be discussed in more detail in Section 7.7, clustered by a pair of two-component Gaussian mixture models, one fitted by a standard algorithm, and one fitted by outlierMBC, which identified and removed twenty outliers. It can be observed that the second component of the standard model does not fit the counterfeit banknotes well. However, the model fits the remaining counterfeit banknotes better after outliers are removed. Another approach for this data set would be to fit a Gaussian mixture model with more than two components rather than removing the outliers, but this would not fully resolve the poor model fit.

Our method is implemented in R as a package called outlierMBC, which is published on CRAN, the Comprehensive R Archive Network.

7.2 Outliers in Model-Based Clustering

Several different approaches have been developed for dealing with outliers in model-based clustering. Rather than explicitly differentiating between outliers and regular observations, some methods aim to accommodate points that are ill-fitting with respect to a Gaussian mixture model by replacing the Gaussian distribution with more heavy-tailed distributions such as in a mixture of t distributions (Peel and McLachlan, 2000; Evans et al., 2015) or

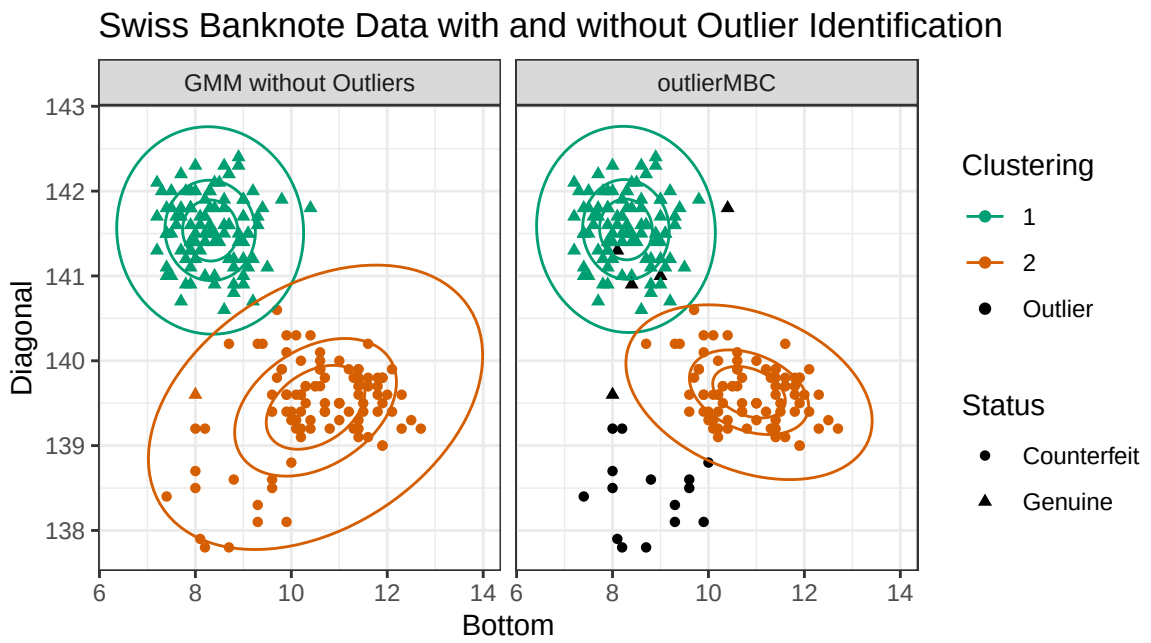


Figure 7.1: A pair of scatter plots for two variables from the six-dimensional Swiss Banknote data set with points coloured according to clustering solutions from two-component Gaussian mixture models. The two plots compare the solution without identifying outliers (left) to the outlierMBC solution (right) which identifies twenty outliers. Density level sets, in the form of ellipses, are displayed for each component.

a mixture of power exponential distributions (e.g., Dang et al., 2015). Other methods use an additional component in the mixture to accommodate outliers; this could be a uniform component (Fraley and Raftery, 2002) or it could have an improper constant density (Coretto and Hennig, 2016).

Contaminated mixture models fit a pair of nested distributions for each component, assigning points further from the component mean, such as outliers, to the outer distribution (Tukey, 1959; Punzo and McNicholas, 2016). Another approach is to remove the outliers instead of assuming that they follow a known distribution. García-Escudero et al. (2008) introduced a method called TCLUST for trimming a pre-specified proportion of the observations with the lowest mixture density from each iteration of its model fitting algorithm.

There are also two important algorithmic approaches — called OCLUST (Clark and McNicholas, 2024) and OTRIMLE (Coretto and Hennig, 2016), respectively — that use distributional results to formulate algorithms for outlier detection. To facilitate a

comparison with outlierMBC, which also uses distributional results, these methods are discussed in Section 7.3.2.

7.3 Methodology

7.3.1 outlierMBC

In this section, we outline the key details of the outlierMBC procedure. While our method operates within a mixture modelling framework, for ease of exposition we first outline the approach for a single multivariate Gaussian distribution. Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be distributed according to a single multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. That is,

$$\boldsymbol{X}_i \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \text{ for } i = 1, \ldots, n. \tag{7.1}$$

Let $\bar{\boldsymbol{X}}$ and $\boldsymbol{S}$ be the sample mean vector and sample covariance matrix of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$. Define a scaled squared sample Mahalanobis distance, U_i, for each observation, $\boldsymbol{X}_i$, as follows,

$$U_i := \frac{n}{(n-1)^2} \left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right)^T \boldsymbol{S}^{-1} \left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right) \text{ for } i = 1, \ldots, n. \tag{7.2}$$

The distances $U_1, \ldots, U_n$ are beta-distributed (Gnanadesikan and Kettenring, 1972; Ververidis and Kotropoulos, 2008; see Appendix A1 for more details),

$$U_i \sim \text{Beta}\left(\frac{p}{2}, \frac{n-p-1}{2}\right). \tag{7.3}$$

In the presence of outliers, the distribution of U_i will depart from its reference beta distribution. outlierMBC fits a multivariate Gaussian model to the data, and then computes a dissimilarity between the observed and theoretical distributions of the scaled squared sample Mahalanobis distances. The lowest density observation is then removed. This process is repeated until a pre-specified maximum number of observations have been removed. The optimal number of outliers, o, can then be determined based on the

computed values of the dissimilarity measure. We propose three approaches for determining o: 1) the value that minimises the observed-theoretical dissimilarity measure; 2) a more conservative 'backtrack' solution, to be used under mild departures from our model assumptions (see Section 7.3.5 for further details); 3) a subjective, user-defined decision based on the graph of the dissimilarity values versus the number of removed observations.

We extend this approach to a Gaussian mixture model with mixture density,

$$f(\boldsymbol{x}_i) = \sum_{g=1}^{G} \pi_g \phi(\boldsymbol{x}_i | \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g), \tag{7.4}$$

and component density,

$$\phi(\boldsymbol{x}_i | \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g) = \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_g|}} \exp\left\{ -\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_g)^T \boldsymbol{\Sigma}_g^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_g) \right\}. \tag{7.5}$$

Such models can be fitted to data using the Expectation-Maximisation (EM) algorithm (Dempster et al., 1977). In principle, outlierMBC is agnostic to the model fitting process, however, it does utilise the mixture R package (Pocuca et al., 2024) to estimate model parameters via the EM algorithm. In the mixture model setting, we compute a dissimilarity for each mixture component and aggregate these component-specific dissimilarities to obtain a single aggregated dissimilarity for each potential set of outliers. We then choose the optimal number of outliers based on this aggregated dissimilarity. Full details of the outlierMBC method are given in Algorithm 4.

7.3.2 Related Methods

outlierMBC and OCLUST (Clark and McNicholas, 2024) both remove potential outliers one by one until a user-specified maximum number of outliers have been removed and then retrospectively estimate the optimal number of outliers. Moreover, both methods rely on the fact that scaled squared sample Mahalanobis distances are beta-distributed (Ververidis and Kotropoulos, 2008; Wilks, 1963), but exploit this result in different ways.

OCLUST extends the result to show that, under certain assumptions, scaled and shifted subset log-likelihood differences are approximately beta-distributed. OCLUST then compares the theoretical and empirical distributions of the scaled and shifted subset log-likelihood differences as potential outliers are removed, in order to determine the optimal number of outliers in the data set. Each subset log-likelihood difference is obtained by fitting a mixture model to the data with and without a given observation. Therefore, if there are n observations, computing n subset log-likelihood differences requires the fitting of $n + 1$ Gaussian mixture models. In contrast, outlierMBC compares the theoretical and empirical distributions of the scaled squared sample Mahalanobis distances themselves. Since the computational cost of computing sample Mahalanobis distances is much lower than the cost of computing subset log-likelihood differences, the run time of outlierMBC is shorter than that of OCLUST. Another benefit of our approach is that Mahalanobis distances are a more familiar and intuitive notion than subset log-likelihood differences.

OTRIMLE also uses the distribution of Mahalanobis distances as part of a model-based clustering and outlier identification algorithm. However, there are two main differences between OTRIMLE and outlierMBC. Firstly, OTRIMLE employs a chi-squared distribution rather than a beta distribution. These are the theoretical distributions of the squared true Mahalanobis distances and of the scaled squared sample Mahalanobis distances, respectively. The true Mahalanobis distances are computed with respect to the unknown true component mean vectors and covariance matrices, whereas the sample Mahalanobis distances are computed with respect to the sample component mean vectors and covariance matrices. Secondly, OTRIMLE compares the theoretical and observed distributions of Mahalanobis distances to decide the improper density level of a noise component and then fits a Gaussian mixture model with this additional outlier component. outlierMBC uses this comparison to directly choose the number of outliers.

7.3.3 Sample Mahalanobis Distance Dissimilarity

Given a p-dimensional sample from a Gaussian mixture model with G components of sizes $n_1, \ldots, n_G$, the theoretical distribution of the scaled squared sample Mahalanobis distances from component g is a beta distribution with parameters $\frac{p}{2}$ and $\frac{n_g - p - 1}{2}$. Using the estimated component sizes from our fitted model, $\hat{n}_1, \ldots, \hat{n}_G$, we can approximate this distribution. We can also compute the empirical distribution of the observed scaled squared sample Mahalanobis distances with respect to our fitted model parameters for each component.

For each component, we evaluate the theoretical and empirical cumulative distribution functions (CDFs) for the scaled squared sample Mahalanobis distances at regular intervals and compute the mean of the absolute differences between their values. This may be seen as a sample approximation of the Wasserstein distance (Panaretos and Zemel, 2019),

$$W_1(X, Y) = \int_{\mathbb{R}} |F_X(t) - F_Y(t)| dt. \tag{7.6}$$

We aggregate these component-specific dissimilarity values by computing a weighted quadratic mean with respect to the component mixture proportions. Our primary choice for the optimal number of outliers is the value which minimises this aggregated dissimilarity.

7.3.4 Gross Outliers

As a pre-processing step, we identify and remove the most obvious outliers, which we call gross outliers following the terminology in Clark and McNicholas (2024). Our gross outlier identification procedure first computes the distance from each observation to its k^{th} nearest neighbour. By default, k is set to 1% of the sample size ($k = \lfloor 0.01 \times n \rfloor$). We require the user to specify the maximum total number of outliers, M. We identify the observations with the M largest k^{th} nearest neighbour distances as potential outliers. We select the next largest k^{th} nearest neighbour distance, outside of these potential outliers, as a reference value. We classify any observation with k^{th} nearest neighbour distance

greater than three times the reference value to be a gross outlier.

7.3.5 backtrack Mechanism

In some situations, particularly when the data does not fully satisfy the assumptions of the Gaussian mixture model, the solution which exactly minimises the sample Mahalanobis distance dissimilarity may remove more observations than desired. For these cases, we have developed a mechanism which allows outlierMBC to select a more conservative number of outliers by moving backwards from the minimum. Each backward step reduces the chosen number of outliers by one and must pass two validity checks in order to be allowed. These checks are defined based on two user-defined thresholds, namely, the maximum step rise and the maximum total rise, which we will refer to as α and β, respectively. Firstly, the increase in the dissimilarity as a result of a single step, as a proportion of the minimum dissimilarity, must be less than α. Secondly, the cumulative increase from all steps cannot exceed β, as a proportion of the minimum dissimilarity. Both of these parameters can be specified by the user, however the defaults are $\alpha = 0.05$ and $\beta = 0.10$, meaning that the dissimilarity cannot increase by more than 5% of the minimum in any one step and cannot increase by more than 10% of the minimum in total. We will refer to the solution obtained by choosing the exact minimum as outlierMBC-minimum and the solution which uses the backtrack choice as outlierMBC-backtrack.

Similarly, OTRIMLE has a procedure which allows it to accept a choice of its improper density value which is sub-optimal with respect to minimising its dissimilarity but results in a lower number of outliers being removed. It does so by minimising a penalised version of the dissimilarity. The penalty term consists of the proportion of outliers multiplied by a user-defined quantity β.

7.3.6 Initialisation

outlierMBC uses a hierarchical clustering approach to construct an initial cluster solution to initialise the first Gaussian mixture model. However, since outlierMBC fits a large

Algorithm 4 outlierMBC

Data: $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subset \mathbb{R}^p$; Indices: $\mathcal{I} = \{1, \ldots, n\}$; Outliers: $\mathcal{O} = \{\}$.
Maximum Number of Outliers: M; Number of Components: G.
for $m = 0$ to M **do**
Let $\sigma : \{1, \ldots, n-m\} \to \{1, \ldots, n\}$ map $\mathcal{X} \setminus \mathcal{O}$ indices to $\mathcal{X}$ indices.
Fit a Gaussian mixture model to $\mathcal{X} \setminus \mathcal{O}$, estimating the component assignment probability matrix, $\hat{Z}$, and the parameters, $\hat{\boldsymbol{\Psi}} = \{\hat{\pi}_g, \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Sigma}}_g\}_{g=1}^G$.
for $g = 1$ to G **do**
Compute the estimated number of data points in component g, $\hat{n}_g$, and then compute the sample covariance matrix of component g, $\hat{\boldsymbol{S}}_g$,

$$\hat{n}_g \leftarrow \sum_{j=1}^{n-m} \hat{Z}_{jg}, \qquad \hat{\boldsymbol{S}}_g \leftarrow \frac{\hat{n}_g}{\hat{n}_g - 1} \hat{\boldsymbol{\Sigma}}_g.$$

Compute the squared sample Mahalanobis distances, $\hat{d}_g(\boldsymbol{x}_{\sigma(j)})^2$, and then rescale them to obtain the scaled squared sample Mahalanobis distance, $\hat{u}_{jg}$,

$$\hat{d}_g(\boldsymbol{x}_{\sigma(j)})^2 = (\boldsymbol{x}_{\sigma(j)} - \hat{\boldsymbol{\mu}}_g)^T \hat{\boldsymbol{S}}_g^{-1} (\boldsymbol{x}_{\sigma(j)} - \hat{\boldsymbol{\mu}}_g), \qquad \hat{u}_{jg} \leftarrow \frac{\hat{n}_g}{(\hat{n}_g - 1)^2} \hat{d}_g(\boldsymbol{x}_{\sigma(j)})^2.$$

Let $\hat{F}_g$ be the empirical CDF of $\{\hat{u}_{jg}\}_{j=1}^{n-m}$ weighted by $\left\{\frac{\hat{Z}_{jg}}{\hat{n}_g}\right\}_{j=1}^{n-m}$, and let F_g be the CDF of a beta distribution with parameters $\frac{p}{2}$ and $\frac{\hat{n}_g - p - 1}{2}$.
Compute the mean of the absolute CDF differences ($T = 10,000$),

$$\hat{D}_{mg} \leftarrow \frac{1}{T} \sum_{t=1}^{T} \left| F_g\left(\frac{t}{T}\right) - \hat{F}_g\left(\frac{t}{T}\right) \right|$$

end for
Aggregate the component-wise dissimilarities, $\hat{D}_{mg}$: $\hat{D}_m \leftarrow \sqrt{\sum_{g=1}^G \hat{\pi}_g (\hat{D}_{mg})^2}$.
Identify the index, out(m), of the lowest mixture density data point, then add $\boldsymbol{x}_{\text{out}(m)}$ to the set of outliers and remove out(m) from the current set of indices,

$$\text{out}(m) \leftarrow \underset{i \in \mathcal{I}}{\arg\min} \left\{ p\left(\boldsymbol{x}_i \middle| \hat{\boldsymbol{\Psi}}\right) \right\}, \qquad \mathcal{O} \leftarrow \mathcal{O} \cup \{\boldsymbol{x}_{\text{out}(m)}\}, \qquad \mathcal{I} \leftarrow \mathcal{I} \setminus \{\text{out}(m)\}.$$

end for
Identify the number of removed points, o, at which the aggregated dissimilarity was minimised, and then classify the first o points removed as outliers,

$$o \leftarrow \underset{m \in \{0, \ldots, M\}}{\arg\min} \left\{ \hat{D}_m \right\}, \qquad \mathcal{O} \leftarrow \{\boldsymbol{x}_{\text{out}(1)}, \boldsymbol{x}_{\text{out}(2)}, \ldots, \boldsymbol{x}_{\text{out}(o)}\}.$$

Fit a G-component Gaussian mixture model to $\mathcal{X} \setminus \mathcal{O}$.

sequence of Gaussian mixture models, we designed two different procedures for how to initialise the remaining models in the sequence. Our first initialisation scheme, which we will refer to as the 'update' scheme, takes the posterior component assignment probability matrix from the fitted previous model, removes the row corresponding to the newly proposed outlier, and then uses this submatrix to initialise the next mixture model. Our second initialisation scheme, which we will refer to as 'reinit', implements hierarchical clustering after every outlier removal and provides a brand new reinitialisation to the next model. The 'update' scheme is usually much quicker and for many data sets, the extra reinitialisation can be redundant. However, the 'reinit' scheme can be more suitable for particularly challenging data sets where there is no clear cluster structure.

7.4 Illustrative Examples

To demonstrate how the method works in relatively straightforward conditions, we applied it to small and large data sets simulated from the same three-component mixture model. The mean vectors, covariance matrices, and mixture proportions of the three components are the same for both data sets. The data sets consist of 1000 or 4000 Gaussian observations and 10 or 40 outliers. The true observations were sampled from a mixture of Gaussian distributions and the outliers were sampled from a uniform distribution across a hyper-rectangle containing the true observations excluding an ellipsoid around each Gaussian distribution. These data sets are shown in Figure 7.2 with observations coloured according to their outlierMBC clustering. outlierMBC-minimum and outlierMBC-backtrack both achieved perfect classification for the small data set while for the large data set, they had two false positives and one false negative, respectively. The dissimilarity curves for both data sets are smooth and stable, as can be seen in Figure 7.3.

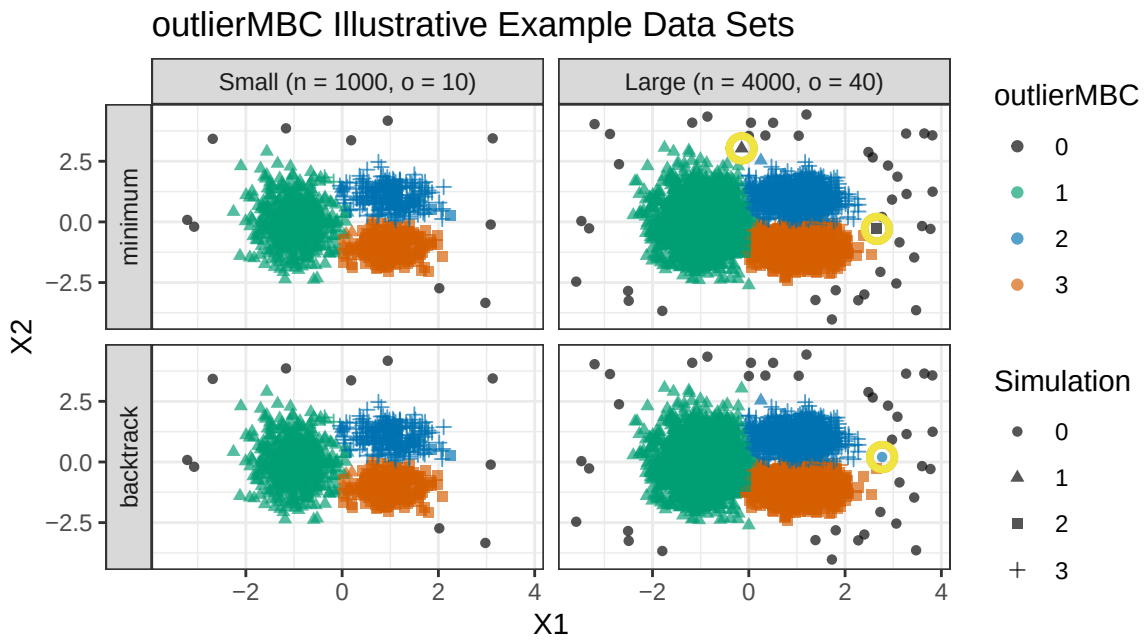


Figure 7.2: Scatter plots of the illustrative data sets with their points coloured according to the outlierMBC solutions and with outlier classification errors circled in yellow.

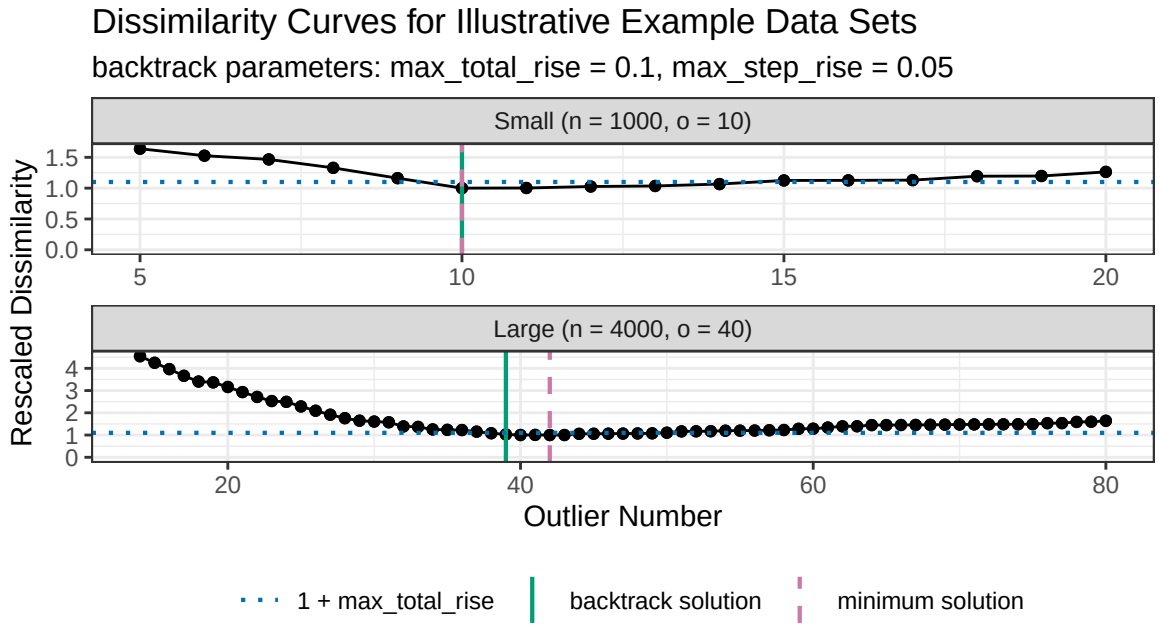


Figure 7.3: Dissimilarity curves for the two illustrative data sets. The dissimilarity values on the y-axis have been rescaled so that the minimum has a value of 1.

7.5 TCLUST Simulation

7.5.1 TCLUST Simulation Setup

This simulation study was designed to mimic one from García-Escudero et al. (2008). The `simulate_gmm` function from the outlierMBC R package was used to simulate 200 data sets, each consisting of 900 observations from three Gaussian components and 100 outliers. Table 7.1 shows the settings which were varied to obtain 200 data sets from 20 different simulation scenarios. Tables 7.2 and 7.3 describe the mean vectors and covariance matrices for the mixture components. Figure 7.4 displays the five simulated data sets that have two variables, equal mixture proportions, and random seed equal to one.

To simulate a data set, we began by sampling 300 observations from each Gaussian component. Then, to simulate an outlier, a sample was drawn from a uniform distribution over the hyper-rectangle containing the Gaussian data. This sample was only accepted if it was extreme with respect to every component in the mixture, in particular, if it was outside that component's theoretical 99^{th} percentile ellipsoid. To determine this, we calculated the sample's Mahalanobis distance from that component's true mean using its true covariance matrix and then checked if it was greater than the 99^{th} percentile of a chi-squared distribution with p degrees of freedom where p is the dimension. Uniform samples were simulated until 100 were accepted as outliers.

7.5.2 TCLUST Simulation Results

We applied outlierMBC-minimum and outlierMBC-backtrack to these data sets, as well as a range of leading model-based methods for clustering and outlier identification. These methods were TCLUST (García-Escudero et al., 2008), ContaminatedMixt, a contaminated Gaussian mixture (Punzo et al., 2018), mclust, a Gaussian mixture with a uniform component (Fraley and Raftery, 2002; Fraley et al., 2022), and OTRIMLE (Coretto and Hennig, 2016). We initialised mclust following the approach proposed by

Table 7.1: Setting options for the TCLUST Simulation.

Simulation Variable	Options	Number
Dimension, p	2 or 6	2
Component Proportions	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ or $\left(\frac{1}{5}, \frac{2}{5}, \frac{2}{5}\right)$	2
Covariance Constants	Model 1, 2, 3, 4, or 5	5
Random Seeds	1, 2, 3, 4, 5, 6, 7, 8, 9, or 10	10

Table 7.2: Component mean vectors and covariance matrices for the TCLUST simulation study. The values of the covariance constants a, b, c, d, e, and f are provided in Table 7.3.

$p = 2$		$p = 6$	
$\boldsymbol{\mu}_g^{(p)}$	$\boldsymbol{\Sigma}_g^{(p)}$	$\boldsymbol{\mu}_g^{(p)}$	$\boldsymbol{\Sigma}_g^{(p)}$
$\boldsymbol{\mu}_1^{(2)} = (0, 8)$	$\boldsymbol{\Sigma}_1^{(2)} = \begin{pmatrix} 1 & 0 \\ 0 & a \end{pmatrix}$	$\boldsymbol{\mu}_1^{(6)} = \left(\boldsymbol{\mu}_1^{(2)}, 0, 0, 0, 0\right)$	$\boldsymbol{\Sigma}_1^{(6)} = \begin{pmatrix} \boldsymbol{\Sigma}_1^{(2)} & 0 \\ 0 & \boldsymbol{I}_4 \end{pmatrix}$
$\boldsymbol{\mu}_2^{(2)} = (8, 0)$	$\boldsymbol{\Sigma}_2^{(2)} = \begin{pmatrix} b & 0 \\ 0 & c \end{pmatrix}$	$\boldsymbol{\mu}_2^{(6)} = \left(\boldsymbol{\mu}_2^{(2)}, 0, 0, 0, 0\right)$	$\boldsymbol{\Sigma}_2^{(6)} = \begin{pmatrix} \boldsymbol{\Sigma}_2^{(2)} & 0 \\ 0 & \boldsymbol{I}_4 \end{pmatrix}$
$\boldsymbol{\mu}_3^{(2)} = (-8, -8)$	$\boldsymbol{\Sigma}_3^{(2)} = \begin{pmatrix} d & e \\ e & f \end{pmatrix}$	$\boldsymbol{\mu}_3^{(6)} = \left(\boldsymbol{\mu}_3^{(2)}, 0, 0, 0, 0\right)$	$\boldsymbol{\Sigma}_3^{(6)} = \begin{pmatrix} \boldsymbol{\Sigma}_3^{(2)} & 0 \\ 0 & \boldsymbol{I}_4 \end{pmatrix}$

Table 7.3: Covariance constants, (a, b, c, d, e, f), for the TCLUST simulation study.

Model 1	Model 2	Model 3
$(1, 1, 1, 1, 0, 1)$	$(5, 1, 5, 1, 0, 5)$	$(5, 5, 1, 3, -2, 3)$

Model 4	Model 5
$(1, 20, 5, 15, -10, 15)$	$(1, 45, 30, 15, -10, 15)$

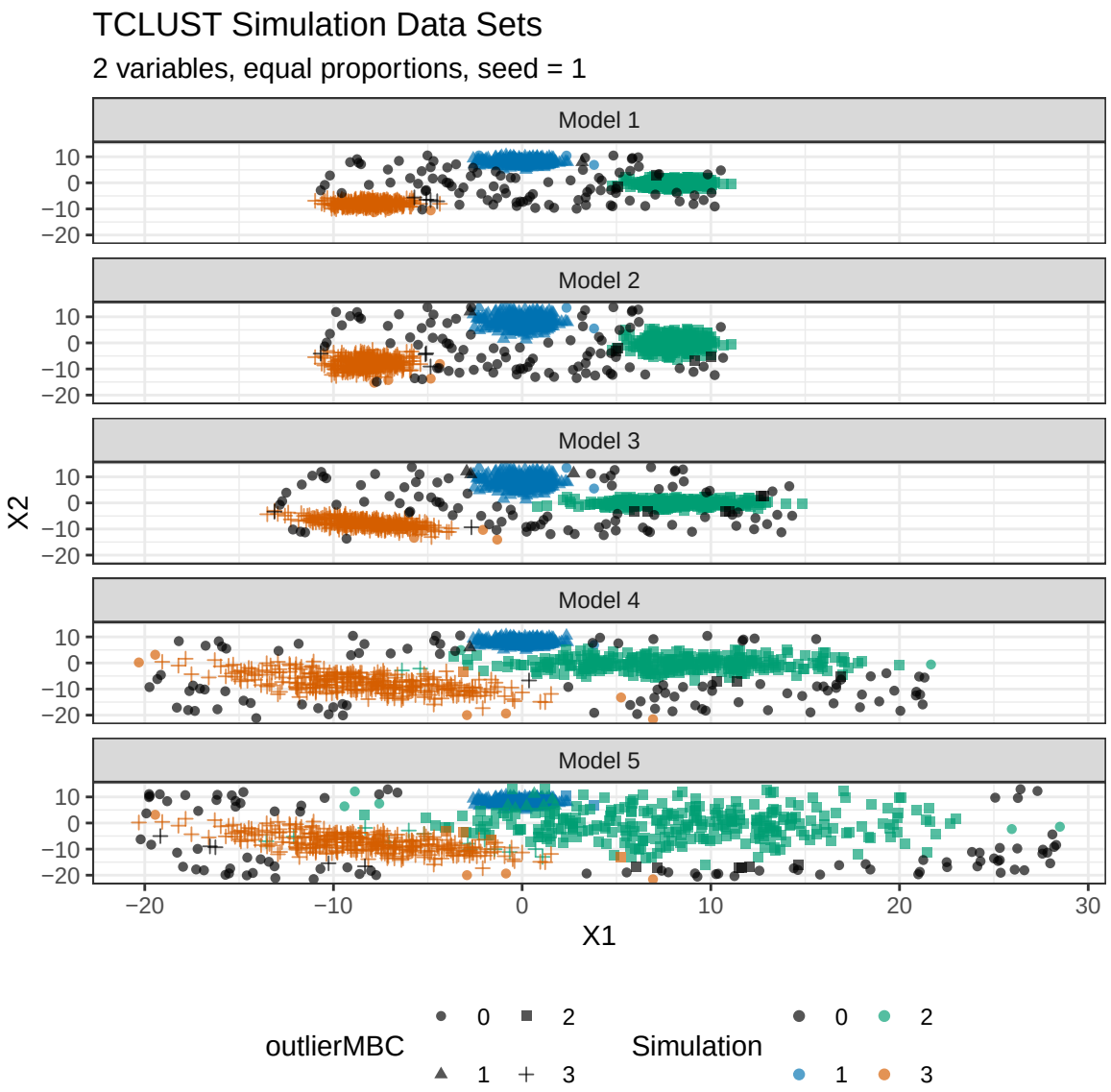


Figure 7.4: Scatter plots for the two-dimensional models (seed = 1) with equal component sizes. Observations' colours correspond to their simulation source and their shapes correspond to the outlierMBC-minimum classification.

Table 7.4: Mean values for the TCLUST simulation study of Adjusted Rand Index (ARI), F_1 Measure with respect to outlier classification only (Outlier F_1), and Numbers of False Positives (#FP), False Negatives (#FN), and Outliers (#Outliers).

Method	ARI	Outlier F_1	#FP	#FN	#Outliers
outlierMBC-minimum	0.96	0.94	4.50	8.00	96.50
outlierMBC-backtrack	0.96	0.92	3.14	11.75	91.39
TCLUST ($\alpha = 0.1$)	0.95	0.93	6.76	6.76	100
mclust	0.94	0.90	25.32	1.02	124.30
ContaminatedMixt	0.89	0.77	32.99	14.26	118.74
OTRIMLE	0.86	0.83	10.13	16.18	93.96

Scrucca (2023).

ContaminatedMixt fits both contaminated and uncontaminated models but for 20 of the 200 data sets in this simulation study its best model according to BIC (Bayesian Information Criterion; Schwarz, 1978) was uncontaminated. This means that for these data sets ContaminatedMixt did not classify any data points as outliers. Similarly, OTRIMLE did not identify any outliers for 14 of the 200 data sets.

Table 7.4 shows that outlierMBC-minimum was the top performing method in terms of both mean F_1 and mean ARI, despite TCLUST being provided with the true number of simulated outliers. outlierMBC-minimum also identified fewer false positives on average than TCLUST, mclust, OTRIMLE, and ContaminatedMixt. outlierMBC-backtrack offered a more conservative alternative with fewer false positives but more false negatives.

ContaminatedMixt, mclust, and OTRIMLE failed quite dramatically for some of the data sets, whereas outlierMBC was more reliable. The highest number of false positives identified by ContaminatedMixt was 295, compared to 204 for mclust, 67 for OTRIMLE, 22 for TCLUST, 14 for outlierMBC-minimum, and 13 for outlierMBC-backtrack. ContaminatedMixt and mclust had more than 50 false positives for 37 data sets and 25 data sets, respectively, while OTRIMLE only exceeded this number once. ContaminatedMixt and OTRIMLE exceeded 50 false negatives 22 times and 25 times, respectively, compared to twice for outlierMBC-backtrack and once for outlierMBC-minimum. This shows that outlierMBC was less likely to produce an extreme number of false positives or false negatives.

Table 7.5: Sizes and properties of the data sets from the OCLUST simulation study.

Data Set	Dimension	No. of Clusters	No. of Points		Sizes of Clusters	
			Original	Added	Smallest	Largest
a1	2	20	3000	210	142	159
a2	2	35	5250	368	143	158
a3	2	50	7500	525	142	159
s1	2	15	5000	350	300	350
s2	2	15	5000	350	300	350
s3	2	15	5000	350	300	350
s4	2	15	5000	350	300	350
Unbalance	2	8	6500	455	100	2000

7.6 OCLUST Simulation

For this simulation study, we replicated eight data sets from Clark and McNicholas (2024) by adding outliers to data sets from Fränti and Sieranoja (2018). In particular, we used the s1, s2, s3, s4, a1, a2, a3, and Unbalance data sets (cs.uef.fi/sipu/datasets). The outliers were sampled uniformly across a hyper-rectangle centred at the mean of each data set with twice the range in each dimension as the data. There was no rejection procedure, meaning that some of these uniform samples lie within the clusters and some false negatives are to be expected when identifying outliers. The sizes and properties of the modified data sets used in this simulation study are shown in Table 7.5. The number of outliers added to each data set was equal to 7% of the original number of points. We set outlierMBC's maximum number of outliers to be equal to 10% of the original number of points, in keeping with the value of OCLUST's equivalent parameter that was used in Clark and McNicholas (2024). When applying outlierMBC, we used the 'reinit' initialisation scheme for the s3 and s4 data sets, and the default 'update' scheme for the other six data sets.

Both outlierMBC and OCLUST implement a preliminary gross outlier removal step. To facilitate a direct comparison and to avoid the subjective elbow choice required by OCLUST's procedure, we implemented both methods with outlierMBC's gross outliers.

Table 7.6: ARI values from the OCLUST simulation study.

Method	a1	a2	a3	s1	s2	s3	s4	Unbal.	Mean
outlierMBC-minimum	0.95	0.94	0.93	0.92	0.88	0.68	0.52	1.00	0.85
outlierMBC-backtrack	0.95	0.94	0.94	0.96	0.90	0.71	0.52	1.00	0.87
OCLUST	0.96	0.94	0.93	0.96	0.90	0.59	0.41	1.00	0.84
mclust	0.97	0.90	0.66	0.96	0.89	0.71	0.57	0.98	0.83
OTRIMLE	0.94	0.93	0.92	0.93	0.89	0.54	0.58	1.00	0.84
ContaminatedMixt	0.80	0.63	0.47	0.87	0.13	0.54	0.22	0.96	0.58

Table 7.7: Outlier F_1 values from the OCLUST simulation study.

Method	a1	a2	a3	s1	s2	s3	s4	Unbal.	Mean
outlierMBC-minimum	0.90	0.87	0.86	0.77	0.77	0.77	0.88	0.97	0.85
outlierMBC-backtrack	0.90	0.88	0.88	0.88	0.86	0.87	0.88	0.97	0.89
OCLUST	0.92	0.88	0.87	0.89	0.86	0.84	0.81	0.97	0.88
mclust	0.92	0.87	0.85	0.87	0.87	0.85	0.88	0.94	0.88
OTRIMLE	0.88	0.85	0.85	0.80	0.80	0.00	0.82	0.94	0.74
ContaminatedMixt	0.00	0.00	0.00	0.73	0.23	0.00	0.00	0.00	0.12

From Tables 7.6 and 7.7, we can see that outlierMBC-backtrack achieves the highest mean ARI and mean F_1 values across these eight data sets. Moreover, Table 7.8 shows that it achieves this while identifying fewer false positives on average than the other methods.

outlierMBC-backtrack outperforms outlierMBC-minimum for these data sets. By design, the number of outliers selected by outlierMBC-backtrack must be less than or equal to the number selected by outlierMBC-minimum. outlierMBC-minimum often identified a high number of outliers relative to the other methods and from Table 7.8 we can see that this resulted in a high number of false positives. By identifying fewer outliers,

Table 7.8: Number of false positives from the OCLUST simulation study.

Method	a1	a2	a3	s1	s2	s3	s4	Unbal.	Mean
outlierMBC-minimum	16	38	55	142	135	111	7	12	64
outlierMBC-backtrack	0	2	5	17	0	5	7	8	6
OCLUST	0	0	1	3	1	22	103	8	17
mclust	1	13	6	29	17	25	25	2	15
OTRIMLE	28	54	83	116	96	0	89	49	64
ContaminatedMixt	0	0	0	163	2202	0	0	0	296

outlierMBC-backtrack was able to avoid these false positives. It also matched or outperformed outlierMBC-minimum with respect to ARI and F_1 for every data set, while selecting fewer outliers. Since the F_1 measure is the harmonic mean of recall and precision, this suggests that outlierMBC-backtrack more than made up for any decrease in its outlier classification recall with an increase in precision. Furthermore, its high ARI score indicates that choosing not to remove these observations did not prevent outlierMBC-backtrack from uncovering the true cluster structure of these data sets.

Figure 7.5 displays six scatter plots of the s3 data set each coloured according to the clustering solution obtained by one of the methods applied in this simulation study. Notably, the OTRIMLE and ContaminatedMixt solutions did not identify any outliers in this case. From Tables 7.6, 7.7, and 7.8, we can observe that outlierMBC-backtrack had the joint highest ARI (0.71) and the highest Outlier F_1 (0.87) for this data set. Among the methods that identified any outliers, it also had the lowest number of false positives (5).

7.7 Swiss Banknote Data

The Swiss Banknote data set consists of 200 observations of 6 numerical variables corresponding to a set of measurements for 100 genuine and 100 counterfeit banknotes. In the TCLUST R package paper (Fritz et al., 2012), the authors propose using TCLUST for this data set with 10% of observations classified as outliers. outlierMBC obtains the exact same solution without having to pre-specify this proportion.

Table 7.9 shows a comparison of the clustering results for a number of leading methods applied to the Swiss Banknote data set. outlierMBC-minimum, outlierMBC-backtrack, TCLUST ($\alpha = 0.1$), and OCLUST reach an identical clustering solution, which is displayed in the right-hand plot of Figure 7.1. mclust and OTRIMLE differ by one point each with mclust removing an additional counterfeit banknote and OTRIMLE retaining a genuine banknote. ContaminatedMixt does not classify any of the banknotes as outliers but does assign 17 counterfeit banknotes to the same cluster as the 100 genuine

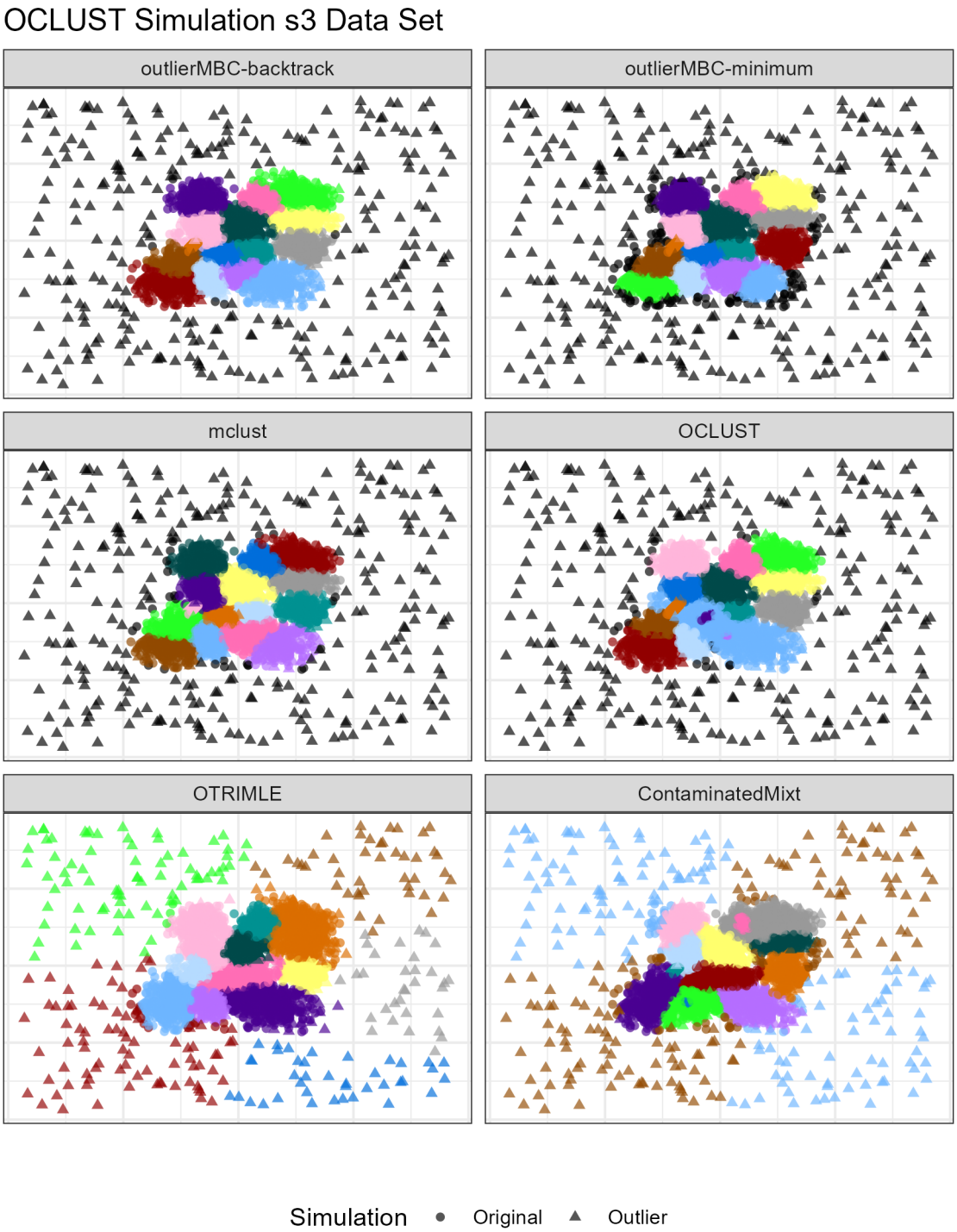


Figure 7.5: Clustering and outlier identification solutions for the s3 data set.

Table 7.9: These tables show each method's cluster assignment of the genuine banknotes (left) and the counterfeit banknotes (right) from the Swiss Banknote data set.

True Class	Genuine		
Assigned Cluster	1	2	Outlier
outlierMBC-minimum	95	0	5
outlierMBC-backtrack	95	0	5
TCLUST ($\alpha = 0.1$)	95	0	5
OCLUST	95	0	5
mclust	95	0	5
OTRIMLE	96	0	4
ContaminatedMixt	100	0	0

True Class	Counterfeit		
Assigned Cluster	1	2	Outlier
outlierMBC-minimum	0	85	15
outlierMBC-backtrack	0	85	15
TCLUST ($\alpha = 0.1$)	0	85	15
OCLUST	0	85	15
mclust	0	84	16
OTRIMLE	0	85	15
ContaminatedMixt	17	83	0

banknotes. In contrast, none of the other methods have both genuine and counterfeit banknotes assigned to the same component.

7.8 healthyFlowData

In this section, we have applied outlierMBC to Sample A1 from the healthyFlowData R package (Azad, 2013). Data from this package were plotted in Section 2.4 and Sample A1, the first sample from Subject A, was analysed in Section 6.4. This flow cytometry data set consists of 19,321 events and 4 markers. Therefore, it is much larger than any of the other data sets featured in this chapter. The next largest is the a3 data set from the OCLUST Simulation which had 8025 data points in total and was only two-dimensional (see Table 7.5). The larger size of this data set and the non-Gaussianity of flow cytometry populations both pose challenges for outlierMBC.

For this analysis, we fitted four Gaussian components and set the maximum number of outliers to be 2000. Figure 7.6 displays outlierMBC's dissimilarity curve for this data set. Our minimum dissimilarity solution chose the optimal number of outliers to be 1204, whereas our more conservative backtrack solution selected 1089. This backtrack solution is displayed in Figure 7.7.

The four populations identified by outlierMBC can be described as follows: CD19+CD3− (orange, $g = 1$), CD19−CD3+CD4+CD8− (blue, $g = 2$), CD19−CD3+CD4−CD8+ (green, $g = 3$), and CD19−CD3− (yellow, $g = 4$). We can associate the first three

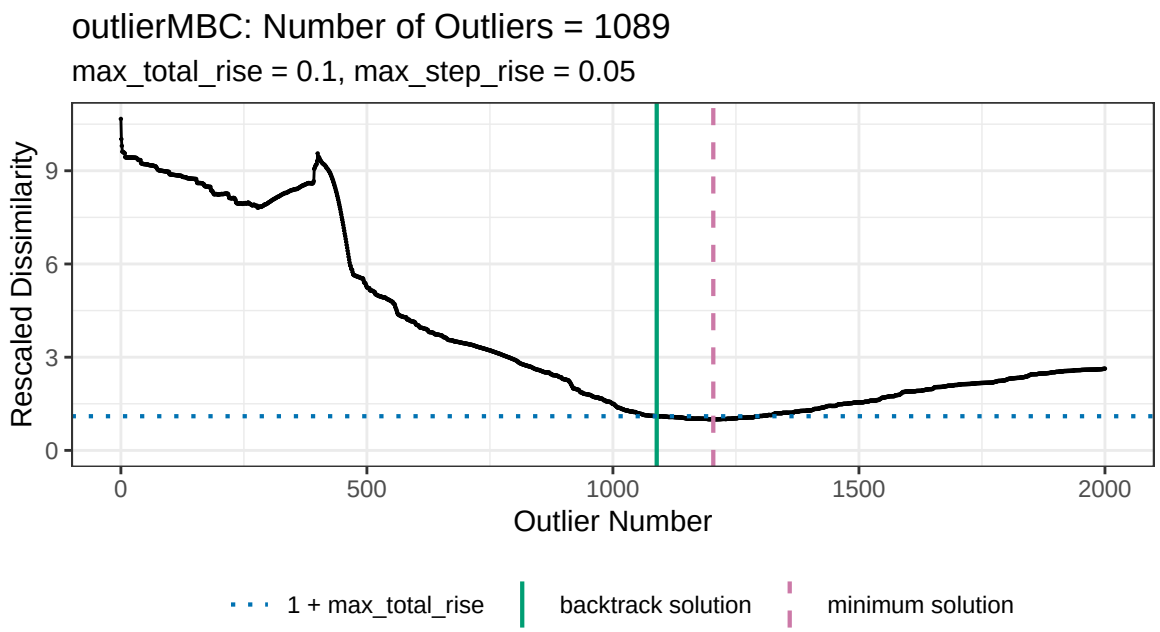


Figure 7.6: Dissimilarity curve for outlierMBC applied to the subject A, sample 1 data set ($n = 19,321$) from the healthyFlowData R package. Our minimum mechanism chose the number of outliers to be 1204, while our 'backtrack' mechanism selected 1089.

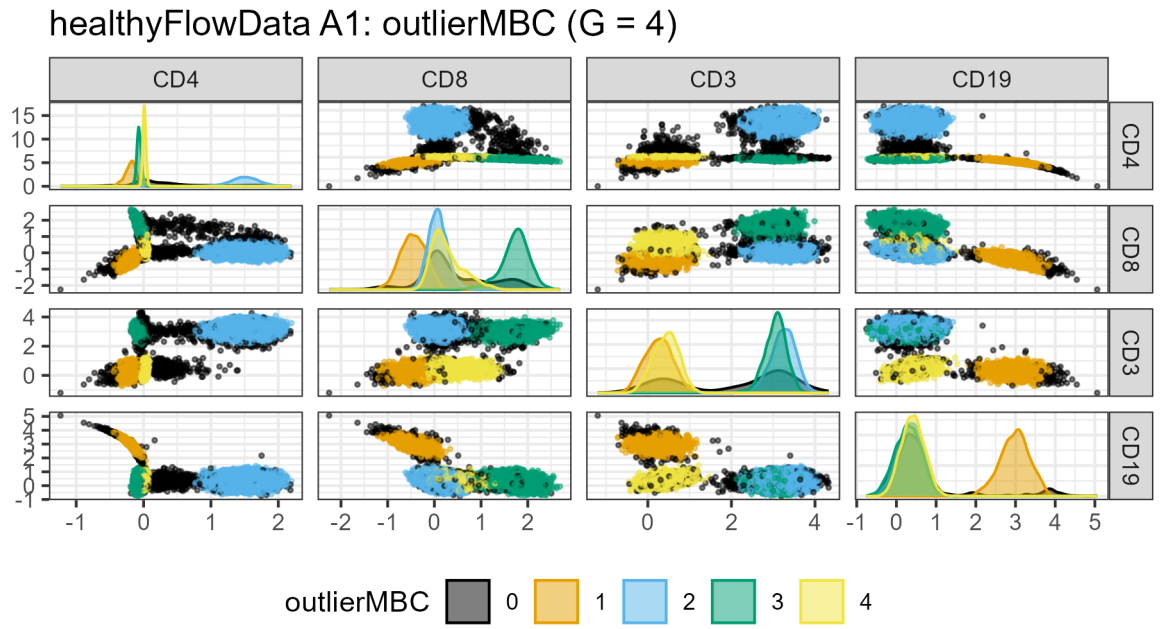


Figure 7.7: Pairs plot of the subject A, sample 1 data set ($n = 19,321$) from the healthyFlowData R package. Events are coloured according to a four-component Gaussian mixture model with 1089 outliers identified by outlierMBC using our 'backtrack' mechanism with 2000 as the maximum number of outliers.

Table 7.10: Contingency table comparing the four-component outlierMBC solution to the gateTree solution from Section 6.4.

	outlierMBC					
gateTree	1	2	3	4	0	Total
B Cells	1559	0	0	0	99	1658
CD4+ T Cells	0	10,178	0	0	128	10,306
CD8+ T Cells	0	0	4033	0	120	4153
Unassigned	0	0	45	2417	742	3204
Total	1559	10,178	4078	2417	1089	19,321

mixture components with known cell populations, namely B Cells, CD4+ T Cells, and CD8+ T Cells. Many of the events identified as outliers by outlierMBC were found in a sparse CD4+CD8+ region between the blue ($g = 2$) and green ($g = 3$) components or were CD3− and CD19− but did not fit the yellow component ($g = 4$) well due to their higher CD4 expression. Some events were also trimmed from the fringes of the mixture components as a result of the non-Gaussianity of the corresponding cell populations. In Table 7.10, we compare outlierMBC's component assignment and outlier identification to the three cell populations identified by gateTree in Section 6.4. From this table, we can observe that the events identified as outliers by outlierMBC consisted primarily of those events left unassigned by gateTree $\left(\frac{742}{1089},\ 68.14\%\right)$, and that three of the outlierMBC components align well with the three gateTree populations. outlierMBC also removed a small proportion of the events identified by gateTree as B Cells $\left(\frac{99}{1658},\ 5.97\%\right)$, CD4+ T Cells $\left(\frac{128}{10{,}306},\ 1.24\%\right)$, and CD8+ T Cells $\left(\frac{120}{4153},\ 2.89\%\right)$. This suggests that the trimming around the edges of these populations is not as severe as it appears in Figure 7.7.

Flow cytometry data sets are challenging for outlierMBC due to their large sample sizes, their unknown numbers of cell populations, and the non-Gaussianity of their populations. As expected based on its assumptions, outlierMBC responds to the non-Gaussianity of the cell populations in this data set by trimming points around their fringes to improve the model fit and reduce the extent to which the remaining data violate its assumptions. Despite this, Table 7.10 suggests that outlierMBC performed reasonably well when identifying the populations which we targeted with gateTree.

7.9 Conclusion

We have presented a method for sequentially identifying outliers by comparing the observed distribution of the data's scaled squared sample Mahalanobis distances to their theoretical distribution. We have replicated simulation studies from García-Escudero et al. (2008) and Clark and McNicholas (2024) and compared outlierMBC to leading methods for model-based clustering and outlier identification. These included TCLUST, OCLUST, mclust, and OTRIMLE. outlierMBC consistently performed well with respect to replicating the true cluster structure (ARI) and classifying outliers (Outlier F_1). It also demonstrated a tendency to identify fewer false positives than the other methods. This is an appealing characteristic for an outlier identification method as unnecessarily removing true observations could result in a misleading view of the data.

8 Sequential Outlier Identification for Linear Cluster-Weighted Models

8.1 Introduction

Cluster-weighted models (CWMs) combine regression and model-based clustering by jointly modelling explanatory variables and a response variable (Gershenfeld, 1997; Ingrassia et al., 2012, 2015). We present an extension of our sequential outlier identification method, outlierMBC, to Gaussian linear cluster-weighted models. That is, CWMs where the relationship between the response variable and the explanatory variables is linear with Gaussian errors. The probability density function of a Gaussian linear cluster-weighted model evaluated at a point $(\boldsymbol{x}_i, y_i)$ is

$$f(\boldsymbol{x}_i, y_i|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\beta}, \boldsymbol{\sigma}^2) = \sum_{g=1}^{G} \pi_g f(y_i|\beta_{0g} + \boldsymbol{\beta}_g^T \boldsymbol{x}_i, \sigma_g^2) f(\boldsymbol{x}_i|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g). \tag{8.1}$$

In other words, a random variable, $(\boldsymbol{X}_i, Y_i)$, from a Gaussian linear cluster-weighted model will be sampled from component g with prior probability π_g, in which case its explanatory variables, $\boldsymbol{X}_i$, will obey a multivariate Gaussian distribution, $\boldsymbol{X}_i \sim \mathcal{N}_p(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$, and the conditional distribution of its response variable, Y_i, given its explanatory variables, $\boldsymbol{X}_i$, will also be Gaussian, $Y_i|\boldsymbol{X}_i = \boldsymbol{x}_i \sim \mathcal{N}(\beta_{0g} + \boldsymbol{\beta}_g^T \boldsymbol{x}_i, \sigma_g^2)$. We will refer to our original method for Gaussian mixture models as outlierMBC-GMM and our extension for Gaussian linear cluster-weighted models as outlierMBC-LCWM. The presence of outliers in a data set can hinder the ability of Gaussian linear cluster-weighted

models to accurately recover the underlying model of the true data points (Torti et al., 2019; García-Escudero et al., 2017; Cappozzo et al., 2023). Like outlierMBC-GMM, outlierMBC-LCWM does not require the number of outliers to be pre-specified and does not require any assumptions about the distribution of the outliers.

8.2 Method

In Chapter 7, we made use of the fact that the squared sample Mahalanobis distances of Gaussian-distributed observations are beta-distributed when scaled appropriately,

$$U_i = \frac{n}{(n-1)^2}\left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right)^T \boldsymbol{S}^{-1}\left(\boldsymbol{X}_i - \bar{\boldsymbol{X}}\right) \sim \text{Beta}\left(\frac{p}{2}, \frac{n-p-1}{2}\right). \tag{8.2}$$

The squared (internally) studentised residuals from a Gaussian multiple linear regression model are also beta-distributed when scaled appropriately (Cook and Weisberg, 1982; Seber and Lee, 2003),

$$V_i = \frac{1}{n-p-1}\left(\frac{Y_i - \hat{Y}_i}{\hat{\sigma}\sqrt{1-h_{ii}}}\right)^2 \sim \text{Beta}\left(\frac{1}{2}, \frac{n-p-2}{2}\right), \tag{8.3}$$

where h_{ii} is the i^{th} diagonal element of the hat matrix, $\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$. Here, we are defining $\hat{\sigma}^2$ as follows,

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{j=1}^{n}(Y_j - \hat{Y}_j)^2.$$

This gives us an alternative characterisation of V_i as

$$V_i = \frac{(Y_i - \hat{Y}_i)^2}{(1-h_{ii})\sum_{j=1}^{n}(Y_j - \hat{Y}_j)^2}.$$

outlierMBC-LCWM exploits the beta distribution of the scaled squared studentised residuals to extend the outlierMBC approach to Gaussian linear CWMs.

At each iteration, m, outlierMBC-LCWM fits a Gaussian linear CWM to the current data, and computes two approximate Wasserstein dissimilarities per mixture component, one for

the scaled squared sample Mahalanobis distances,

$$\hat{D}_{mg}^{(U)} = \frac{1}{T}\sum_{t=1}^{T}\left|F_g^{(U)}\left(\frac{t}{T}\right) - \hat{F}_g^{(U)}\left(\frac{t}{T}\right)\right|, \tag{8.4}$$

and one for the scaled squared studentised residuals,

$$\hat{D}_{mg}^{(V)} = \frac{1}{T}\sum_{t=1}^{T}\left|F_g^{(V)}\left(\frac{t}{T}\right) - \hat{F}_g^{(V)}\left(\frac{t}{T}\right)\right|, \tag{8.5}$$

where $F_g^{(U)}$ and $F_g^{(V)}$ are the theoretical cumulative distribution functions (CDFs) of U and V, $\hat{F}_g^{(U)}$ and $\hat{F}_g^{(V)}$ are the observed CDFs of U and V, and $T = 10,000$. It then aggregates these response and covariate dissimilarities for each component,

$$\hat{D}_{mg} = \sqrt{\frac{1}{2}\left(\hat{D}_{mg}^{(U)}\right)^2 + \frac{1}{2}\left(\hat{D}_{mg}^{(V)}\right)^2}, \tag{8.6}$$

before aggregating across components to obtain a single aggregated dissimilarity for each iteration,

$$\hat{D}_m = \sqrt{\sum_{g=1}^{G}\hat{\pi}_g\hat{D}_{mg}^2}. \tag{8.7}$$

outlierMBC-LCWM then identifies the observation with the lowest joint mixture density as a potential outlier and removes it from the data before the next iteration. As in outlierMBC-GMM, the graph of the aggregated dissimilarity, $\hat{D}_m$, versus the number of removed points, m, is used to choose the optimal number of outliers. The minimum dissimilarity and 'backtrack' solutions are both available, and the user has the option to manually select the number of outliers based on the graph or external factors.

8.3 Simulated Data Example

For this example, we simulated 120 data points from the three-component linear cluster-weighted model described in Table 8.1. We also simulated 12 outliers using a uniform distribution around each component with a rejection criterion to control how unlikely they were under the model.

Table 8.1: Simulation information for the data set in Section 8.3.

Points	Outliers	Mean	Variance	Intercept	Slope	Std. Dev.
30	4	3	1.0	0	0	1
30	4	6	0.1	−75	15	1
60	4	3	1.0	0	5	1

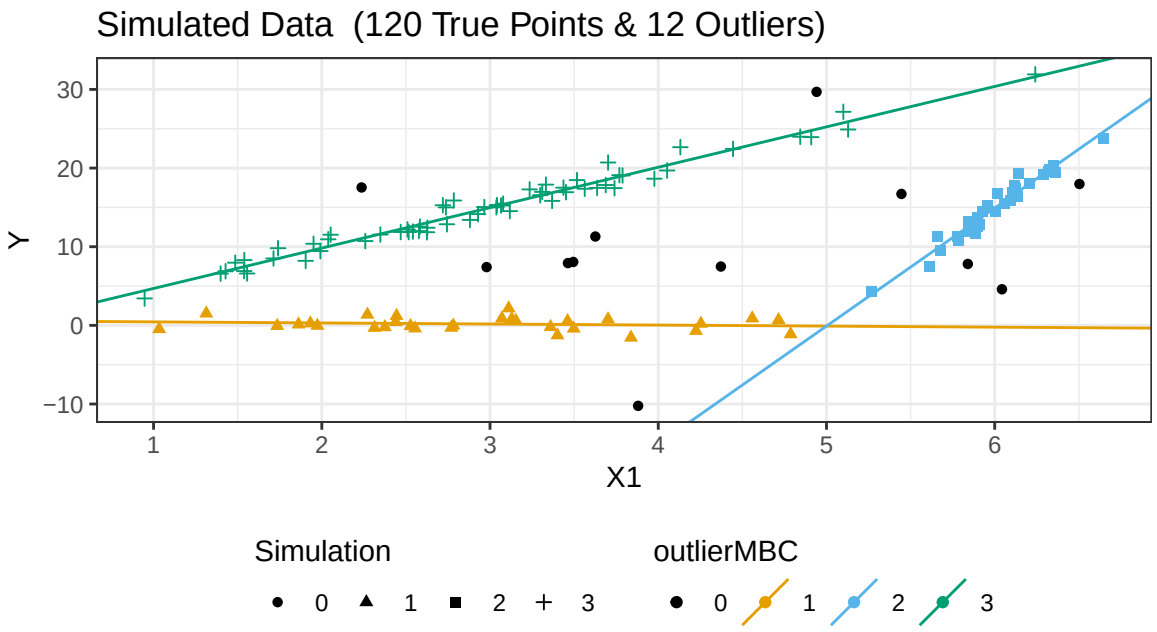


Figure 8.1: Scatter plot of the 120 simulated true points and 12 simulated outliers. It is coloured according to the outlierMBC-LCWM backtrack solution which has correctly identified all 12 outliers with no false positives. The minimum solution instead chose to remove 13 points, with one false positive.

Figure 8.1 illustrates that when applied to this data set, outlierMBC-LCWM is able to perfectly identify the 12 outliers and accurately model the true points. We also fitted a standard linear CWM to this data set to illustrate the difficulty of trying to model the data without removing the outliers. We computed the Mean Squared Error (MSE) for both models using only the 120 true points. outlierMBC-LCWM obtains an MSE value of 0.89, while the standard linear CWM obtains a value of 4.33. This demonstrates that the presence of outliers impacts the ability of the linear CWM to correctly model the response variable.

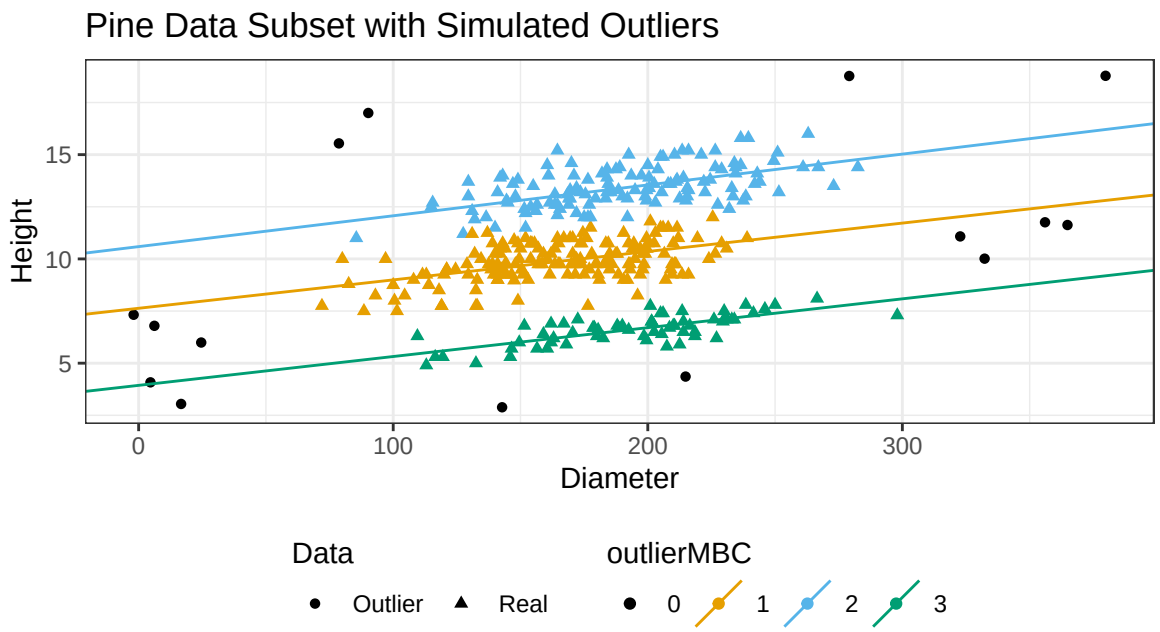


Figure 8.2: Scatter plot of the pine data set after removing 12 data points with extreme values (Diameter $>$ 335 or Height $>$ 22), then adding 15 simulated outliers. It is coloured according to the outlierMBC-LCWM solution which has perfectly identified the 15 simulated outliers. The minimum and backtrack solutions are identical for this data set.

8.4 Real Data Example with Simulated Outliers

The original pine data set from the TCLUST R package consists of height (m) and diameter (mm) measurements for 362 pine trees. There is a clear linear relationship between these two variables as well as an apparent group structure. We augmented this data set by removing 12 points with Diameter $>$ 335 or Height $>$ 22, then adding 15 simulated outliers. Figure 8.2 shows that outlierMBC-LCWM perfectly identified these 15 simulated outliers. If we evaluate the Mean Squared Error for only the 350 true observations, we obtain values of 0.47 for outlierMBC-LCWM and 1.72 for a standard linear CWM fitted to the full augmented data set.

In Figure 8.3, we have plotted the aggregated dissimilarity curve for outlierMBC-LCWM applied to the augmented pine data described above. This corresponds to the value, $\hat{D}_m$, defined in Equation 8.7. We have also derived and plotted two additional quantities, $\hat{D}_m^{(U)}$

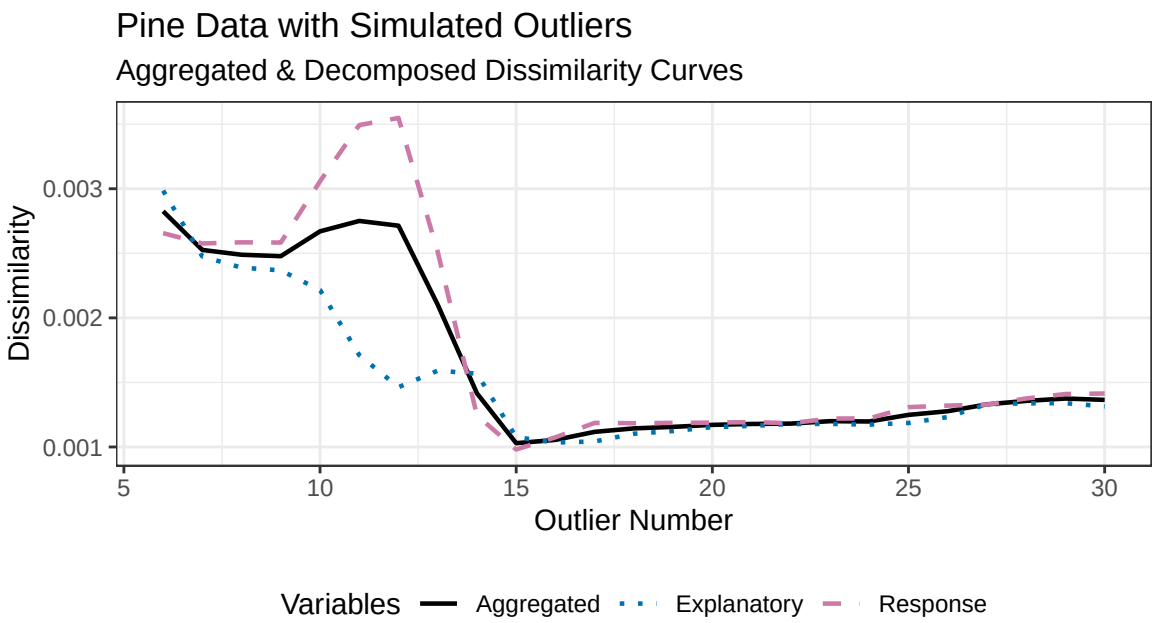


Figure 8.3: Dissimilarity curves for outlierMBC-LCWM applied to the modified pine data set displayed in Figure 8.2 after removing five gross outliers using our `find_gross` function. The solid black line represents $\{\hat{D}_m\}_{m=6}^{30}$ (Equation 8.7), while the dotted blue line and the dashed purple line represent $\left\{\hat{D}_m^{(U)}\right\}_{m=6}^{30}$ and $\left\{\hat{D}_m^{(V)}\right\}_{m=6}^{30}$ (Equation 8.8).

and $\hat{D}_m^{(V)}$, for visualisation purposes. These are defined to be

$$\hat{D}_m^{(U)} = \sqrt{\sum_{g=1}^{G} \hat{\pi}_g \left(\hat{D}_{mg}^{(U)}\right)^2} \text{ and } \hat{D}_m^{(V)} = \sqrt{\sum_{g=1}^{G} \hat{\pi}_g \left(\hat{D}_{mg}^{(V)}\right)^2}, \tag{8.8}$$

where $\hat{D}_{mg}^{(U)}$ and $\hat{D}_{mg}^{(V)}$ are defined in Equations 8.4 and 8.5, respectively. Importantly, $\hat{D}_m$ is not a function of $\hat{D}_m^{(U)}$ and $\hat{D}_m^{(V)}$. However, we can consider $\hat{D}_m^{(U)}$ and $\hat{D}_m^{(V)}$ to represent aggregated dissimilarities for the explanatory and response variables, respectively. In Figure 8.3, we can see that the curve for the aggregated dissimilarity, $\hat{D}_m$, stays between the curves for the aggregated explanatory and response dissimilarities, $\hat{D}_m^{(U)}$ and $\hat{D}_m^{(V)}$. A rise in the aggregated response dissimilarity between $m = 9$ and $m = 12$ is tempered by a decline in the aggregated explanatory dissimilarity. We can also observe that both the aggregated explanatory and response dissimilarities are minimised near fifteen, the number of outliers chosen by outlierMBC.

8.5 Conclusion

By jointly modelling explanatory variables and the response variable, while performing clustering, cluster-weighted models offer a powerful approach for tackling regression problems with a group structure. However, outliers can prevent them from recovering the underlying model. We have presented a method for robustly carrying out model fitting in the presence of outliers by accurately identifying and removing these points. outlierMBC-LCWM demonstrated strong performance on a fully simulated data set and on a real data set with simulated outliers. We believe that the methodical nature of its outlier removal and its minimal assumptions make outlierMBC-LCWM a promising solution for fitting linear CWMs to data with outliers.

The data simulation and model fitting algorithms used in this work are implemented in the outlierMBC R package, which is published on CRAN.

9 Conclusion

9.1 Summary

9.1.1 Overview

In this thesis, we have reviewed some of the leading model-based and non-model-based methods for cell population identification in flow cytometry (Chapters 2 and 3), as well as methods which utilise user information describing the cell populations (Chapter 4). We have also introduced a novel method for user-informed cell population identification and explored the use of constraints for user-informed mixture modelling (Chapters 5 and 6). Finally, we have presented a novel method for sequentially identifying outliers in Gaussian mixture models and Gaussian linear cluster-weighted models (Chapters 7 and 8).

It is clear from Chapters 2, 3, and 4 that the challenge of identifying cell populations in flow cytometry data has motivated a great deal of research. The variety and number of methods which have been developed to address the issues raised by this data is a testament to the difficulty of this problem. Due to the complexity and noisiness of the data, it can often be challenging to discern biologically meaningful differences between events and construct cell populations which are known to be significant in the immunological literature. By employing user-informed constraints in Chapters 5 and 6, we have been able to focus on partitioning the data in a biologically meaningful way. More generally, noise in any data set can hinder the ability of a clustering algorithm to uncover the data's true group structure, particularly if this noise occurs in the form of outliers, whose outsized effect on the parameters of a mixture model can distort its estimated

distribution for the whole data set. In Chapter 7, we presented outlierMBC, a method for sequentially removing outliers while fitting a Gaussian mixture model which determines the optimal number of outliers with respect to the Wasserstein distance between the fitted and theoretical distributions of the scaled squared sample Mahalanobis distances. We further extended this method to Gaussian linear cluster-weighted models in Chapter 8 based on the distribution of scaled squared studentised residuals.

9.1.2 User-Informed Cell Population Identification

In Chapter 4, we discussed a handful of user-informed methods for cell population identification in flow cytometry. That is, methods that are designed to make use of biological knowledge about the data provided by their user, without requiring labelled training data like a classification algorithm would. In contrast to the wide array of unsupervised methods which we reviewed in Chapter 2 and Chapter 3, there has been relatively little research into the development of such algorithms. We believe that this is a promising area for progress due to flow cytometry data's lack of a 'ground truth' class structure and the wealth of relevant knowledge in the immunological community.

As such, we developed gateTree, our own method for constructing cell populations based on user-provided descriptions. In doing so, we have built on the research of three previous methods, in particular. The ACDC method (Lee et al., 2017) introduced the 'cell-type marker table' which gateTree uses to format the cell population descriptions provided by its user. Meanwhile, our kernel density splitting mechanism is similar to the approach used by flowDensity (Malek et al., 2015) and our GMM boundary splitting mechanism mirrors that of cytometree (Commenges et al., 2018). gateTree will first try to use the kernel density valley mechanism, since the assumption that a marker is distributed according to a two-component Gaussian distribution with equal variance terms is rarely accurate. This allows it to determine an optimal split in cases where the marker's distribution is skewed or imbalanced, but there is sufficient data to obtain an accurate kernel density estimate. However, it employs the GMM boundary mechanism as a backup option, which can be useful if the kernel density estimate is unable to delineate the smaller of the two

components. We believe that the strong performance that gateTree has achieved on a range of simulated and real data sets demonstrate that this is a meaningful contribution to this field.

In addition to developing the gateTree method, we also applied it to construct constraints for constrained Gaussian mixture models. When applying this approach to a publicly available benchmark data set, we demonstrated that our constrained mixtures were able to outperform an unconstrained Gaussian mixture model when identifying target populations for which gateTree-constrained sets had been provided. We also carried out an exploratory analysis of another real data set and highlighted how this approach can be used to extend a gateTree solution and explore its unassigned events by identifying additional populations that were not described by the user.

9.1.3 Sequential Outlier Identification for Gaussian Mixtures

In Chapter 7, we introduced outlierMBC, our method for sequentially removing outliers while fitting a sequence of Gaussian mixture models. We believe that by removing outliers one at a time, outlierMBC offers an intuitive and cautious approach to outlier identification. Importantly, outlierMBC does not assume that the outliers follow a known distribution and it does not require the number of outliers to be known *a priori*. The theoretical result underpinning outlierMBC, that scaled squared sample Mahalanobis distances follow a beta distribution (Appendix A1), provides a justified choice for the optimal number of outliers. However, our graphical outputs give the user an intuitive and informative plot, based on which they can make alternative choices for the number of outliers. Our software implementation produces graphs which show clearly how the sequence of models improves as outliers are removed by plotting the aggregated Wasserstein dissimilarity between the fitted and theoretical distributions of these distances. outlierMBC performs well compared to other leading model-based outlier identification methods when applied to a range of data sets.

In Chapter 8, we extended our method to Gaussian linear cluster-weighted models. By

combining mixture modelling and multiple linear regression, these models are a promising tool for a variety of statistical modelling problems. However, there has been less research on outlier identification in cluster-weighted modelling than in traditional model-based clustering. We extended our outlierMBC method to this setting by taking advantage of the similarities between studentised residuals and sample Mahalanobis distances.

9.2 Further Work

9.2.1 Two-Dimensional Splitting in gateTree

gateTree currently only implements univariate splitting. However, we have developed a method for finding a two-dimensional split given a pair of variables, x and y. This method is implemented as a stand-alone function in the gateTree package but it is not incorporated into the main gateTree function. Creating a two-dimensional version of the full gateTree method would require a new algorithm to search for the best split from a selection of bivariate pairs rather than from a selection of single variables.

A non-vertical, straight-line split in two dimensions would be defined by the equation of that line, $y = mx + c$. We can rearrange this equation as $-mx + y = c$. If we define a new variable, z, to be equal to the left-hand side of this rearranged equation, $z = -mx + y$, then we can rewrite the equation as $z = c$. This means that once we have defined the slope of the line, finding the intercept becomes a univariate problem for which we can use the existing gateTree splitting mechanisms. Moreover, we can quantify the goodness of the resulting splits using the valley depth or scaled BIC difference, which were discussed in Section 5.2.1. This would allow us to perform a grid search over a range of possible slope values by estimating the optimal split for each slope value and quantifying the goodness of each slope value's optimal split, then choosing the slope value that results in the best split. In other words, a two-dimensional line is defined by its slope, which we treat as a discrete value and select via a grid search, and its intercept, which we treat as a continuous value and estimate via our univariate splitting mechanisms.

The current two-dimensional splitting functions in the gateTree package require the user to describe the slope of the line as "incline", "vertical", or "decline". Each of these choices corresponds to a selection of nine slope angles spanning 60° arcs centred on 45°, 90°, and 135°, respectively.

- incline $\rightarrow \{15^\circ, 25^\circ, 35^\circ, 45^\circ, 55^\circ, 65^\circ, 75^\circ\}$
- vertical $\rightarrow \{60^\circ, 70^\circ, 80^\circ, 90^\circ, 100^\circ, 110^\circ, 120^\circ\}$
- decline $\rightarrow \{105^\circ, 115^\circ, 125^\circ, 135^\circ, 145^\circ, 155^\circ, 165^\circ\}$

Extending the main gateTree function to implement two-dimensional splits would require a new format for users to specify the population descriptions and a reworking of the internal code to search over variable pairs. Table 9.1 shows a prototype format for the two-dimensional population description table. "decline" splits differentiate between 'double-positive' $(++)$ and 'double-negative' $(--)$ events, whereas "incline" splits differentiate between 'positive-negative' $(+-)$ and 'negative-positive' $(-+)$ events. If a population is defined as positive for a pair of variables with respect to a "decline" split, then it is positive for both the x variable and the y variable $(++)$, and if it is positive with respect to an "incline" split, then it is positive for the x variable and negative for the y variable $(+-)$. Accordingly, if a population is negative for a "decline" split, then it is double-negative $(--)$, and if it is negative for an "incline" split, then it is negative-positive $(-+)$. Therefore, for any "decline" slopes, the order of the x and y variables has no effect on the sign. Meanwhile, for "incline" slopes, swapping the x and y variables is equivalent to changing the sign for that variable pair from $+1$ to -1 or vice versa.

Given a slope angle, θ, we define a new variable z as a linear combination of x and y, $z = \alpha x + \beta y$. If the slope angle, θ, is equal to 90°, then we define $\alpha = 1$ and $\beta = 0$. Otherwise, we compute the slope, $m = \tan(\theta)$, then define $\alpha = \frac{|m|}{\sqrt{1+m^2}}$ and $\beta = \frac{-\text{sgn}(m)}{\sqrt{1+m^2}}$. This gives us that $-\frac{\alpha}{\beta} = m$, $\alpha > 0$, and $\alpha^2 + \beta^2 = 1$. The first of these properties, $-\frac{\alpha}{\beta} = m$, is essential for the equation of the line to be correct, while the second, $\alpha > 0$, is necessary for describing populations as positive or negative with respect to the

Table 9.1: A potential format for a table containing the information required for the user to describe multiple populations in terms of several variable pairs. The details in this table are for illustrative purposes only and are not based on a real data example. It may be beneficial to allow the user to specify some of the angles themselves, as we have done for the CD19 splits in the table.

Population	Quantity	Pair 1	Pair 2	Pair 3
CD4+ T Cells	x	CD3	CD19	CD4
CD4+ T Cells	y	SSC-A	SSC-A	CD8
CD4+ T Cells	Sign	$+1$	-1	$+1$
CD4+ T Cells	Slope	vertical	90	incline
CD8+ T Cells	x	CD3	CD19	CD4
CD8+ T Cells	y	SSC-A	SSC-A	CD8
CD8+ T Cells	Sign	$+1$	-1	-1
CD8+ T Cells	Slope	vertical	90	incline
B Cells	x	CD3	CD19	CD4
B Cells	y	SSC-A	SSC-A	CD8
B Cells	Sign	-1	$+1$	0
B Cells	Slope	vertical	90	none

transformed variable. The third property, $\alpha^2 + \beta^2 = 1$, is not strictly necessary and is only in place so that each slope corresponds to a unique pair of coefficients. However, the function does have an optional argument for min-max scaling the transformed variable to have minimum and maximum values that suit the user. For a split location s, the intercept, c, is $\frac{s}{\beta}$, since $\alpha x + \beta y = s$ is equivalent to $y = -\frac{\alpha}{\beta}x + \frac{s}{\beta}$.

In addition to $z = \alpha x + \beta y$, other noteworthy transformations are $z = \min(x, y)$, which delineates an upper-right quadrant $(++)$, and $z = \max(x, y)$, which delineates a lower-left quadrant $(--)$. These are shown in Figure 9.1, along with "incline" and "decline" straight-line splits.

9.2.2 gateTree Split Location Shrinkage

Currently in gateTree, once we have selected which variable is to be split next, the split location for a given sample is either the estimated split location for that sample or a substitute value based on the optimal split locations for other samples. However, we would like to explore using a non-binary convex combination of this substitute location and the sample-specific location. This would shrink the implemented sample-specific split

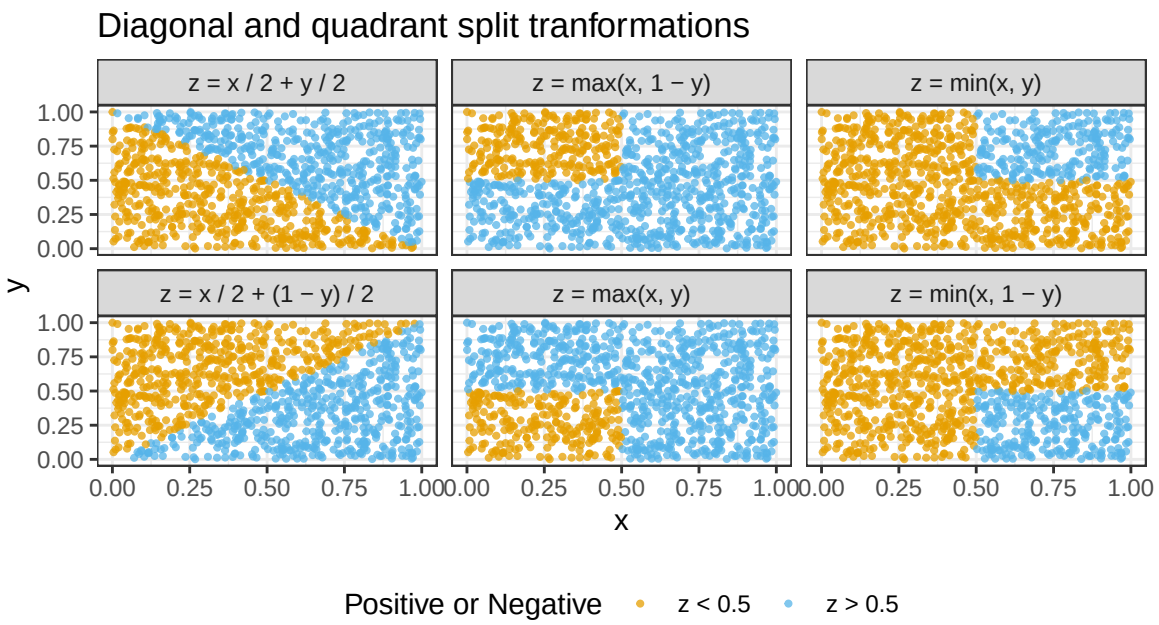


Figure 9.1: Scatter plots of uniformly distributed points coloured according to whether they would have a value of greater than 0.5 under a series of bivariate to univariate transformations.

locations towards the cross-sample substitute split location. Moreover, if the coefficients of the convex combination were computed based on the goodness of the split, then we could smoothly transition between relying solely on the estimated sample-specific split location and relying solely on the cross-sample substitute split location.

For simplicity, we will refer to split locations and scores rather than valleys and valley depths or GMM boundaries and scaled BIC differences. For a given variable, denote the samples' split locations and their corresponding scores as x_i and s_i for $i = 1, \ldots, n$. The current gateTree implementation's final split location for sample i is $y_i = \mathbb{I}(s_i \geq t) \times x_i + \mathbb{I}(s_i < t) \times \mu$, where μ is the substitute value, computed as a weighted median of locations, x_j, with respect to the adjusted scores, $\mathbb{I}(s_j \geq t) \times s_j$, and t is the minimum score threshold.

We can formulate this problem in a pseudo-Bayesian framework, by defining the sample-specific split location, x_i, to be Gaussian-distributed with mean ν_i, $X_i \sim \mathcal{N}(\nu_i, \sigma_i^2)$, where ν_i has a prior distribution, $\nu_i \sim \mathcal{N}(\mu, \gamma_i^2)$. Here, we are assuming the variance terms of both Gaussian distributions, σ_i^2 and γ_i^2, and the cross-sample split

location, μ, to be known. Then, we would define the final split location, y_i, to be the posterior mean of ν_i given x_i, $\mathbb{E}(\nu_i|x_i) = x_i \times \left(\frac{{\gamma_i}^2}{{\gamma_i}^2+{\sigma_i}^2}\right) + \mu \times \left(\frac{{\sigma_i}^2}{{\gamma_i}^2+{\sigma_i}^2}\right)$. In this setting, our current implementation is equivalent to setting γ_i^2 equal to $\mathbb{I}(s_i \geq t)$ and σ_i^2 equal to $\mathbb{I}(s_i < t)$. We could construct a non-binary convex combination by choosing non-binary values of γ_i^2 and σ_i^2, potentially as continuous functions of the split scores, s_i.

Shrinking the sample split locations towards the cross-sample value would reduce the inter-sample variation in split location. This would provide increased consistency when interpreting the difference between positive and negative expression levels for a given marker across samples. In our current implementation, if a sample has a 'good' split, then we implement that split for that sample, and if it has a 'bad' split, then we propagate a split from other samples. At the other end of the spectrum, we could conceivably use the same value for all samples. Some researchers may favour flexibility, allowing split locations to be as sample-specific as possible, whereas others may prefer consistency. Therefore, we believe that providing users with options to control the degree of inter-sample split variation via shrinkage could be beneficial and enable users to further adapt our approach to suit their requirements.

9.2.3 Trimodal Splitting in gateTree

The current implementation of gateTree assumes that each variable's distribution is bimodal. If there are three expression levels for a particular marker, for example, negative, low/mid, and high, then gateTree's valley splitting mechanism will either identify the valley between the negative and low/mid expression peaks or the valley between the low/mid and high expression peaks. This means that if the user wants to identify events with high or negative expression levels of the given marker, gateTree might identify those but also include events with low/mid expression levels. Meanwhile, its GMM boundary mechanism will fit a two-component mixture.

To differentiate between the three modes of a trimodal distribution using kernel density valleys, we could search for the deepest valley first, then separate the marker into two

parts, either side of that deepest valley, and search for the deepest valley in each of the two parts. This should result in either two or three valleys and we could select the two deepest as our potential split locations. Which of the two we implemented would depend on how the user described the target population.

For the GMM boundary mechanism, we would fit a three-component mixture instead of a two-component mixture and to compute the scaled BIC difference, we would compare the three-component GMM to the one-component GMM.

Implementing trimodal splits would also require the user to be able to specify that a marker is trimodal. Currently, the population description table can only have values of -1, 0, and $+1$. Perhaps, we could denote the middle of three expression levels by 0.5 and the highest as 1.5. This would allow the user to indicate that they expect the marker to be trimodal, even if they are not targeting populations featuring all three of the expression levels.

9.2.4 gateTree-Informed Bayesian Priors

In Chapter 5, we showed that our gateTree method can effectively utilise minimal information in the form of population descriptions to identify targeted cell populations. In Chapter 6, we used output from gateTree to improve the performance of a mixture model both through must-link constraints and pre-fixed labels. However, it would be interesting to explore other approaches for using gateTree's output to improve mixture models' performance. One alternative approach would be to operate within a Bayesian framework and construct informative priors based on the populations identified by gateTree.

9.2.5 Combining gateTree and outlierMBC via Constraints

Since the outlierMBC R package is built on the mixture package, which allows its user to pre-label some observations before fitting a mixture model, it would be possible to implement a gateTree-constrained version of outlierMBC. This would enable us to use gateTree to construct target populations based on user descriptions, then extend this

solution to a Gaussian mixture model while sequentially identifying outliers. This naïve implementation would not differentiate between constrained and unconstrained events when removing outliers.

Constrained GMMs require careful initialisation which takes the constraints into account. Therefore, initialising each outlierMBC iteration's GMM during the outlier identification process would require a similarly cautious approach. outlierMBC's "update" initialisation procedure, using the component assignment probabilities estimated during the previous iteration, would be suitable because these probabilities arise from the constrained model and hence account for the constraints. However, outlierMBC's "reinit" scheme would need to be modified to implement a constrained reinitialisation between iterations.

In Section 7.8, we observed that when applied to flow cytometry data, outlierMBC may remove events from the edges of cell populations due to their non-Gaussianity. Therefore, while outliers are present in flow cytometry data, it may be more fruitful to develop non-Gaussian versions of our constrained mixture approach than integrating it into our outlierMBC method. Extending our constrained mixture approach to non-Gaussian models would be relatively straightforward, whereas outlierMBC is intrinsically reliant on Gaussianity assumptions.

9.2.6 Non-Gaussian Extensions of outlierMBC

One of the main limitations of the outlierMBC method is that it can only be used for Gaussian mixture models or Gaussian linear cluster-weighted models. In order to extend it to another distribution, we would need to specify an analogue of the sample Mahalanobis distance and then derive its theoretical distribution under the alternative distribution. This may be possible but it is not work that we have undertaken as part of this thesis. Extending the method to more flexible distributions is innately appealing, but the notion of an outlier might be less clear-cut with respect to these models. For example, the main appeal of a t distribution is the ability of its heavy tails to more readily accommodate observations far from the mean, so adding an outlier trimming step may be redundant in

many situations.

9.2.7 Robust Model Parameters in outlierMBC

In outlierMBC, we iteratively refit the model after each potential outlier is removed. While there are still outliers present, the model fitting process will try to accommodate them. If the model parameters are strongly biased by the presence of these outliers, this could reduce the magnitude of the outliers' sample Mahalanobis distances with respect to these parameters. We could increase the robustness of the method by using alternative parameter estimates such as those produced using the Minimum Covariance Determinant method (Hubert et al., 2018; Rousseeuw, 1985). Another option would be to impose eigenvalue ratio constraints on the covariance matrices, similarly to TCLUST (García-Escudero et al., 2008).

9.2.8 Model Selection for outlierMBC

In outlierMBC, the number of components and the estimated number of outliers are strongly related. Increasing the number of components can decrease the estimated number of outliers as this increases the model's flexibility and its ability to accommodate ill-fitting observations. It is also conceivable that if there were too few mixture components used, requiring the mixture to model two well-separated clusters with a single component, then outliers between those clusters could be missed by the algorithm. In this case, increasing the number of components might lead to more outliers being identified. Since each potential number of components could lead to a different estimated number of outliers (and possibly a different selection of outliers), choosing both of these quantities simultaneously is not straightforward. Making a direct choice with respect to a model selection criterion such as BIC is not possible due to the fact that by removing different outliers, two models can effectively end up fitted to different data sets. Moreover, since the outliers are being trimmed and no assumptions are being made about their distribution, we cannot compute a density value for them.

A graphical approach that we have explored is to apply outlierMBC for a range of

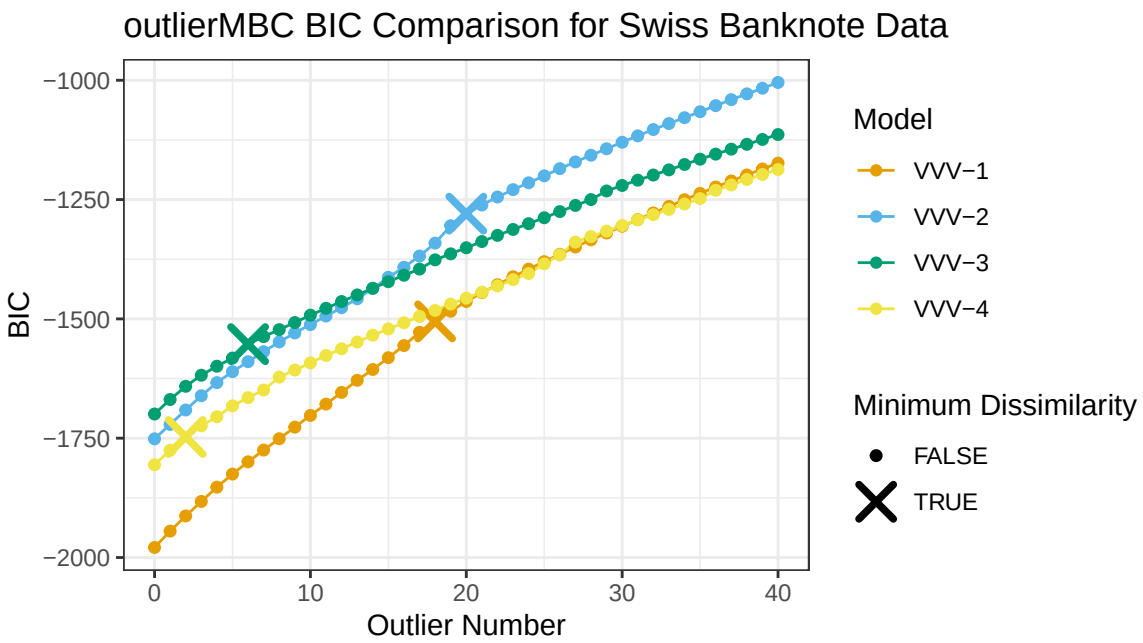


Figure 9.2: BIC plot for the Swiss Banknote data set, comparing one-, two-, three-, and four-component models. According to this graph, if less than fifteen potential outliers are removed, then the three-component model has a better BIC, but once fifteen or more are removed, then the two-component model has a better BIC. This illustrates the connection between component number and outlier number, since the third component consists primarily of low-density observations which can be deemed to be outliers.

component numbers, computing each model's BIC score at every potential number of outliers, then plot the models' BIC scores against the number of outliers removed. We also mark on the curve the number of outliers which each model deemed optimal with respect to the aggregated sample Mahalanobis dissimilarity. While this plot does provide some insight into how model fit, in terms of BIC, is affected by varying the number of components and the number of outliers, we do not have a procedure for choosing the number of components based on this graph.

Perhaps a consensus / ensemble approach could be beneficial, in which we identify a plausible range for the component number and then analyse which observations are consistently identified as outliers across these models. Since we always know the order in which outlierMBC removes the outliers, this ranking information could also be utilised in this consensus approach.

We believe that the relationship between model selection and outlier identification is an area which has not been fully explored and it is an aspect of outlierMBC which requires

further research .

9.2.9 Reductions in Computational Time

The current implementation of outlierMBC is written in R with the model fitting process carried out using the mixture package, which is partly implemented in C++. Rewriting parts of outlierMBC's code in C++ could significantly reduce its computational time.

While gateTree would also benefit from being rewritten in C++ rather than R, another possibility for reducing its computational time would be parallelisation. There are two opportunities for parallelisation in the gateTree algorithm, the first being at the population level and the second being at the sample level for each population. It is worth noting that gateTree's current implementation is quite fast already. In particular, the density valley mechanism is very fast and since this is our primary mechanism, if no GMM boundaries are required, then the whole procedure does not take much time.

Each target population could be constructed in parallel, since there is no interaction between the construction processes of different populations in the current implementation. For example, all target populations on the same tree share the same first split, but the software is written so that this split is computed independently and identically for each row of the population description table. Although this is an inefficiency in the current implementation, it lends itself to parallelisation, which would greatly decrease the computational time.

All of the samples are involved in the construction of each target population, so full sample-level parallelisation would not be possible. However, we should be able to parallelise the initial sample-specific estimation of the optimal splits for every valid variable. This would be less straightforward than parallelising at the population level, but it could have a significant effect, particularly if we parallelised the GMM boundary search across samples, since this seems to be the main computational bottleneck of the algorithm.

Bibliography

Abe, K., K. Minoura, Y. Maeda, H. Nishikawa, and T. Shimamura (2020). Model-based clustering for flow and mass cytometry data with clinical information. *BMC Bioinformatics 21*(Suppl 13), 393.

Aghaeepour, N., P. Chattopadhyay, M. Chikina, T. Dhaene, S. Van Gassen, M. Kursa, B. N. Lambrecht, M. Malek, G. J. McLachlan, Y. Qian, P. Qiu, Y. Saeys, R. Stanton, D. Tong, C. Vens, S. Walkowiak, K. Wang, G. Finak, R. Gottardo, T. Mosmann, G. P. Nolan, R. H. Scheuermann, and R. R. Brinkman (2016). A benchmark for evaluation of algorithms for identification of cellular correlates of clinical outcomes. *Cytometry. Part A: The Journal of the International Society for Analytical Cytology 89*(1), 16–21.

Aghaeepour, N., R. Nikolic, H. H. Hoos, and R. R. Brinkman (2011). Rapid cell population identification in flow cytometry data. *Cytometry Part A 79A*(1), 6–13.

Akaike, H., B. N. Petrov, and F. Csaki (1973). Second international symposium on information theory.

Andrews, J. L. and P. D. McNicholas (2012). Model-based clustering, classification, and discriminant analysis via mixtures of multivariate t-distributions. *Statistics and Computing 22*(5), 1021–1029.

Azad, A. (2013). healthyFlowData: Healthy dataset used by the flowMatch package. R package version 1.46.0.

Azad, A., S. Pyne, and A. Pothen (2012). Matching phosphorylation response patterns of

antigen-receptor-stimulated T cells via flow cytometry. *BMC Bioinformatics 13*(S2), S10.

Azzalini, A. and A. Dalla Valle (1996). The Multivariate Skew-Normal Distribution. *Biometrika 83*(4), 715–726.

Babu, G., A. Gowen, M. Fop, and I. C. Gormley (2025). A consensus-constrained parsimonious Gaussian mixture model for clustering hyperspectral images. *Advances in Data Analysis and Classification 19*(2), 323–359.

Banfield, J. D. and A. E. Raftery (1993). Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics 49*(3), 803–821.

Baudry, J.-P. and G. Celeux (2015). EM for mixtures: Initialization requires special care. *Statistics and Computing 25*(4), 713–726.

Baudry, J.-P., A. E. Raftery, G. Celeux, K. Lo, and R. Gottardo (2010). Combining Mixture Components for Clustering. *Journal of Computational and Graphical Statistics 19*(2), 332–353.

Becht, E., L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell (2019). Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology 37*(1), 38–44.

Biernacki, C., G. Celeux, and G. Govaert (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(7), 719–725.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*(10), P10008.

Bouveyron, C., G. Celeux, T. B. Murphy, and A. E. Raftery (2019). *Model-Based Clustering and Classification for Data Science: With Applications in R*. Cambridge

Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press.

Box, G. E. P. and D. R. Cox (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society. Series B (Methodological) 26*(2), 211–252.

Brookes, M. (2020). The Matrix Reference Manual. `http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html`.

Caligola, S., L. Giacobazzi, S. Canè, A. Vella, A. Adamo, S. Ugel, R. Giugno, and V. Bronte (2024). GateMeClass: Gate Mining and Classification of cytometry data. *Bioinformatics 40*(5), btae322.

Cappozzo, A., L. A. García-Escudero, F. Greselin, and A. Mayo-Iscar (2023). Graphical and Computational Tools to Guide Parameter Choice for the Cluster Weighted Robust Model. *Journal of Computational and Graphical Statistics 32*(3), 1195–1214.

Celeux, G. and G. Govaert (1995). Gaussian parsimonious clustering models. *Pattern Recognition 28*(5), 781–793.

Chan, C., F. Feng, J. Ottinger, D. Foster, M. West, and T. B. Kepler (2008). Statistical mixture modeling for cell subtype identification in flow cytometry. *Cytometry Part A 73A*(8), 693–701.

Cheung, M., J. J. Campbell, L. Whitby, R. J. Thomas, J. Braybrook, and J. Petzing (2021). Current trends in flow cytometry automated data analysis software. *Cytometry Part A 99*(10), 1007–1021.

Clark, K. M. and P. D. McNicholas (2024). Finding outliers in Gaussian model-based clustering. *Journal of Classification 41*, 313–337.

Commenges, D., C. Alkhassim, R. Gottardo, B. Hejblum, and R. Thiébaut (2018). cytometree: A binary tree algorithm for automatic gating in cytometry analysis. *Cytometry Part A 93*(11), 1132–1140.

Cook, R. D. and S. Weisberg (1982). *Residuals and Influence in Regression*. Monographs on Statistics and Applied Probability. Chapman and Hall.

Coretto, P. and C. Hennig (2016). Robust improper maximum likelihood: Tuning, computation, and a comparison with other methods for robust Gaussian clustering. *Journal of the American Statistical Association 111*(516), 1648–1659.

Cron, A., C. Gouttefangeas, J. Frelinger, L. Lin, S. K. Singh, C. M. Britten, M. J. P. Welters, S. H. van der Burg, M. West, and C. Chan (2013). Hierarchical Modeling for Rare Event Detection and Cell Subset Alignment across Flow Cytometry Samples. *PLoS Computational Biology 9*(7), e1003130.

Dang, U. J., R. P. Browne, and P. D. McNicholas (2015). Mixtures of multivariate power exponential distributions. *Biometrics 71*(4), 1081–1089.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*(1), 1–38.

Dundar, M., F. Akova, H. Z. Yerebakan, and B. Rajwa (2014). A non-parametric Bayesian model for joint cell clustering and cluster matching: identification of anomalous sample phenotypes with random effects. *BMC Bioinformatics 15*(1), 314.

Evans, K., T. Love, and S. W. Thurston (2015). Outlier identification in model-based cluster analysis. *Journal of Classification 32*(1), 63–84.

Ferguson, T. S. (1973). A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics 1*(2), 209–230.

Finak, G., A. Bashashati, R. Brinkman, and R. Gottardo (2009). Merging Mixture Components for Cell Population Identification in Flow Cytometry. *Advances in Bioinformatics 2009*, 247646.

Finak, G., M. Langweiler, M. Jaimes, M. Malek, J. Taghiyar, Y. Korin, K. Raddassi, L. Devine, G. Obermoser, M. L. Pekalski, N. Pontikos, A. Diaz, S. Heck, F. Villanova,

N. Terrazzini, F. Kern, Y. Qian, R. Stanton, K. Wang, A. Brandes, J. Ramey, N. Aghaeepour, T. Mosmann, R. H. Scheuermann, E. Reed, K. Palucka, V. Pascual, B. B. Blomberg, F. Nestle, R. B. Nussenblatt, R. R. Brinkman, R. Gottardo, H. Maecker, and J. P. McCoy (2016). Standardizing Flow Cytometry Immunophenotyping Analysis from the Human ImmunoPhenotyping Consortium. *Scientific Reports 6*(1), 20686.

Finak, G., J.-M. Perez, A. Weng, and R. Gottardo (2010). Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics 11*(1), 546.

Fisher, R. A. (1922). On the Mathematical Foundations of Theoretical Statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 222*, 309–368.

Fraley, C. (1998). Algorithms for Model-Based Gaussian Hierarchical Clustering. *SIAM Journal on Scientific Computing 20*(1), 270–281.

Fraley, C. and A. E. Raftery (2002). Model-Based Clustering, Discriminant Analysis, and Density Estimation. *Journal of the American Statistical Association 97*(458), 611–631.

Fraley, C., A. E. Raftery, L. Scrucca, T. B. Murphy, and M. Fop (2022). mclust: Gaussian Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation.

Franczak, B. C., R. P. Browne, and P. D. McNicholas (2014). Mixtures of Shifted Asymmetric Laplace Distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 36*(6), 1149–1157.

Freulon, P., J. Bigot, and B. P. Hejblum (2023). CytOpT: Optimal transport with domain adaptation for interpreting flow cytometry data. *The Annals of Applied Statistics 17*(2), 1086–1104.

Fritz, H., L. A. García-Escudero, and A. Mayo-Iscar (2012). tclust: An R package for a trimming approach to cluster analysis. *Journal of Statistical Software 47*, 1–26.

Fränti, P. and S. Sieranoja (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence 48*(12), 4743–4759.

Gallegos, M. T. and G. Ritter (2005). A Robust Method for Cluster Analysis. *The Annals of Statistics 33*(1), 347–380.

García-Escudero, L. A., A. Gordaliza, F. Greselin, S. Ingrassia, and A. Mayo-Iscar (2017). Robust estimation of mixtures of regressions with random covariates, via trimming and constraints. *Statistics and Computing 27*(2), 377–402.

García-Escudero, L. A., A. Gordaliza, C. Matrán, and A. Mayo-Iscar (2008). A general trimming approach to robust cluster Analysis. *The Annals of Statistics 36*(3), 1324–1345.

Gershenfeld, N. (1997). Nonlinear Inference and Cluster-Weighted Modeling. *Annals of the New York Academy of Sciences 808*(1), 18–24.

Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371*(1984), 20110553.

Ghahramani, Z., G. E. Hinton, et al. (1996). The EM algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto.

Gnanadesikan, R. and J. R. Kettenring (1972). Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics 28*(1), 81–124.

Gormley, I. C., T. B. Murphy, and A. E. Raftery (2023). Model-Based Clustering. *Annual Review of Statistics and Its Application 10*(1), 573–595.

Hartigan, J. A. (1975). *Clustering Algorithms* (99th ed.). USA: John Wiley & Sons, Inc.

Hartigan, J. A. and P. M. Hartigan (1985). The Dip Test of Unimodality. *The Annals of Statistics 13*(1), 70–84.

Hejblum, B. P., C. Alkhassim, R. Gottardo, F. Caron, and R. Thiébaut (2019). Sequential Dirichlet process mixtures of multivariate skew t-distributions for model-based clustering of flow cytometry data. *The Annals of Applied Statistics 13*(1), 638–660.

Hennig, C. (2010). Methods for merging Gaussian mixture components. *Advances in Data Analysis and Classification 4*(1), 3–34.

Hennig, C. (2015). What are the true clusters? *Pattern Recognition Letters 64*, 53–62.

Hennig, C., M. Meila, F. Murtagh, and R. Rocci (2015). *Handbook of Cluster Analysis*. Chapman & Hall / CRC.

Herzenberg, L. A., J. Tung, W. A. Moore, L. A. Herzenberg, and D. R. Parks (2006). Interpreting flow cytometry data: a guide for the perplexed. *Nature Immunology 7*(7), 681–685.

Hsiao, C., M. Liu, R. Stanton, M. McGee, Y. Qian, and R. H. Scheuermann (2016). Mapping cell populations in flow cytometry data for cross-sample comparison using the Friedman-Rafsky test statistic as a distance measure: FCM Cross-Sample Comparison. *Cytometry Part A 89*(1), 71–88.

Hu, Z., C. Jujjavarapu, J. J. Hughey, S. Andorf, H.-C. Lee, P. F. Gherardini, M. H. Spitzer, C. G. Thomas, J. Campbell, P. Dunn, J. Wiser, B. A. Kidd, J. T. Dudley, G. P. Nolan, S. Bhattacharya, and A. J. Butte (2018). MetaCyto: A Tool for Automated Meta-analysis of Mass and Flow Cytometry Data. *Cell Reports 24*(5), 1377–1388.

Hubert, M., M. Debruyne, and P. J. Rousseeuw (2018). Minimum covariance determinant and extensions. *WIREs Computational Statistics 10*(3), e1421.

Ingrassia, S., S. C. Minotti, and G. Vittadini (2012). Local Statistical Modeling via a Cluster-Weighted Approach with Elliptical Distributions. *Journal of Classification 29*(3), 363–401.

Ingrassia, S., A. Punzo, G. Vittadini, and S. C. Minotti (2015). The Generalized Linear Mixed Cluster-Weighted Model. *Journal of Classification 32*(1), 85–113.

Ionita, M., R. Schretzenmair, D. Jones, J. Moore, L.-S. Wang, and W. Rogers (2021). Tailor: Targeting heavy tails in flow cytometry data with fast, interpretable mixture modeling. *Cytometry Part A 99*(2), 133–144.

Jasra, A., C. C. Holmes, and D. A. Stephens (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. *Statistical Science 20*(1), 50–67.

Ji, D., E. Nalisnick, Y. Qian, R. H. Scheuermann, and P. Smyth (2018). Bayesian Trees for Automated Cytometry Data Analysis. In *Proceedings of the 3rd Machine Learning for Healthcare Conference*, pp. 465–483. PMLR.

Johnsson, K., J. Wallin, and M. Fontes (2016). BayesFlow: latent modeling of flow cytometry cell populations. *BMC Bioinformatics 17*(1), 25.

Kass, R. E. and A. E. Raftery (1995). Bayes Factors. *Journal of the American Statistical Association 90*(430), 773–795.

Kobak, D. and P. Berens (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications 10*(1), 5416.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE 78*(9), 1464–1480.

Lee, H.-C., R. Kosoy, C. E. Becker, J. T. Dudley, and B. A. Kidd (2017). Automated cell type discovery and classification through knowledge transfer. *Bioinformatics 33*(11), 1689–1695.

Lee, S. X. (2019). CytoFA: Automated Gating of Mass Cytometry Data via Robust Skew Factor Analzyers. In Q. Yang, Z.-H. Zhou, Z. Gong, M.-L. Zhang, and S.-J. Huang (Eds.), *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, Cham, pp. 514–525. Springer International Publishing.

Lee, S. X. and G. J. McLachlan (2016). Finite mixtures of canonical fundamental skew t-distributions. *Statistics and Computing 26*(3), 573–589.

Lee, S. X. and G. J. McLachlan (2022). An overview of skew distributions in model-based clustering. *Journal of Multivariate Analysis 188*, 104853.

Levine, J. H., E. F. Simonds, S. C. Bendall, K. L. Davis, E.-a. D. Amir, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, R. Finck, A. L. Gedman, I. Radtke, J. R. Downing, D. Pe'er, and G. P. Nolan (2015). Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell 162*(1), 184–197.

Li, H., U. Shaham, K. P. Stanton, Y. Yao, R. R. Montgomery, and Y. Kluger (2017). Gating mass cytometry data by deep learning. *Bioinformatics 33*(21), 3423–3430.

Liechti, T., L. M. Weber, T. M. Ashhurst, N. Stanley, M. Prlic, S. Van Gassen, and F. Mair (2021). An updated guide for the perplexed: cytometry in the high-dimensional era. *Nature Immunology 22*(10), 1190–1197.

Lin, L. and B. P. Hejblum (2021). Bayesian mixture models for cytometry data analysis. *WIREs Computational Statistics 13*(4), e1535.

Lin, L. and J. Li (2017). Clustering with Hidden Markov Model on Variable Blocks. *Journal of Machine Learning Research 18*(110), 1–49.

Lin, T. I. (2009). Maximum likelihood estimation for multivariate skew normal mixture models. *Journal of Multivariate Analysis 100*(2), 257–265.

Liu, P., S. Liu, Y. Fang, X. Xue, J. Zou, G. Tseng, and L. Konnikova (2020). Recent Advances in Computer-Assisted Algorithms for Cell Subtype Identification of Cytometry Data. *Frontiers in Cell and Developmental Biology 8*.

Liu, X., W. Song, B. Y. Wong, T. Zhang, S. Yu, G. N. Lin, and X. Ding (2019). A comparison framework and guideline of clustering methods for mass cytometry data. *Genome Biology 20*(1), 297.

Lo, K., R. R. Brinkman, and R. Gottardo (2008). Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A 73A*(4), 321–332.

Lo, K. and R. Gottardo (2012). Flexible mixture modeling via the multivariate t distribution with the Box-Cox transformation: an alternative to the skew-t distribution. *Statistics and Computing 22*(1), 33–52.

Lock, M. (2003). A Density-based Clustering Method for Flow Cytometry Data.

Lux, M., R. R. Brinkman, C. Chauve, A. Laing, A. Lorenc, L. Abeler-Dörner, and B. Hammer (2018). flowLearn: fast and precise identification and quality checking of cell populations in flow cytometry. *Bioinformatics 34*(13), 2245–2253.

Maaten, L. v. d. and G. Hinton (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research 9*(86), 2579–2605.

Maecker, H. T., J. P. McCoy, and R. Nussenblatt (2012). Standardizing immunophenotyping for the Human Immunology Project. *Nature Reviews Immunology 12*(3), 191–200.

Malek, M., M. J. Taghiyar, L. Chong, G. Finak, R. Gottardo, and R. R. Brinkman (2015). flowDensity: reproducing manual gating of flow cytometry data by automated density-based cell population identification. *Bioinformatics 31*(4), 606–607.

McInnes, L., J. Healy, and J. Melville (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426 [stat].

McLachlan, G. J., S. X. Lee, and S. I. Rathnayake (2019). Finite Mixture Models. *Annual Review of Statistics and Its Application 6*(1), 355–378.

McLachlan, G. J. and D. Peel (1998). Robust cluster analysis via mixtures of multivariate t-distributions. In A. Amin, D. Dori, P. Pudil, and H. Freeman (Eds.), *Advances in Pattern Recognition*, Lecture Notes in Computer Science, Berlin, Heidelberg, pp. 658–666. Springer.

McLachlan, G. J. and D. Peel (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley.

McNicholas, P. D. (2016). Model-Based Clustering. *Journal of Classification 33*(3), 331–373.

McNicholas, P. D. and T. B. Murphy (2008). Parsimonious Gaussian mixture models. *Statistics and Computing 18*(3), 285–296.

Melnykov, V. (2013). Challenges in model-based clustering. *WIREs Computational Statistics 5*(2), 135–148.

Melnykov, V., I. Melnykov, and S. Michael (2016). Semi-supervised model-based clustering with positive and negative constraints. *Advances in Data Analysis and Classification 10*(3), 327–349.

Minoura, K., K. Abe, Y. Maeda, H. Nishikawa, and T. Shimamura (2019). Model-based cell clustering and population tracking for time-series flow cytometry data. *BMC Bioinformatics 20*(Suppl 23), 633.

Minoura, K., K. Abe, Y. Maeda, H. Nishikawa, and T. Shimamura (2021). CYBERTRACK2.0: zero-inflated model-based cell clustering and population tracking method for longitudinal mass cytometry data. *Bioinformatics 37*(11), 1632–1634.

Murphy, K., C. Viroli, and I. C. Gormley (2020). Infinite Mixtures of Infinite Factor Analysers. *Bayesian Analysis 15*(3), 937–963.

Naim, I., S. Datta, J. Rebhahn, J. S. Cavenaugh, T. R. Mosmann, and G. Sharma (2014). SWIFT — scalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, Part 1: Algorithm design. *Cytometry Part A 85*(5), 408–421.

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

Nguyen, R., S. Perfetto, Y. D. Mahnke, P. Chattopadhyay, and M. Roederer (2013). Quantifying spillover spreading for comparing instrument performance and aiding in multicolor panel design. *Cytometry Part A 83A*(3), 306–315.

Orlova, D. Y., S. Meehan, D. Parks, W. A. Moore, C. Meehan, Q. Zhao, E. E. B. Ghosn, L. A. Herzenberg, and G. Walther (2018). QFMatch: multidimensional flow and mass cytometry samples alignment. *Scientific Reports 8*(1), 3291.

O'Hagan, A., T. B. Murphy, and I. C. Gormley (2012). Computational aspects of fitting mixture models via the Expectation–Maximization algorithm. *Computational Statistics & Data Analysis 56*(12), 3843–3864.

O'Hagan, A., T. B. Murphy, I. C. Gormley, P. D. McNicholas, and D. Karlis (2016). Clustering with the multivariate normal inverse Gaussian distribution. *Computational Statistics & Data Analysis 93*, 18–30.

Panaretos, V. M. and Y. Zemel (2019). Statistical aspects of Wasserstein distances. *Annual Review of Statistics and Its Application 6*(Volume 6, 2019), 405–431.

Parks, D. R., M. Roederer, and W. A. Moore (2006). A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry Part A 69A*(6), 541–551.

Pearson, K. (1894). Contributions to the Mathematical Theory of Evolution. *Philosophical Transactions of the Royal Society of London. A 185*, 71–110.

Pedersen, C. B. and L. R. Olsen (2020). Algorithmic Clustering Of Single-Cell Cytometry Data—How Unsupervised Are These Analyses Really? *Cytometry Part A 97*(3), 219–221.

Peel, D. and G. J. McLachlan (2000). Robust mixture modelling using the t distribution. *Statistics and Computing 10*(4), 339–348.

Pocuca, N., R. P. Browne, and P. D. McNicholas (2024). mixture: Mixture models for clustering and classification.

Punzo, A., A. Mazza, and P. D. McNicholas (2018). Contaminatedmixt: An R package for fitting parsimonious mixtures of multivariate contaminated Normal distributions. *Journal of Statistical Software 85*, 1–25.

Punzo, A. and P. D. McNicholas (2016). Parsimonious mixtures of multivariate contaminated normal distributions. *Biometrical Journal 58*(6), 1506–1537.

Pyne, S., X. Hu, K. Wang, E. Rossin, T.-I. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. D. Jager, and J. P. Mesirov (2009). Automated high-dimensional flow cytometric data analysis. *Proceedings of the National Academy of Sciences 106*(21), 8519–8524.

Pyne, S., S. X. Lee, K. Wang, J. Irish, P. Tamayo, M.-D. Nazaire, T. Duong, S.-K. Ng, D. Hafler, R. Levy, G. P. Nolan, J. Mesirov, and G. J. McLachlan (2014). Joint Modeling and Registration of Cell Populations in Cohorts of High-Dimensional Flow Cytometric Data. *PLoS ONE 9*(7), e100334.

Qian, Y., C. Wei, F. Eun-Hyung Lee, J. Campbell, J. Halliley, J. A. Lee, J. Cai, Y. M. Kong, E. Sadat, E. Thomson, P. Dunn, A. C. Seegmiller, N. J. Karandikar, C. M. Tipton, T. Mosmann, I. Sanz, and R. H. Scheuermann (2010). Elucidation of seventeen human peripheral blood B-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data. *Cytometry Part B: Clinical Cytometry 78B*(S1), S69–S82.

Qiu, P. (2017). Toward deterministic and semiautomated SPADE analysis. *Cytometry. Part A: The Journal of the International Society for Analytical Cytology 91*(3), 281–289.

Qiu, P., E. F. Simonds, S. C. Bendall, K. D. Gibbs, R. V. Bruggner, M. D. Linderman, K. Sachs, G. P. Nolan, and S. K. Plevritis (2011). Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. *Nature Biotechnology 29*(10), 886–891.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworths.

Roederer, M. (2002). Compensation in Flow Cytometry. *Current Protocols in Cytometry 22*(1), 1.14.1–1.14.20.

Rousseeuw, P. (1985). Multivariate Estimation with High Breakdown Point. In W. Grossmann, G. C. Pflug, I. Vincze, and W. Wertz (Eds.), *Mathematical Statistics and Applications*, pp. 283–297. Dordrecht: Springer Netherlands.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20*, 53–65.

Saeys, Y., S. Van Gassen, and B. N. Lambrecht (2016). Computational flow cytometry: helping to make sense of high-dimensional immunology data. *Nature Reviews Immunology 16*(7), 449–462.

Samusik, N., Z. Good, M. H. Spitzer, K. L. Davis, and G. P. Nolan (2016). Automated mapping of phenotype space with single-cell data. *Nature Methods 13*(6), 493–496.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics 6*(2), 461–464.

Scrucca, L. (2023). Entropy-based anomaly detection for Gaussian mixture modeling. *Algorithms 16*(4), 195.

Scrucca, L., M. Fop, T. B. Murphy, and A. E. Raftery (2016). mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R journal 8*(1), 289–317.

Scrucca, L. and A. E. Raftery (2015). Improved initialisation of model-based clustering using Gaussian hierarchical partitions. *Adv. Data Anal. Classif. 9*(4), 447–460.

Seber, G. A. F. and A. J. Lee (2003). Departures from Assumptions: Diagnosis and Remedies. In *Linear Regression Analysis*, Wiley Series in Probability and Statistics, pp. 266–328. John Wiley & Sons, Inc.

Shekhar, K., P. Brodin, M. M. Davis, and A. K. Chakraborty (2014). Automatic Classification of Cellular Expression by Nonlinear Stochastic Embedding (ACCENSE). *Proceedings of the National Academy of Sciences of the United States of America 111*(1), 202–207.

Shental, N., A. Bar-hillel, T. Hertz, and D. Weinshall (2003). Computing Gaussian Mixture Models with EM Using Equivalence Constraints. In *Advances in Neural Information Processing Systems*, Volume 16. MIT Press.

Stephens, M. (2000). Dealing With Label Switching in Mixture Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology 62*(4), 795–809.

Stubbington, M. J. T., O. Rozenblatt-Rosen, A. Regev, and S. A. Teichmann (2017). Single-cell transcriptomics to explore the immune system in health and disease. *Science 358*(6359), 58–63.

Sörensen, T., S. Baumgart, P. Durek, A. Grützkau, and T. Häupl (2015). immunoClust – An automated analysis pipeline for the identification of immunophenotypic signatures in high-dimensional cytometric datasets. *Cytometry. Part A: The Journal of the International Society for Analytical Cytology 87*(7), 603–615.

The FlowCAP Consortium, The DREAM Consortium, N. Aghaeepour, G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann (2013). Critical assessment of automated flow cytometry data analysis techniques. *Nature Methods 10*(3), 228–238.

Torti, F., D. Perrotta, M. Riani, and A. Cerioli (2019). Assessing trimming methodologies for clustering linear regression data. *Advances in Data Analysis and Classification 13*(1), 227–257.

Tukey, J. W. (1959). *A Survey of Sampling from Contaminated Distributions*. Princeton University.

Van Gassen, S., B. Callebaut, M. J. Van Helden, B. N. Lambrecht, P. Demeester, T. Dhaene, and Y. Saeys (2015). FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A 87*(7), 636–645.

Van Gassen, S., B. Gaudilliere, M. S. Angst, Y. Saeys, and N. Aghaeepour (2020). CytoNorm: A Normalization Algorithm for Cytometry Data. *Cytometry Part A 97*(3), 268–278.

Ververidis, D. and C. Kotropoulos (2008). Gaussian mixture modeling by exploiting the Mahalanobis distance. *IEEE Transactions on Signal Processing 56*(7), 2797–2811.

Wade, S. and Z. Ghahramani (2018). Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis 13*(2), 559–626.

Wattenberg, M., F. Viégas, and I. Johnson (2016). How to Use t-SNE Effectively. *Distill 1*(10), e2.

Weber, L. M. and M. D. Robinson (2016). Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. *Cytometry Part A 89*(12), 1084–1096.

Weber, L. M. and C. Soneson (2019). HDCytoData: Collection of high-dimensional cytometry benchmark datasets in Bioconductor object formats. *F1000Research 8*, 1459.

Wilks, S. S. (1963). Multivariate statistical outliers. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002) 25*(4), 407–426. Publisher: Springer.

Wolfe, D. A. (2012). Ranked Set Sampling: Its Relevance and Impact on Statistical Inference. *International Scholarly Research Notices 2012*(1), 568385.

Zhu, X. and V. Melnykov (2018). Manly transformation in finite mixture modeling. *Computational Statistics & Data Analysis 121*, 190–208.

A1 Distribution of Scaled Squared Sample Mahalanobis Distances

A1.1 Summary

This appendix proves that the scaled squared sample Mahalanobis distances discussed in Chapters 7 and 8 are beta-distributed. This is the central result on which outlierMBC is based. The proof contained in this appendix is not original, but is presented here for convenience (Ververidis and Kotropoulos, 2008; Wilks, 1963; Gnanadesikan and Kettenring, 1972; Clark and McNicholas, 2024).

A1.2 Definitions

Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be independent and identically distributed Gaussian random variables with mean vector, $\boldsymbol{\mu}$, and covariance matrix, $\boldsymbol{\Sigma}$, that is $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Denote $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ by $\boldsymbol{\mathcal{X}}$ and the set obtained by removing the j^{th} random variable, $\boldsymbol{X}_j$ from $\boldsymbol{\mathcal{X}}$ by $\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j$.

Definition 1. *Define $\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}$ and $\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ to be the sample means of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j$, respectively. That is,*

$$\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}_i, \text{ and } \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j} = \frac{1}{n-1} \sum_{i \neq j} \boldsymbol{X}_i.$$

Definition 2. *Define $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}$ and $\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ to be the scatter matrices of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j$,*

respectively. That is,

$$\boldsymbol{A}_{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{n} (\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T, \text{ and } \boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j} = \sum_{i \neq j} (\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j})(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j})^T.$$

Definition 3. *Define $\boldsymbol{S}_{\boldsymbol{\mathcal{X}}}$ and $\boldsymbol{S}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ to be the sample covariance matrices of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j$, respectively. That is,*

$$\boldsymbol{S}_{\boldsymbol{\mathcal{X}}} = \frac{1}{n-1} \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}, \text{ and } \boldsymbol{S}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j} = \frac{1}{n-2} \boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}.$$

Definition 4. *Define Q_j to be the ratio of the determinants of $\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ and $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}$,*

$$Q_j = \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|}.$$

A1.3 Proof Outline

We will first prove Theorem 1 which states that the scatter matrix determinant ratio, Q_j, is beta-distributed. Then, we will show that Q_j can be expressed in terms of the squared sample Mahalanobis distance of $\boldsymbol{X}_j$. From this, it will follow that squared sample Mahalanobis distances are beta-distributed when scaled appropriately (Theorem 2).

Lemma 4 shows that the scatter matrices $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}$ and $\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ differ by a term $\boldsymbol{b}_j \boldsymbol{b}_j^T$ where $\boldsymbol{b}_j \in \mathbb{R}^p$. The determinants of these scatter matrices are the numerator and denominator of Q_j. In the proof of Lemma 8, we replace $\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j}$ with $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}} - \boldsymbol{b}_j \boldsymbol{b}_j^T$ and derive an expression for Q_j in terms of the j^{th} sample Mahalanobis distance. In the proof of Theorem 1, we replace $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}$ with $\boldsymbol{A}_{\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j} + \boldsymbol{b}_j \boldsymbol{b}_j^T$ and derive an expression for Q_j in terms of a quadratic form that follows a Hotelling's $\mathcal{T}^2$ distribution.

Theorem 1. *Q_j is beta-distributed with parameters $\frac{n-p-1}{2}$ and $\frac{p}{2}$. That is,*

$$Q_j \sim \textit{Beta}\left(\frac{n-p-1}{2}, \frac{p}{2}\right).$$

Theorem 2. *The Mahalanobis distance between a Gaussian random variable, $\boldsymbol{X}_j$, and*

the sample mean, $\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}$, with respect to the sample covariance matrix, $\bar{\boldsymbol{S}}_{\boldsymbol{\mathcal{X}}}$, is beta-distributed with parameters $\frac{p}{2}$ and $\frac{n-p-1}{2}$, when scaled by a factor of $\frac{n}{(n-1)^2}$. That is,

$$\frac{n}{(n-1)^2}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}}^{-1}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}) \sim \textit{Beta}\left(\frac{p}{2}, \frac{n-p-1}{2}\right).$$

A1.4 Lemmas

Lemma 1. *The scatter matrix of $\boldsymbol{\mathcal{X}} \setminus \boldsymbol{X}_j$, $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}$, is a Wishart random variable with scale matrix $\boldsymbol{\Sigma}$ and $n-2$ degrees of freedom. That is,*

$$\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j} \sim \mathcal{W}_p(\boldsymbol{\Sigma}, n-2).$$

Lemma 2. *If $\boldsymbol{W}$ is Wishart-distributed with scale matrix $\boldsymbol{I}_p$ and k degrees of freedom, and $\boldsymbol{Z}$ is Gaussian-distributed with mean vector $\boldsymbol{0}$ and covariance matrix $\boldsymbol{I}_p$, and $\boldsymbol{W}$ and $\boldsymbol{Z}$ are independent of each other, then $k\boldsymbol{Z}^T\boldsymbol{W}^{-1}\boldsymbol{Z}$ is distributed according to Hotelling's $\mathcal{T}^2$ distribution with parameters p and k. That is, if $\boldsymbol{W} \sim \mathcal{W}_p(\boldsymbol{I}_p, k)$ and $\boldsymbol{Z} \sim \mathcal{N}_p(\boldsymbol{0}, \boldsymbol{I}_p)$ are independent, then $k\boldsymbol{Z}^T\boldsymbol{W}^{-1}\boldsymbol{Z} \sim \mathcal{T}^2(p,k)$.*

Lemma 3 (Matrix Determinant Lemma)**.** *Let $\boldsymbol{M}$ be a $p \times p$ matrix, and let $\boldsymbol{u}$ and $\boldsymbol{v}$ be p-dimensional vectors. Then,*

$$|\boldsymbol{M} + \boldsymbol{u}\boldsymbol{v}^T| = |\boldsymbol{M}|(1 + \boldsymbol{v}^T\boldsymbol{M}^{-1}\boldsymbol{u}).$$

Proof of Lemma 3 (Brookes, 2020).

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{u} \\ -\boldsymbol{v}^T & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{M} & \boldsymbol{0} \\ -\boldsymbol{v}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{M}^{-1}\boldsymbol{u} \\ 0 & 1 + \boldsymbol{v}^T\boldsymbol{M}^{-1}\boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{u} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{M} + \boldsymbol{u}\boldsymbol{v}^T & \boldsymbol{0} \\ -\boldsymbol{v}^T & 1 \end{bmatrix}$$

$$\therefore |\boldsymbol{M}||1 + \boldsymbol{v}^T\boldsymbol{M}^{-1}\boldsymbol{u}| = |\boldsymbol{M} + \boldsymbol{u}\boldsymbol{v}^T|$$

□

Lemma 4. *We can express $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}$ as the sum of $\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}$ and the outer product of two*

p-dimensional vectors, as follows,

$$\boldsymbol{A}_{\mathcal{X}} = \boldsymbol{A}_{\mathcal{X}\setminus \boldsymbol{X}_j} + \boldsymbol{u}\boldsymbol{v}^T, \quad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^p.$$

Proof of Lemma 4.

$$\begin{aligned}
\boldsymbol{A}_{\mathcal{X}} - \boldsymbol{A}_{\mathcal{X}\setminus \boldsymbol{X}_j} &= \sum_{i=1}^{n} (\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\mathcal{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\mathcal{X}})^T - \sum_{i\neq j} (\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j})(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j})^T \\
&= \left[\sum_{i=1}^{n} \boldsymbol{X}_i \boldsymbol{X}_i^T - n\bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T\right] - \left[\sum_{i\neq j} \boldsymbol{X}_i \boldsymbol{X}_i^T - (n-1)\bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j} \bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j}^T\right] \\
&= \boldsymbol{X}_j \boldsymbol{X}_j^T - \left[n\bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T - (n-1)\bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j} \bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j}^T\right]
\end{aligned}$$

We can simplify the above expression further by expressing $\bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j}$ in terms of $\bar{\boldsymbol{X}}_{\mathcal{X}}$ and $\boldsymbol{X}_j$, as $\bar{\boldsymbol{X}}_{\mathcal{X}\setminus \boldsymbol{X}_j} = \frac{n\bar{\boldsymbol{X}}_{\mathcal{X}} - \boldsymbol{X}_j}{n-1}$.

$$\begin{aligned}
\boldsymbol{A}_{\mathcal{X}} - \boldsymbol{A}_{\mathcal{X}\setminus \boldsymbol{X}_j} &= \boldsymbol{X}_j \boldsymbol{X}_j^T - \left[n\bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T - (n-1)\left(\frac{n\bar{\boldsymbol{X}}_{\mathcal{X}} - \boldsymbol{X}_j}{n-1}\right)\left(\frac{n\bar{\boldsymbol{X}}_{\mathcal{X}} - \boldsymbol{X}_j}{n-1}\right)^T\right] \\
&= \boldsymbol{X}_j \boldsymbol{X}_j^T - n\bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T + \frac{n^2 \bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T - n\bar{\boldsymbol{X}}_{\mathcal{X}} \boldsymbol{X}_j^T - n\boldsymbol{X}_j \bar{\boldsymbol{X}}_{\mathcal{X}}^T + \boldsymbol{X}_j \boldsymbol{X}_j^T}{n-1} \\
&= \frac{n}{n-1}\left[\boldsymbol{X}_j \boldsymbol{X}_j^T + \bar{\boldsymbol{X}}_{\mathcal{X}} \bar{\boldsymbol{X}}_{\mathcal{X}}^T - \bar{\boldsymbol{X}}_{\mathcal{X}} \boldsymbol{X}_j^T - \boldsymbol{X}_j \bar{\boldsymbol{X}}_{\mathcal{X}}^T\right] \\
&= \frac{n}{n-1}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}})(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}})^T \\
&= \left[\sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}})\right]\left[\sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}})\right]^T \\
&= \boldsymbol{b}_j \boldsymbol{b}_j^T \quad \text{where } \boldsymbol{b}_j := \sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}}) \\
\boldsymbol{A}_{\mathcal{X}} &= \boldsymbol{A}_{\mathcal{X}\setminus \boldsymbol{X}_j} + \boldsymbol{b}_j \boldsymbol{b}_j^T
\end{aligned}$$

□

Lemma 5. *$\boldsymbol{b}_j$ is a Gaussian random variable with mean vector $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. That is,*

$$\boldsymbol{b}_j = \sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\mathcal{X}}) \sim \mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma}).$$

Proof of Lemma 5. We can express $\boldsymbol{b}_j$ as the sum of two independent Gaussian random

variables.

$$\begin{aligned}
\boldsymbol{b}_j &= \sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}) \\
&= \sqrt{\frac{n}{n-1}}\left[\boldsymbol{X}_j - \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{X}_i\right] \\
&= \sqrt{\frac{n}{n-1}}\left[\left(1-\frac{1}{n}\right)\boldsymbol{X}_j - \frac{1}{n}\sum_{i\neq j}\boldsymbol{X}_i\right] \\
&= \sqrt{\frac{n}{n-1}}\left[\frac{n-1}{n}\boldsymbol{X}_j - \frac{n-1}{n}\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\right] \\
&= \sqrt{\frac{n-1}{n}}\left[\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\right]
\end{aligned}$$

Since we know the distributions of $\boldsymbol{X}_j$ and $\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}$, and they are independent of each other, this allows us to derive the probability distribution of $\boldsymbol{b}_j$. In particular, $\boldsymbol{X}_j \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j} \sim \mathcal{N}_p\left(\boldsymbol{\mu}, \frac{1}{n-1}\boldsymbol{\Sigma}\right)$.

$$\begin{aligned}
\boldsymbol{b}_j &\sim \mathcal{N}_p\left(\sqrt{\frac{n-1}{n}}\boldsymbol{\mu} - \sqrt{\frac{n-1}{n}}\boldsymbol{\mu}, \left[\sqrt{\frac{n-1}{n}}\right]^2\boldsymbol{\Sigma} + \left[\sqrt{\frac{n-1}{n}}\right]^2\left(\frac{1}{n-1}\right)\boldsymbol{\Sigma}\right) \\
&\sim \mathcal{N}_p\left(\boldsymbol{0}, \left[\frac{n-1}{n} + \left(\frac{n-1}{n}\right)\left(\frac{1}{n-1}\right)\right]\boldsymbol{\Sigma}\right) \\
&\sim \mathcal{N}_p\left(\boldsymbol{0}, \boldsymbol{\Sigma}\right)
\end{aligned}$$

□

Lemma 6. *Define T_j^2 to be the quadratic form $\boldsymbol{b}_j^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j$. T_j^2 follows Hotelling's $\mathcal{T}^2$ distribution with parameters p and $n-2$. That is,*

$$T_j^2 = \boldsymbol{b}_j^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j \sim \mathcal{T}^2(p, n-2).$$

Proof of Lemma 6. From Lemmas 5 and 1, we can derive the distributions of $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j$ and $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\boldsymbol{\Sigma}^{-\frac{1}{2}}$. These are $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j \sim \mathcal{N}_p(\boldsymbol{0}, \boldsymbol{I}_p)$ and $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\boldsymbol{\Sigma}^{-\frac{1}{2}} \sim \mathcal{W}_p(\boldsymbol{I}_p, n-2)$. Then, Lemma 2 implies that $(n-2)\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j\right)^T\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^{-1}\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j\right)$ is distributed according to Hotelling's $\mathcal{T}^2$ distribution with parameters p and $n-2$.

Moreover, this expression is equal to $\boldsymbol{b}_j^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1} \boldsymbol{b}_j$, as shown below.

$$(n-2)\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j\right)^T \left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}\boldsymbol{\Sigma}^{-\frac{1}{2}}\right)^{-1}\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{b}_j\right) = (n-2)\boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j = \boldsymbol{b}_j^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j$$

□

Lemma 7. *The probability density function of a Hotelling $\mathcal{T}^2$ random variable with parameters a and b is*

$$f_{\mathcal{T}^2}(t^2) = \frac{\Gamma\left(\frac{b-1}{2}\right)}{b\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)}\left(\frac{t^2}{b}\right)^{\frac{a}{2}-1}\left(1+\frac{t^2}{b}\right)^{-\frac{b-1}{2}}.$$

Proof of Lemma 7. If $t^2 \sim \mathcal{T}^2(a,b)$, then $u = \frac{b-a+1}{ab}t^2 \sim \mathcal{F}_{a,b-a+1}$. The probability density function of an $\mathcal{F}$-distributed random variable u with parameters a and $b-a+1$ is given below.

$$\begin{aligned}
f_{\mathcal{F}}(u) &= \frac{1}{\text{Beta}\left(\frac{a}{2},\frac{b-a+1}{2}\right)}\left(\frac{a}{b-a-1}\right)^{\frac{a}{2}} u^{\frac{a}{2}-1}\left(1+\frac{a}{b-a+1}u\right)^{-\frac{a+(b-a-1)}{2}} \\
&= \frac{\Gamma\left(\frac{a}{2}+\frac{b-a-1}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)}\left(\frac{a}{b-a-1}\right)^{\frac{a}{2}} u^{\frac{a}{2}-1}\left(1+\frac{a}{b-a+1}u\right)^{-\frac{b-1}{2}} \\
&= \frac{\Gamma\left(\frac{b-1}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)}\left(\frac{a}{b-a-1}\right)^{\frac{a}{2}} u^{\frac{a}{2}-1}\left(1+\frac{a}{b-a+1}u\right)^{-\frac{b-1}{2}}
\end{aligned}$$

Using the transformation, $u = g^{-1}(t^2) = \frac{b-a+1}{ab}t^2$, we can derive the pdf of t^2, through the change of variables formula, $f_{\mathcal{T}^2}(t^2) = \left|\frac{d}{dt}g^{-1}(t^2)\right| f_{\mathcal{F}}(g^{-1}(t^2))$.

$$
\begin{aligned}
f_{\mathcal{T}^2}(t^2) &= \frac{b-a+1}{ab} f_{\mathcal{F}}\left(\frac{b-a+1}{ab}t^2\right) \\
&= \frac{b-a+1}{ab} \frac{\Gamma\left(\frac{b-1}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)} \left(\frac{a}{b-a-1}\right)^{\frac{a}{2}} \left(\frac{b-a+1}{ab}t^2\right)^{\frac{a}{2}-1} \\
&\quad \times \left(1+\frac{a}{b-a+1}\frac{b-a+1}{ab}t^2\right)^{-\frac{b-1}{2}} \\
&= \frac{b-a+1}{ab} \frac{\Gamma\left(\frac{b-1}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)} \left(\frac{a}{b-a-1}\right) \left(\frac{t^2}{b}\right)^{\frac{a}{2}-1} \left(1+\frac{t^2}{b}\right)^{-\frac{b-1}{2}} \\
&= \frac{\Gamma\left(\frac{b-1}{2}\right)}{b\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b-a-1}{2}\right)} \left(\frac{t^2}{b}\right)^{\frac{a}{2}-1} \left(1+\frac{t^2}{b}\right)^{-\frac{b-1}{2}}
\end{aligned}
$$

□

Lemma 8. *We can express Q_j in terms of the sample Mahalanobis distance of the j^{th} random variable, $\boldsymbol{X}_j$, as follows,*

$$Q_j = 1 - \frac{n}{(n-1)^2}(\boldsymbol{X_j} - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}}^{-1}(\boldsymbol{X_j} - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}).$$

Proof of Lemma 8. Using the final line of the proof for Lemma 4, we can obtain the following expression for Q_j:

$$Q_j = \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus \boldsymbol{X}_j}|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|} = \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}} + (-\boldsymbol{b}_j)\boldsymbol{b}_j^T|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|}$$

By applying the Matrix Determinant Lemma (Lemma 3) to the above equation, we can eliminate the denominator, as follows:

$$Q_j = \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}} + (-\boldsymbol{b}_j)\boldsymbol{b}_j^T|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|} = \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|\left[1 + \boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}^{-1}(-\boldsymbol{b}_j)\right]}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|} = 1 - \boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}^{-1}\boldsymbol{b}_j$$

Substituting in the definitions of $\boldsymbol{b}_j$ and the sample covariance matrix of $\boldsymbol{\mathcal{X}}$, we obtain the

result.

$$
\begin{aligned}
Q_j &= 1 - \boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}^{-1} \boldsymbol{b}_j \\
&= 1 - \left[\sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})\right]^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}^{-1} \left[\sqrt{\frac{n}{n-1}}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})\right] \\
&= 1 - \frac{n}{n-1}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}}^{-1} (\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})
\end{aligned}
$$

$$
\begin{aligned}
Q_j &= 1 - \frac{n}{n-1}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T \left[\frac{1}{n-1}\boldsymbol{S}_{\boldsymbol{\mathcal{X}}}^{-1}\right](\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}}) \\
&= 1 - \frac{n}{(n-1)^2}(\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}}^{-1} (\boldsymbol{X}_j - \bar{\boldsymbol{X}}_{\boldsymbol{\mathcal{X}}})
\end{aligned}
$$

□

A1.5 Theorem Proofs

Proof of Theorem 1. By applying Lemma 4 and Lemma 3 to the definition of Q_j, we can obtain an expression for Q_j in terms of $\boldsymbol{b_j}$ and $\boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X_j}}$ that does not involve determinants.

$$
\begin{aligned}
Q_j &= \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}}|} \\
Q_j &= \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j} + \boldsymbol{b}_j\boldsymbol{b}_j^T|} \\
Q_j &= \frac{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}|}{|\boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}|(1 + \boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j)} \\
Q_j &= \frac{1}{1 + \boldsymbol{b}_j^T \boldsymbol{A}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j} \\
Q_j &= \frac{1}{1 + \frac{1}{n-2}\boldsymbol{b}_j^T \boldsymbol{S}_{\boldsymbol{\mathcal{X}}\setminus\boldsymbol{X}_j}^{-1}\boldsymbol{b}_j} \\
Q_j &= \frac{1}{1 + \frac{1}{n-2}T_j^2}
\end{aligned}
$$

Define a function $g_n : \mathbb{R} \to \mathbb{R}$ such that $g_n(t^2) = \left(1 + \frac{t^2}{n-2}\right)^{-1}$. Define $h_n : \mathbb{R} \to \mathbb{R}$ to be the inverse of g_n, $q \mapsto (n-2)\left(\frac{1}{q} - 1\right)$. Then, $Q_j = \left(1 + \frac{T_j^2}{n-2}\right)^{-1} = g_n(T_j^2)$ and $T_j^2 = (n-2)\left(\frac{1}{Q_j} - 1\right) = h_n(Q_j)$.

By Lemma 6 and 7, we know that the probability density function of T_j^2 is given by the following formula:

$$f_{T_j^2}(t_j^2) = \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} \left(\frac{t_j^2}{n-2}\right)^{\frac{p}{2}-1} \left(1+\frac{t_j^2}{n-2}\right)^{-\frac{n-1}{2}}.$$

Since Q_j is a function of T_j^2, we can derive the probability density function of Q_j as follows:

$$f_{Q_j}(q_j) = f_{T_j^2}(h_n(q_j)) \times \left|\frac{d}{dq_j} h_n(q_j)\right|.$$

$$\begin{aligned}
f_{T_j^2}(h_n(q_j)) &= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} \left(\frac{h_n(q_j)}{n-2}\right)^{\frac{p}{2}-1} \left(1+\frac{h_n(q_j)}{n-2}\right)^{-\frac{n-1}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} \left(\left(\frac{1}{q_j}-1\right)\right)^{\frac{p}{2}-1} \left(1+\left(\frac{1}{q_j}-1\right)\right)^{-\frac{n-1}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} \left(\frac{1}{q_j}-1\right)^{\frac{p}{2}-1} \left(\frac{1}{q_j}\right)^{-\frac{n-1}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} \left(\frac{1-q_j}{q_j}\right)^{\frac{p}{2}-1} (q_j)^{\frac{n-1}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{1-\frac{p}{2}} (q_j)^{\frac{n-1}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{1+\frac{n-1}{2}-\frac{p}{2}} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{\frac{n-p-1}{2}+1}
\end{aligned}$$

$$
\begin{aligned}
f_{Q_j}(q_j) &= \left| \frac{d}{dq_j} h_n(q_j) \right| \times f_{T_j^2}(h_n(q_j)) \\
&= \left[(n-2) q_j^{-2} \right] \times \left[\frac{\Gamma\left(\frac{n-1}{2}\right)}{(n-2)\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{\frac{n-p-1}{2}+1} \right] \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{\frac{n-p-1}{2}+1-2} \\
&= \frac{\Gamma\left(\frac{n-1}{2}\right)}{\Gamma\left(\frac{n-p-1}{2}\right)\Gamma\left(\frac{p}{2}\right)} (1-q_j)^{\frac{p}{2}-1} (q_j)^{\frac{n-p-1}{2}-1}
\end{aligned}
$$

This is the probability density function of a beta random variable with parameters $\frac{p}{2}$ and $\frac{n-p-1}{2}$. □

Proof of Theorem 2. Theorem 2 follows from Theorem 1 and Lemma 8 by the symmetry of the beta distribution. □