Efficient Bayesian methods for mixture models with genetic applications

Daiane Aparecida Zuanetti

Daiane Aparecida Zuanetti

Métodos Bayesianos eficientes para modelos de mistura com aplicações em genética

Tese apresentada ao Departamento de Estatística – DEs-UFSCar e ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Estatística - Programa Interinstitucional de Pós-Graduação em Estatística.

Área de Concentração: Estatística

Orientador: Prof. Dr. Luis Aparecido Milan

USP – São Carlos
Novembro de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

```
       Zuanetti, Daiane Aparecida
Z93e      Efficient Bayesian methods for mixture models
       with genetic applications / Daiane Aparecida
       Zuanetti; orientador Luis Aparecido Milan. -- São
       Carlos, 2016.
          224 p.


          Tese (Doutorado - Programa Interinstitucional de
       Pós-graduação em Estatística) -- Instituto de Ciências
       Matemáticas e de Computação, Universidade de São
       Paulo, 2016.


          1. Mixture models. 2. Bayesian methods. 3. QTL
       mapping. 4. Big data clustering. I. Milan, Luis
       Aparecido, orient. II. Título.
```

Daiane Aparecida Zuanetti

Efficient Bayesian methods for mixture models with genetic applications

Thesis submitted to the Departamento de Estatística – DEs-UFSCar and to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP, in partial fulfillment for the PhD degree in Statistics - Interinstitucional program of graduation in Statistics.

Concentration Area: Statistics

Advisor: Prof. Dr. Luis Aparecido Milan

USP – São Carlos
November 2016

Agradecimentos

Agradeço a Deus que me deu todas as condições necessárias para completar com êxito todas as etapas da minha vida até a conclusão do doutorado. Deus também me presenteou com uma família maravilhosa que me ensinou o significado das palavras: confiança, respeito, humildade, ajuda, incentivo, carinho, amor e responsabilidade. Aos meus pais, Luiz e Sebastiana, irmãos, José Antonio e Patrícia, e sobrinhos Antonia, Gustavo e Betina, qualquer palavra de agradecimento seria insuficiente para expressar a importância de cada um na minha vida. Aqui, basta registrar que eles são os exemplos de vida nos quais me espelho.

Aos orientadores, gestores e demais professores, agradeço por terem sido mediadores do meu conhecimento e terem despertado em mim a busca contínua de desenvolvimento e informações. Agradeço também aos demais familiares e amigos com quem partilhei momentos felizes e que me acompanharam em instantes de crescimento e superação.

Por fim, agradeço à CAPES por ter financiado este projeto de doutorado e meu estágio no exterior.

Resumo

Nós propomos métodos Bayesianos para selecionar e estimar diferentes tipos de modelos de mistura que são amplamente utilizados em Genética e Biologia Molecular. Especificamente, propomos métodos direcionados pelos dados para selecionar e estimar um modelo de mistura generalizado, que descreve o modelo de mistura usual (independente) e o de primeira ordem numa mesma estrutura, e modelos de mapeamento de QTL com dados independentes e familiares. Para agrupar genes através de modelos de mistura, nós propomos três métodos Bayesianos não-paramétricos: o processo de Dirichlet aninhado que possibilita agrupamento de distribuições e, um algoritmo preditivo recursivo e outro Bayesiano não-paramétrico exato para agrupar dados de alta dimensão. Analisamos e comparamos o desempenho dos métodos propostos e dos procedimentos tradicionais de seleção e estimação de modelos e agrupamento de dados em conjuntos de dados simulados e reais. Os métodos propostos são mais flexíveis, aprimoram a convergência dos algoritmos e apresentam estimativas mais precisas em muitas situações. Além disso, nós propomos procedimentos para estimar o genótipo não observável dos QTLs e de pais faltantes e melhorar a probabilidade Mendeliana de herança genética do genótipo dos descendentes através da estrutura condicional de independência entre as variáveis. Também sugerimos aplicar medidas de diagnóstico para verificar a qualidade do ajuste dos modelos de mapeamento de QTLs.

Palavras-chave: Modelos de mistura; métodos Bayesianos direcionados pelos dados; métodos Bayesianos não-paramétricos; mapeamento de QTL; agrupamento de distribuições; agrupamento de dados de alta dimensão; dados em família; interação de genes.

Abstract

We propose Bayesian methods for selecting and estimating different types of mixture models which are widely used in Genetics and Molecular Biology. We specifically propose data-driven selection and estimation methods for a generalized mixture model, which accommodates the usual (independent) and the first-order (dependent) models in one framework, and QTL (quantitative trait *locus*) mapping models for independent and pedigree data. For clustering genes through a mixture model, we propose three nonparametric Bayesian methods: a marginal nested Dirichlet process (NDP), which is able to cluster distributions and, a predictive recursion clustering scheme (PRC) and a subset nonparametric Bayesian (SNOB) clustering algorithm for clustering big data. We analyze and compare the performance of the proposed methods and traditional procedures of selection, estimation and clustering in simulated and real data sets. The proposed methods are more flexible, improve the convergence of the algorithms and provide more accurate estimates in many situations. In addition, we propose methods for estimating nonobservable QTLs genotypes and missing parents and improve the Mendelian probability of inheritance of nonfounder genotype using conditional independence structures. We also suggest applying diagnostic measures to check the goodness of fit of QTL mapping models.

Keywords: Mixture models; data-driven Bayesian methods; nonparametric Bayesian methods; QTL mapping; clustering distributions; clustering big data; pedigree data; gene interaction.

Contents

List of Figures **xiv**

List of Tables **xvi**

1 Introduction **1**

1.1 **Genetic background** 2

1.2 **Proposals and structure** 3

2 Generalized mixture model **6**

2.1 **Introduction** 6

2.2 **Generalized mixture model** 8

2.3 **Bayesian approach** 10

2.3.1 Updating the number of components K 11

2.3.2 Transition between independent and first-order mixture models 14

2.3.3 Algorithm 15

2.4 **Mixture of dependent binomial distributions** 16

2.5 **Applications** 18

2.5.1 Simulated data sets 19

2.5.2 Rolling thumbtacks 21

2.5.3 Number of diagnosed cases of diabetes 22

2.6 **Use of the first-order mixture model in Genetics and Molecular Biology** 24

2.7 **Discussion** 25

2.8 **Appendices** 26

2.8.1 Validity of split-merge acceptance probability 26

2.8.2 Validity of dependence order acceptance probability 26

2.8.3 Additional mixing and statistical convergence diagnostics of simulated data sets 26

2.8.4 Generalized mixture of multinomial distributions 27

2.8.5 R codes to carry out DDRJ 30

3 QTL mapping as a mixture model **46**

3.1 **Introduction** 46

3.2 **Model for quantitative traits** 48

3.3 **Bayesian approach** 50

3.3.1 DDRJ 52

3.4 **Applications** 58

3.4.1 Simulated data sets 58

3.4.2 Bone mineral density data set 62

3.5 **Discussion** 64

3.6 **Appendices** 65

3.6.1 Conditional *a posteriori* distribution of parameters 65

3.6.2 Additional information about the RJ and DDRJ chains of simulated data sets 68

3.6.3 R codes to carry out DDRJ 69

4 QTL mapping model checking **91**

4.1 **Introduction** 91

4.2 **Bayesian model checking** 92

4.3 **Checking the bone mineral density QTL mapping model** 94

4.4 **Discussion** 97

5 A model for QTL mapping of pedigree data **99**

5.1 **Introduction** 99

5.2 **Model for quantitative traits of pedigree data** 101

5.2.1 Transmission probabilities 104

5.2.2 Missing founder's parents 107

5.3 **Bayesian approach** 108

5.3.1 Algorithm DDRJ for pedigree data 110

5.4 **Applications** 111

5.5 **Discussion** 114

5.6 **Appendices** 114

5.6.1 R codes to carry out DDRJ 114

5.6.2 Codes used to simulate data sets in SimPed 140

6 A marginal NDP – clustering distributions **142**

6.1 **Introduction** 142

6.2 **Nested Dirichlet process** 144

6.3 **The *a posteriori* simulation for the marginal NDP** 146

6.3.1 Gibbs sampling transition probabilities 146

6.3.2 Transition probabilities for distributional clusters 147

6.4 **Simulation** 150

6.4.1 Marginal NDP 150

6.4.2 Clustering distributions by k-means 151

6.5 **Clustering DMR genes** 154

6.5.1 Data 154

6.5.2 Results 154

6.6 **Discussion** 157

6.7 **Appendices** 157

6.7.1 R codes to carry out marginal NDP in simulated data set 157

7 Big data clustering using mixture model 172

7.1 **Introduction** 173

7.1.1 Clustering methods 173

7.1.2 Big data clustering 174

7.2 **Predictive recursion clustering (PRC)** 176

7.2.1 Predictive recursion 176

7.2.2 Merging similar components and removing order dependence 178

7.2.3 PRC algorithm 179

7.3 **Subset nonparametric Bayesian (SNOB)** 180

7.3.1 Clustering each shard and estimating local clusters 181

7.3.2 Estimating global clusters 183

7.4 **Simulation** 185

7.5 **Clustering genes by their GE-GE interactions** 188

7.5.1 Data 188

7.5.2 PRC results 188

7.5.3 SNOB results 189

7.6 **Discussion** 190

7.7 **Appendices** 192

7.7.1 R codes to carry out PRC for simulated data set 192

7.7.2 R codes to carry out SNOB for simulated data set 200

8 Conclusions 214

Bibliography 216

List of Figures

2.1 Number of diagnosed cases of diabetes by ages (per 1000 population). 23

2.2 The *a posteriori* probability estimate of $P(S_t = 1|\mathbf{y}, \mathbf{s}_{-t}, \boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)$ 25

3.1 Trace plot of K for $\sigma = 0.5$: (A) RJ sequence and (B) DDRJ sequence. 59

3.2 Trace plot of K for $\sigma = 1.0$: (A) RJ sequence and (B) DDRJ sequence. 59

3.3 Trace plot of K for $\sigma = 1.5$: (A) RJ sequence and (B) DDRJ sequence. 59

4.1 Diagnostic measures of goodness of fit: (A) normal probabilistic plot of *a posteriori* mean of studentized residuals; (B) the *a posteriori* distribution of studentized residuals versus iterations plot; (C) the *a posteriori* distribution of studentized residuals versus predicted values plot; (C) index plot of ICPO; (D) index plot of global influence; (E) index plot of normalized importance weight variance. 95

4.2 Diagnostic measures of goodness of fit: (A) normal probabilistic plot of *a posteriori* mean of studentized residuals; (B) the *a posteriori* distribution of studentized residuals versus iterations plot; (C) the *a posteriori* distribution of studentized residuals versus predicted values plot; (C) index plot of ICPO; (D) index plot of global influence; (E) index plot of normalized importance weight variance. 98

5.1 Pedigree represented by a DAG. 102

5.2 DAG for a family of a pair of parents and their offspring. 103

5.3 Position (M) of the first estimated QTL when $\sigma = 2$. 113

6.1 Simulation truth. 150

6.2 Panel (A) shows the simulation truth for cluster membership by plotting $I(s_i = s_j)$ (black for equality). Panel (B) plots the *a posteriori* probabilities $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$ (black for $\bar{p}_{ij} = 1$, white for 0). 152

6.3 Estimated $\widehat{F}_k = E(F_k \mid \boldsymbol{y})$. Compare with Figure 6.1. 152

6.4 Histograms of GE-GE interactions for four genes. The inference goal is to group all J genes into clusters with similar distributions of GE-GE interactions. 154

6.5 The *a posteriori* co-clustering probability $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$ for DMR genes. Numbers under each cluster in the diagonal represent clusters' label. 155

6.6 Estimated cluster-specific distributions $\widehat{F}_k = E(F_k \mid \boldsymbol{y})$. 156

7.1 PRC clusters. Each boxplot shows the distribution of the respective quantile across all genes in the cluster. 190

7.2 SNOB clusters. Each boxplot shows the distributions of the respective quantile across all genes in the cluster. 191

List of Tables

2.1 Effective sample size of K sequence in DDRJ and RJ chains. 20

2.2 DDRJ *a posteriori* probability for dependence structure of the model. The highest probability of each situation is in boldface type. 20

2.3 DDRJ *a posteriori* probability for the number of components K. Higher probabilities are in boldface type. 21

2.4 RJ *a posteriori* probability for the number of components K. Higher probabilities are in boldface type. 21

2.5 DDRJ and RJ *a posteriori* probability for the number of components K. 22

2.6 The *a posteriori* probability for the number of components K. 24

2.7 Estimates and 95% credibility intervals for diagnosed diabetes rates. 24

2.8 IAT of K sequence in DDRJ and RJ chains. 27

2.9 Acceptance rate of split-merge proposals. 27

2.10 Acceptance rate of dependence order proposals in DDRJ chains. 27

2.11 The *a posteriori* probability for dependence structure of the model obtained by DDRJ. 28

2.12 The *a posteriori* probability for the number of components K obtained by DDRJ. 29

2.13 Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 50$ and $\mathbf{P}_1$. 29

2.14 Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 50$ and $\mathbf{P}_3$. 29

2.15 Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 100$ and $\mathbf{P}_1$. 30

2.16 Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 100$ and $\mathbf{P}_3$. 30

3.1 ESS of K sequences. 59

3.2 The *a posteriori* probability for K. 60

3.3 The *a posteriori* estimates of the models parameters. 61

3.4 MIM estimates of the parameters. 61

3.5 DDRJ *a posteriori* probability for K in each chromosome. 63

3.6 DDRJ estimates and 95% credibility intervals of parameters. 64

3.7 Effective sample size of RJ and DDRJ chains. 69

4.1 Point estimates and 95% credibility intervals for parameters. 94

4.2 The *a posteriori* probabilities for K in each chromosome without atypical observations. 96

4.3 Estimates and 95% credibility interval for QTLs' location without atypical observations. 96

4.4 Estimates and 95% credibility interval for parameters of model with 5 QTLs and without outliers. 97

5.1 Values of $s^*_{p_{ik}}$ for each combination of $l_{p_{ik}}$, $l_{p_{p_ik}}$ and $l_{m_{p_ik}}$ considering heterozygous father. 105

5.2 Transmission probabilities when $r_t \approx 1/2 \approx r_{t+1}$. 106

5.3 The *a posteriori* probability for K. 112

5.5 Estimates and standard errors of variance components models. 113

5.4 The *a posteriori* estimates for models. 114

6.1 Parameters of the true distributions used to simulate the data set, where w represents the component weight. 150

6.2 Misclassification table of true and estimated clusters of $\tilde{\boldsymbol{y}}$. 153

7.1 Misclassification rate of clustering. 187

7.2 Frequency of the most 6 representative PRC clusters. 189

7.3 Frequency of the most 12 representative SNOB clusters. 190

Chapter 1

Introduction

Mixture models have been applied in many research areas since they describe data coming from a mixture of subpopulations that cannot be adequately modeled by any standard parametric family of distributions. In Genetics and Molecular Biology, specifically, mixture models generalize some important models used for identifying groups of homogeneous segments, genes or proteins, mapping quantitative trait *locus* (QTL) or clustering genes.

The hidden Markov model (HMM), for example, has been widely used to describe homogeneous segments (Boys & Henderson, 2002, 2004; Boys *et al*, 2000; Churchill, 1989, 1992; Muri, 1998; Zuanetti, 2006), detect and align remotely homologous sequences that provide information about the protein's function, structure or evolution (Gough *et al*, 2001; Lee *et al.* , 2009; Söding 2005) and impute missing genotype of QTLs or single nucleotide polymorphism (SNPs) in DNA sequencing (Broman, 2006; Druet & Georges, 2010; Kang *et al*, 2010). The HMM can be written as a mixture model with first-order dependence and the usual independent mixture model is a particular case of the first-order mixture model (Meira, 2014).

QTL mapping models, used for locating regions associated with quantitative traits in the genome, can also be characterized as an independent or dependent mixture model according to the relative relationship among individuals. A phenotype is usually modeled as a linear function of the additive and dominance effects of the QTL genotypes and several methods have been developed to estimate the position and characterization of QTLs.

In addition, mixture models are used for clustering observations. Fraley & Raftery (2002, 2007); Fraley *et al.* (2012) describe and review methodological framework for data clustering using mixture models. However, most of the model-based clustering methods or deterministic clustering schemes fail for big data, due to computational constraints and the need to access all data simultaneously.

The relevant issues in estimating mixture models are identifying the number of components, estimation of parameters and clustering the observations according to the component they belong. Therefore, Bayesian inference arises as an attractive alternative since it combines parameters estimation, model selection and clustering in a powerful way and carry out them jointly. The reversible jump (RJ) (Green, 1995; Green & Richardson, 2001; Richardson & Green, 1997) is the usual Markov chain Monte Carlo (MCMC) method for selecting and estimating a mixture model and clustering the observations.

The most important characteristic in the MCMC is that it mixes well, *i.e.*, that it moves around the possible models and associated parameter spaces rather easily, and quickly finds its stationary distribution. Improving the existing MCMC estimation methods to obtain better mixing, accuracy, faster convergence and capability to deal with big data has been a topic of research for many authors.

1.1 Genetic background

Geneticists and molecular biologists have aimed at locating regions associated with quantitative traits (or phenotype as, for example, eye color, weight, affected/normal, etc) in the genome. These chromosomal regions are known as QTLs and the QTL mapping model has been an important method for identifying genetic causes of diseases or other characteristics of living beings.

For a given individual, one chromosome in each pair derives from the DNA of his mother and the other from the DNA of his father. A specific segment of chromosome is known as a *locus* and we typically refer to the individual's DNA at this *locus* as its gene. Different variants of a gene are called alleles and pair of alleles (AA, Aa or aa) is referenced as the genotype. If both alleles are of the same type, we say the genotype is homozygous otherwise heterozygous.

QTLs effects may be divided into three components:

- additive effect: results from the direct action of each allele in the chromosomes;
- dominance effect: results from the combined action of alleles in chromosomes in the same *locus* (intra-*locus* interaction); and
- epistasis: results from the combined action of alleles at different *loci* (inter-*locus* interaction).

As the location in the genome of a QTL is unknown, we can use the genetic markers for identifying and mapping QTLs' locations. The location of markers are known and specified through the linkage map. The genotype of genetic markers is also known.

In a genetic linkage map, the distance between two *loci* is estimated by the recombination fraction between them. The recombination fraction r between two *loci* is defined as the probability that genes segregating to the gamete at these *loci* come from different parental chromosomes. For *loci* close together, r is approximately 0 and, when *loci* are far away, r tends to be 1/2, indicating that the *loci* are segregating independently. That is, under assumptions of the meiosis model for most diploid species, r ranges from 0 to 1/2.

There are several link functions between the genetic distance and the recombination fraction. Among them, stands out the Haldane function (Haldane, 1919) defined as

$$r = \tfrac{1-\exp(-2d_H)}{2} \text{ or } d_H = -\tfrac{1}{2}\ln(1-2r),$$

where $d_H \longrightarrow \infty$ when $r \longrightarrow 0.5$. The distance measure is usually specified in centiMorgans (cM) or Morgans (M).

1.2 Proposals and structure

In this thesis, the main subject of study is the mixture models. We consider different types of mixture models, all of them widely used in Genetics and Molecular Biology, and propose more efficient Bayesian methods for selecting and estimating the best fitting model and clustering the observations.

First, we describe a generalized mixture model which accommodates the usual (independent) and first-order (dependent) models in one framework and propose an efficient and accurate data-driven reversible jump (DDRJ) to implement the model selection and model fitting. The procedure is able to select between independent and first-order mixture models and estimates the number of components in the mixture. The update of the number of components is made by splitting and merging moves avoiding arbitrary transformations of current parameters. This simplifies the methodology and accelerates the search procedure of the best fitting model, since more suitable and efficient candidates for changing the dimension are generated using the data.

We also propose a birth-death-merge DDRJ for multiple QTL mapping. It simulates a more likely location for a new QTL using the available data, chooses a QTL to be excluded according to its importance in the current model or merges the effects of two consecutive QTLs if their genotype are correlated. Consequently, candidates are more likely to be accepted and the space of possible models are easily explored. The merge movement of consecutive QTLs is efficient under tested conditions to avoid identification of false QTLs. In addition, we briefly describe some Bayesian statistics used for model checking and propose using them to check the goodness of fit of a QTL mapping model.

Considering QTL mapping model for pedigree data, we suggest a model which describes the dependence structure among individuals and variables through conditional independence structures. The proposed model considers that the segregation of a gene in a *locus* depends on Mendelian segregation and also is correlated with genotypes of flanking markers (at left and at right). The assumption of linked genotypes of nearby *loci* is desirable because the chance of a genetic recombination through meiosis process is low and the genotype of flanking *loci* is informative and improves the model which considers sole the Mendel probability of inheritance for parents' genotypes. We also extend the DDRJ for mapping QTLs of independent individuals to estimate the number, positions, additive and dominance effects of QTLs in pedigree data.

The proposed DDRJ methods also have the advantage of providing intervalar estimates with information about the uncertainty of estimates. Usual methods generally provide only point estimates or asymptotic confidence intervals for big samples.

We also discuss clustering for distributions of gene-gene interactions through mixture models. The aim is inference on groups of genes that are similar in terms of the distribution of their interactions with other genes. We use a nested Dirichlet process *a priori* (NDP) (Rodriguez *et al*, 2008) for the desired grouping of distributions. We show how inference in the NDP can be implemented exactly, rather than the approximate inference based on finite truncations that is used in the original NDP. The proposed MCMC scheme clarifies the nature of the NDP as a prior for nested clustering.

For clustering big data, we propose two nonparametric mixture model-based methods. The first scheme is an incremental clustering algorithm called predictive recursion clustering (PRC). It is based on a predictive recursion algorithm by Newton *et al.* (1998), which is usually applied to approximate the *a posteriori* mean of a DP (Dirichlet process) mixture model. We use the terms of the mixture approximation to define clusters and add a step to allocate observations to the cluster that maximizes the marginal *a posteriori* after all observations have been included in the model. It is an approximate method that avoids a full MCMC *a posteriori* simulation. The second method is a distributed computing algorithm called subset nonparametric Bayesian (SNOB) clustering. It is an exact method that divides the data into smaller groups, referred to as shards, across multiple machines and identifies local clusters. In a second step, we combine the local clusters to determine global clusters in a MapReduce or Hadoop framework. The method is simulation exact, in the sense that the global clusters are built on the basis of a probability model for the full original data.

This thesis is organized as follows. The first section of each chapter shows a bibliographic review and discuss in more details the problem studied in the specific chapter. Chapter 2 describes a generalized mixture model, details the Bayesian methodology for this model and proposes the data-driven procedure for selecting and estimating it. This chapter also specifies the methodology for a binomial mixture model and shows RJ and DDRJ performance on simulated and real data sets. Chapter 3 writes a model for independent quantitative traits as a mixture model and proposes a specific DDRJ for model selection and estimation. It also analyzes the performance of the DDRJ and compare it with the RJ performance in simulated and real data sets. In Chapter 4, we describe some Bayesian diagnostic measures for evaluating the goodness of fit and show how to use them in a QTL mapping problem. We apply the suggested measures to evaluate an estimated QTL mapping model for a real data set. Chapter 5 proposes a model for quantitative traits in pedigree data using the conditional independence structure among variables and individuals. This chapter also analyzes the performance of the method and compares it with the usual mixed variance component model performance in simulated data sets. In Chapter 6, we briefly define the NDP and propose the MCMC algorithm to estimate a marginal NDP. We present a simulated example to illustrate the performance of NDP in clustering histograms with same features of GE coefficients distributions. We apply the marginal NDP to cluster the coefficients distributions of DNA mismatch repair (DMR) genes and compare the NDP results with results under a k-means method, which is a widely used deterministic method for clustering. In Chapter 7, we discuss the predictive recursion algorithm of Newton *et al.* (1998) and develop the PRC method for clustering big data. We also proposes the SNOB method and applies both methods in two benchmark data sets and one simulated data set to explore the performance of both schemes in clustering big data. We compare their performance with DP-means, DBSCAN, SUGS and EM clustering. In addition, we cluster the gene data set using the two proposed methods. Finally, we show conclusions in Chapter 8.

CHAPTER

2

Generalized mixture model [1]

In this chapter, we present a generalization of the usual (independent) mixture model to accommodate a Markovian first-order mixing distribution. We propose the data-driven reversible jump, a Markov chain Monte Carlo (MCMC) procedure, for estimating the *a posteriori* probability for each possible model in a model selection procedure beyond estimating the corresponding parameters.

Simulated data sets modeling through the proposed method shows excellent performance in the convergence of the MCMC, model selection and precision of parameters estimates. The proposed method is also easier to implement as it is not necessary to define arbitrary deterministic transformations to perform transdimensional moves as in the case of usual reversible jump. We apply the proposed method to analyze USA diabetes incidence data sets.

2.1 **Introduction**

One of the challenges in Epidemiology is to study the disease occurrence patterns and evolution trying to understand the differences between different periods of time or groups of people and identify causes and work with preventive policies. In situations of heterogeneous data sets, mixture models have been applied to classification, clustering and describe differences in trajectory among subgroups of independent and longitudinal data.

Two relevant issues in estimating mixture models are identifying the number of components and the estimation of parameters. The identification of the number of components is basically a model selection issue. The usual methods to select the number of components are the expectation-maximization (EM) algorithm (Dempster *et al*, 1977) combined with a model

[1] This chapter is based on the manuscript "A generalized mixture model applied to diabetes incidence data" accepted for publication (Zuanetti & Milan, To appear b).

selection criterion such as Akaike's information criterion (AIC), Bayesian information criterion (BIC), among others (McLachlan & Peel, 2004) and Bayesian methods combined with estimating procedures such as the reversible jump (RJ) (Green, 1995; Green & Richardson, 2001; Richardson & Green, 1997).

Both EM and RJ usually show convergence and accuracy problems. The EM algorithm shows dependence of the final solution on the starting values and convergence to a local maximum and the RJ may have difficulties moving between different models.

Improving the existing Markov chain Monte Carlo (MCMC) estimation methods to obtain more accuracy and faster convergence has been a topic of research for many authors. Tierney & Mira (1999) modify the basic Metropolis-Hastings algorithm allowing a second attempt to update the current state in case of rejection of the first proposal. Green & Mira (2001) extend this strategy to RJ. The evaluation of the acceptance probability to ensure the reversibility of the process is also an obstacle to making complete use of the idea in terms of both efficiency and range of problems to which it can be applied. Brooks *et al.* (2003) propose procedures to draw the random variables which complete the parametric space in RJ and improve the acceptance probability. Pandolfi *et al.* (2014) propose a generalization of the multiple-try Metropolis algorithm which consists of drawing several proposals at each step and randomly choosing one of them on the basis of weights that may be arbitrarily chosen. All these proposals refer to the usual (independent) mixture model.

Fan *et al.* (2009) use a marginal density estimator to settle the distribution from which the between-model proposals are drawn in an RJ algorithm. They avoid difficulties in specifying the deterministic functions and the associated proposal distribution, but the resulting class of proposal distributions requires moderate computational effort. Jain & Neal (2004), Jain & Neal (2007) and Saraiva & Milan (2012) use data-driven MCMC methods to estimate a mixture model considering independent variables and obtain better convergence and accuracy. By data-driven MCMC, we mean that the observed data is used to define the proposal distribution (Jain & Neal, 2004; Saraiva & Milan, 2012).

First-order mixture models, also known as hidden Markov models (HMMs; Rabiner, 1989; Rabiner & Juang 1986; Zucchini & MacDonald, 2009), have been applied to study longitudinal data sets mainly to identify homogeneous segments in DNA sequencing (Boys & Henderson , 2002, 2004; Boys *et al*, 2000; Muri, 1998), as well as detect and align multiple DNA sequences providing information about a protein function, structure or evolution (Gough *et al*, 2001; Lee *et al*, 2009; Söding 2005) and disease progression (Jackson *et al*, 2003).

Considering a first-order mixture model, Visser *et al.* (2010) propose estimating the model using EM and the forward-backward algorithm with a fixed number of components and selecting the number of components using model selection criteria. Robert *et al.* (2000) propose selecting and estimating a first-order mixture model jointly through RJ. Despite testing different deterministic functions and hyperparameter values, they note that the mixing of MCMC is not as good as expected. Shi *et al.* (2002) propose a birth-death MCMC to estimate first-order models where parameters associated with the new component are drawn from an arbitrary density and then the observations are reallocated. Spezia (2010) estimates the number of components through RJ and then runs another MCMC with a fixed number of components to estimate the parameters.

We describe a generalized mixture model which accommodates the usual (independent) and the first-order (dependent) models in one framework and propose an efficient and accurate data-driven reversible jump (DDRJ) to implement the model selection and model fitting. The procedure is able to select between independent and first-order mixture models and estimates the number of components in the mixture. The method may also identify the best fitting distribution if it is required although this is not our focus here. The update of the number of components is made by splitting and merging moves avoiding arbitrary transformations of parameters. Finally, we apply the DDRJ to diabetes data incidence and verify possible changes in the diabetes incidence over the last three decades. We also compare the trajectory of age groups.

The chapter is organized as follows: Section 2.2 describes the first-order mixture model and independent model in one framework; Section 2.3 proposes the data-driven MCMC algorithm; Section 2.4 specifies the methodology for a mixture of binomial distributions directing it to epidemiological data; and in Section 2.5, the DDRJ is applied to the simulated data sets comparing its performance with the RJ. It is also applied to data from Beckett and Diaconis (1994) and USA incidences of diabetes data; Section 2.6 shows the use of the first-order mixture model in Genetics and Molecular Biology. Finally, we conclude with a discussion in Section 2.7 and appendices in Section 2.8.

2.2 Generalized mixture model

In this section, we briefly describe the generalized mixture model which accommodates the independent and first-order mixture models (HMMs) in the same framework.

Let $\mathbf{S} = (S_1, S_2, ..., S_T)$ be a Markov chain, where $S_t \in \{1, 2, ..., K\}$ for $t = 1, 2, ..., T$,

$$Pr\left(S_t = s_t | S_{t-1} = s_{t-1}, ..., S_1 = s_1\right) = Pr\left(S_t = s_t | S_{t-1} = s_{t-1}\right) = p_{s_{t-1}s_t}$$

for $t = 2, 3, ..., T$ and $Pr\left(S_1 = s_1\right) = p_{0s_1}$. Let $\mathbf{Y} = (Y_1, Y_2, ..., Y_T)$ be a sequence of random variables with density given by $f_{Y_t|S_t=k}(y_t) = f_{Y_t}(y_t|\boldsymbol{\theta}_k)$ with $k \in \{1, 2, ..., K\}$ and $\boldsymbol{\theta}_k$ the parameters of the probability distribution associated to the k-th state of S_t.

The model has the following conditional independence structures:

$$S_t \perp \{S_1, Y_1, ..., S_{t-2}, Y_{t-2}, Y_{t-1}\}|S_{t-1} \text{ and } Y_t \perp \{S_1, Y_1, ..., S_{t-1}, Y_{t-1}\}|S_t$$

for $t = 2, 3, ..., T$.

For a fixed value K, the parameters of the model are

1. $\mathbf{p}_0 = (p_{01}, ..., p_{0K})$ the initial probability for $\mathbf{S}$, where $p_{0k} = Pr(S_1 = k)$, $k = 1, 2, .., K$;

2. $\mathbf{P} = \{p_{jk}\}$ the transition matrix of $\mathbf{S}$, where $p_{jk} = Pr(S_{t+1} = k|S_t = j)$, $j, k = 1, 2, ..., K$; and

3. $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K)$ the parameters of the distribution associated to each state of S_t.

The joint distribution of $\mathbf{Y}$ and $\mathbf{S}$ is

$$f_{\mathbf{Y},\mathbf{S}}(\mathbf{y}, \mathbf{s}|\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}) = \prod_{t=1}^{T} p_{s_{t-1}s_t} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right), \tag{2.1}$$

where $p_{s_0s_1} = p_{0s_1}$.

As $\mathbf{S}$ is nonobservable, the marginal distribution of $\mathbf{Y}$ is

$$\begin{aligned} f_{\mathbf{Y}}(\mathbf{y}|\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}) &= \sum_{\mathbf{s}} f_{\mathbf{Y},\mathbf{S}}(\mathbf{y}, \mathbf{s}|\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}) = \sum_{s_1=1}^{K} \cdots \sum_{s_T=1}^{K} \prod_{t=1}^{T} p_{s_{t-1}s_t} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right) \\ &= \sum_{s_1=1}^{K} p_{s_0s_1} f_{Y_1}\left(y_1|\boldsymbol{\theta}_{s_1}\right) \ldots \sum_{s_T=1}^{K} p_{s_{T-1}s_T} f_{Y_T}\left(y_T|\boldsymbol{\theta}_{s_T}\right) \\ &= \prod_{t=1}^{T} \sum_{s_t=1}^{K} p_{s_{t-1}s_t} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right) \end{aligned} \tag{2.2}$$

which characterizes each variable Y_t as a mixture of K distributions (Meira, 2014).

If $\mathbf{S} = (S_1, S_2, ..., S_T)$ is a sequence of independent variables,

$$Pr\left(S_t = s_t|S_{t-1} = s_{t-1}\right) = Pr\left(S_t = s_t\right) = p_{s_t}$$

where $t = 1, ..., T$ and $\sum_{s_t=1}^{K} p_{s_t} = 1$. The marginal distribution of $\mathbf{Y}$ is

$$\begin{aligned} f_{\mathbf{Y}}(\mathbf{y}|\boldsymbol{\theta}, \mathbf{p}) &= \sum_{\mathbf{s}} f_{\mathbf{Y},\mathbf{S}}(\mathbf{y}, \mathbf{s}) = \sum_{s_1=1}^{K} \cdots \sum_{s_T=1}^{K} \prod_{t=1}^{T} p_{s_t} f_{Y_t}\left(y_t|\theta_{s_t}\right) \\ &= \prod_{t=1}^{T} \sum_{s_t=1}^{K} p_{s_t} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right) \end{aligned} \tag{2.3}$$

which is the independent mixture model, a special case of the first-order mixture model (2.2) (Meira, 2014).

The likelihood function for $\boldsymbol{\theta}$, $\mathbf{P}$ and $\mathbf{p}_0$ is

$$L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}|\mathbf{y}, \mathbf{s}) = \prod_{t=1}^{T} p_{s_{t-1}s_t} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right) = \prod_{k=1}^{K} \left\{ \left(\prod_{j=0}^{K} p_{jk}^{n_{jk}} \right) \left(\prod_{t=1}^{T} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_t}\right)^{I(S_t=k)} \right) \right\}, \tag{2.4}$$

where $n_{0k} = I(S_1 = k)$ and $n_{jk} = \sum_{t=1}^{T-1} I(S_t = j, S_{t+1} = k)$ is the observed number of transitions from component j to component k for $j, k = 1, ..., K$.

2.3 Bayesian approach

Considering the number of components K unknown and $\mathbf{p}_j$'s and $\boldsymbol{\theta}_k$'s independent, for $j = 0, 1, ..., K$ and $k = 1, ..., K$, the joint *a priori* distribution for parameters $\boldsymbol{\theta}$, $\mathbf{p}_0$, $\mathbf{P}$ and K is

$$\begin{aligned}\pi(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K) &= \pi(K)\pi(\mathbf{p}_0|K)\pi(\mathbf{P}|K)\pi(\boldsymbol{\theta}|K) \\ &= \pi(K)\left(\prod_{k=0}^{K} \pi(\mathbf{p}_k|K)\right)\left(\prod_{k=1}^{K} \pi(\boldsymbol{\theta}_k)\right). \end{aligned} \tag{2.5}$$

Assume

1. $K \sim$ Uniform$(1, 2, ..., K_{max})$, where K_{max} is a specified maximum value of K;
2. $\mathbf{p}_j|K \sim$ Dirichlet$(\gamma_{j1}, ..., \gamma_{jK})$ for $j = 0, ..., K$ and where $\gamma_{jK} > 0$ are known hyperparameters; and
3. $\boldsymbol{\theta}_k$, for $k = 1, ..., K$, are independent *a priori* and $\pi(\boldsymbol{\theta}|K) = \prod_{j=1}^{K} \pi(\boldsymbol{\theta}_k)$.

We propose a data-driven MCMC scheme to select and estimate model (2.1). The DDRJ is implemented by three steps: in the first step, the current values of parameters are updated using a Gibbs sampling procedure; in the second step, the number of components of the mixture model is updated using a split-merge move; and in the third step, we evaluate the transition between independent and first-order dependent models. These steps are described as follows.

Combining the likelihood function in eq. (2.4) and the *a priori* distribution, we obtain the conditional *a posteriori* distributions from which the basic Gibbs sampling moves are implemented.

The parameters $\mathbf{p}_0$ and $\mathbf{p}_k$ are updated from the conditional distributions

$$\mathbf{p}_0|(K, \mathbf{y}, \mathbf{s}, \mathbf{P}, \boldsymbol{\theta}) \sim \text{Dirichlet}(\gamma_{01} + n_{01}, ..., \gamma_{0K} + n_{0K}) \text{ and}$$

$$\mathbf{p}_k|(K, \mathbf{y}, \mathbf{s}, \mathbf{P}_{-\mathbf{p}_k}, \mathbf{p}_0, \boldsymbol{\theta}) \sim \text{Dirichlet}(\gamma_{k1} + n_{k1}, ..., \gamma_{kK} + n_{kK}),$$

for $k = 1, ..., K$.

S_t's are simulated from their conditional *a posteriori* distribution given by

$$\begin{aligned} Pr(S_t = k|\mathbf{y}, \mathbf{S}_{-t} = \mathbf{s}_{-t}, \cdots) &= Pr(S_t = k|Y_t = y_t, S_{t-1} = s_{t-1}, S_{t+1} = s_{t+1}, \cdots) \\ &= \frac{Pr(S_t = k, Y_t = y_t, S_{t+1} = s_{t+1}|S_{t-1} = s_{t-1}, \cdots)}{Pr(Y_t = y_t, S_{t+1} = s_{t+1}|S_{t-1} = s_{t-1}, \cdots)} \\ &= \frac{p_{s_{t-1}k} f_{Y_t}(y_t|\boldsymbol{\theta}_k) p_{ks_{t+1}}}{\sum_{k=1}^{K} p_{s_{t-1}k} f_{Y_t}(y_t|\boldsymbol{\theta}_k) p_{ks_{t+1}}}, \end{aligned} \tag{2.6}$$

where $\cdots$ represents $(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)$, $p_{s_0 k} = p_{0k}$, $p_{k S_{T+1}} = 1$ and $f_{Y_t}(y_t|\boldsymbol{\theta}_k)$ is the density of Y_t associated to the k-th component of S_t. Therefore, $S_t|(\mathbf{y}, \mathbf{s}_{-t}, \cdots) \sim$ multinomial$(1, (\delta_{t1}, ..., \delta_{tK}))$, where $\delta_{tk} = Pr(S_t = k|\mathbf{y}, \mathbf{s}_{-t}, \cdots)$.

2.3.1 Updating the number of components K

Transdimensional moves to update the number of components are implemented by split and merge procedures. The split breaks a component into two components increasing K by one and the merge joins two components decreasing K by one unity.

Let $x = (K, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta})$ be the current state with K components and $x^* = (K^*, \mathbf{P}^*, \mathbf{p}_0^*, \boldsymbol{\theta}^*)$ the state of the proposed movement where signal $*$ denotes either a split or a merge. The proposed movement is accepted according to Metropolis-Hastings probability $\Psi(x^*|x) = min(1, A^*)$, where

$$A^* = \frac{L(\boldsymbol{\theta}^*, \mathbf{p}_0^*, \mathbf{P}^*, K^*|\mathbf{y}, \mathbf{s}^*)}{L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s})} \frac{\pi(\boldsymbol{\theta}^*, \mathbf{p}_0^*, \mathbf{P}^*, K^*)}{\pi(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)} \frac{q(x|x^*)}{q(x^*|x)}, \tag{2.7}$$

and $q(\cdot|\cdot)$ is the proposal distribution from which we draw the proposed movement.

The procedure of choosing a split or a merge move is described by the following two steps:

1. Choose a pair of observations, t_1 and t_2, using the discrete uniform$(1, ..., T)$ distribution without replacement.

2. If $s_{t_1} = s_{t_2}$, both observations belong to the same component and a split is proposed. Otherwise, if $s_{t_1} \neq s_{t_2}$, a merge of the components indicated by s_{t_1} and s_{t_2} is proposed.

The probability of choosing a merge movement of components s_{t_1} and s_{t_2} is

$$d_{s_{t_1} s_{t_2}} = \frac{2 n_{s_{t_1}} n_{s_{t_2}}}{T(T-1)}, \tag{2.8}$$

where $n_{s_{t_i}} = \sum_{t=1}^{T} I(s_t = s_{t_i})$ is the number of observations allocated to component s_{t_i}, $i = 1, 2$. A split of component $s_{t_*} = s_{t_1} = s_{t_2}$ is chosen with probability

$$b_{s_{t_*}} = \frac{n_{s_{t_*}}(n_{s_{t_*}} - 1)}{T(T-1)}, \tag{2.9}$$

where $n_{s_{t_*}} = \sum_{t=1}^{T} I(s_t = s_{t_*})$ is the number of observations allocated to component s_{t_*}. Note that $b_{s_{t_*}} + d_{s_{t_1} s_{t_2}}$ is not necessarily 1 since when $K > 1$, there are $K!(K-1)!$ pairs of components to be merged and K components to be split and one move among these possibilities is chosen.

Split

In a split of component s_{t_*} we redistribute its observations to the two new components and simulate parameters for them. Let $\mathbf{s}^{sp} = (s_1^{sp}, ..., s_T^{sp})$ be the configuration of $\mathbf{S}$ after the split.

1. **Reallocating observations:** For $t = 1, ..., T$ do

   - If $t = t_1$, allocate y_{t_1} to the $(K+1)$-th component, $s_{t_1}^{sp} = K + 1$.
   - If $t = t_2$, keep y_{t_2} in the same component, $s_{t_2}^{sp} = s_{t_2}$.
   - For the remaining observations in the splitting component, $s_t = s_{t_*}$, allocate y_t to component $K+1$ with probability $Pr_{s_{t_1}^{sp}}(t)$ or keep it in component s_{t_*} with probability $1 - Pr_{s_{t_1}^{sp}}(t)$. $Pr_{s_{t_1}^{sp}}(t)$ is chosen arbitrarily.
   - if $s_t \neq s_{t_*}$, keep y_t in the same component, $s_t^{sp} = s_t$.

   The new configuration $\mathbf{s}^{sp}$ of $\mathbf{S}$ has probability

$$Pr(\mathbf{s}^{sp}|\mathbf{s}) = \prod_{t:s_t^{sp}=s_{t_1}^{sp}} Pr_{s_{t_1}^{sp}}(t) \prod_{t:s_t^{sp}=s_{t_2}^{sp}} \left(1 - Pr_{s_{t_1}^{sp}}(t)\right), \tag{2.10}$$

   where $Pr_{s_{t_1}^{sp}}(t_1) = 1$ and $Pr_{s_{t_1}^{sp}}(t_2) = 0$. When $Pr_{s_{t_1}^{sp}}(t) = 1/2$, for $t \notin \{t_1, t_2\}$, this probability simplifies to

$$Pr(\mathbf{s}^{sp}|\mathbf{s}) = \left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}} + n_{s_{t_2}^{sp}} - 2}. \tag{2.11}$$

2. **Drawing parameters of the new components:** Conditional on $\mathbf{S} = \mathbf{s}^{sp}$, draw candidate-values $x^{sp} = (K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp})$ from their conditional *a posteriori* distributions. Moving from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^{sp}$ we just update $\boldsymbol{\theta}_{s_{t_1}^{sp}}$ and $\boldsymbol{\theta}_{s_{t_2}^{sp}}$, the parameters of components which have their configuration changed.

The proposal distribution of a split is

$$\begin{aligned} q(x^{sp}|x) = {} & b_{s_{t_*}} Pr(\mathbf{s}^{sp}|\mathbf{s}) \pi\left(\boldsymbol{\theta}_{s_{t_1}^{sp}}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}_{-\boldsymbol{\theta}_{s_{t_1}^{sp}}}\right) \\ & \times \pi\left(\boldsymbol{\theta}_{s_{t_2}^{sp}}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}_{-\boldsymbol{\theta}_{s_{t_2}^{sp}}}\right) \pi\left(\mathbf{p}_0^{sp}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \boldsymbol{\theta}^{sp}\right) \\ & \times \prod_{j=1}^{k+1} \pi\left(\mathbf{p}_j^{sp}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}_{-\mathbf{p}_j^{sp}}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}\right), \end{aligned} \tag{2.12}$$

where $\pi(\cdot|\cdot)$ is the conditional *a posteriori* distributions used to draw the candidate-values.

The acceptance probability for the split move is $\Psi(x^{sp}|x) = min(1, A^{sp})$, where A^{sp} is given by equation (2.7) as the product of the likelihoods ratio

$$\frac{L(\boldsymbol{\theta}^{sp}, \mathbf{p}_0^{sp}, \mathbf{P}^{sp}, K+1|\mathbf{y}, \mathbf{s}^{sp})}{L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s})} = \frac{\prod\limits_{t:s_t^{sp}=s_{t_1}^{sp}} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_{t_1}^{sp}}\right) \prod\limits_{t:s_t^{sp}=s_{t_2}^{sp}} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_{t_2}^{sp}}\right)}{\prod\limits_{t:s_t=s_{t_*}} f_{Y_t}\left(y_t|\boldsymbol{\theta}_{s_{t_*}}\right)} \frac{\prod\limits_{k=1}^{K+1}\left(\prod\limits_{j=0}^{K+1} p_{jk}^{sp\,n_{jk}^{sp}}\right)}{\prod\limits_{k=1}^{K}\left(\prod\limits_{j=0}^{K} p_{jk}^{n_{jk}}\right)}, \tag{2.13}$$

the *a priori* distributions ratio

$$\frac{\pi(\boldsymbol{\theta}^{sp}, \mathbf{p}_0^{sp}, \mathbf{P}^{sp}, K+1)}{\pi(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)} = \frac{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1}\gamma_{jk}\right)}{\prod_{k=1}^{K+1}\Gamma\left(\gamma_{jk}\right)}\prod_{k=1}^{K+1} p_{jk}^{sp^{\gamma_{jk}-1}}\right)\right)\pi(\boldsymbol{\theta}_{s_{t_1}^{sp}})\pi(\boldsymbol{\theta}_{s_{t_2}^{sp}})}{\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K}\gamma_{jk}\right)}{\prod_{k=1}^{K}\Gamma\left(\gamma_{jk}\right)}\prod_{k=1}^{K} p_{jk}^{\gamma_{jk}-1}\right)\right)\pi(\boldsymbol{\theta}_{s_{t_*}})} \tag{2.14}$$

and the proposal distributions ratio

$$\begin{aligned}\frac{q(x|x^{sp})}{q(x^{sp}|x)} &= \frac{2n_{s_{t_1}^{sp}}n_{s_{t_2}^{sp}}}{n_{s_{t_*}}(n_{s_{t_*}}-1)\left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}}+n_{s_{t_2}^{sp}}-2}} \\ &\times \frac{\pi\left(\boldsymbol{\theta}_{s_{t_*}}|\mathbf{y}, \mathbf{s}, K, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta}_{-\boldsymbol{\theta}_{s_{t_*}}}\right)}{\pi\left(\boldsymbol{\theta}_{s_{t_1}^{sp}}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}_{-\boldsymbol{\theta}_{s_{t_1}^{sp}}}\right)\pi\left(\boldsymbol{\theta}_{s_{t_2}^{sp}}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}_{-\boldsymbol{\theta}_{s_{t_2}^{sp}}}\right)} \\ &\times \frac{\pi\left(\mathbf{p}_0|\mathbf{y}, \mathbf{s}, K, \mathbf{P}, \boldsymbol{\theta}\right)\prod_{j=1}^{K}\pi\left(\mathbf{p}_j|\mathbf{y}, \mathbf{s}, K, \mathbf{P}_{-\mathbf{p}_j}\mathbf{p}_0, \boldsymbol{\theta}\right)}{\pi\left(\mathbf{p}_0^{sp}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}, \boldsymbol{\theta}^{sp}\right)\prod_{j=1}^{K+1}\pi\left(\mathbf{p}_j^{sp}|\mathbf{y}, \mathbf{s}^{sp}, K+1, \mathbf{P}^{sp}_{-\mathbf{p}_j^{sp}}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp}\right)}.\end{aligned} \tag{2.15}$$

If $\gamma_{jk} = 1$ for $k = 1, ..., K$ and $j = 0, 1, ..., K$ the *a priori* distributions ratio simplifies to

$$\frac{\pi(\boldsymbol{\theta}^{sp}, \mathbf{p}_0^{sp}, \mathbf{P}^{sp}, K+1)}{\pi(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)} = \frac{(K!)K^{K+1}\pi(\boldsymbol{\theta}_{s_{t_1}^{sp}})\pi(\boldsymbol{\theta}_{s_{t_2}^{sp}})}{\pi(\boldsymbol{\theta}_{s_{t_*}})}. \tag{2.16}$$

Merge

In a merge movement, we join the observations of two selected components and draw the parameters of this new component. Let $\mathbf{s}^{mg}$ be the configuration of $\mathbf{S}$ after a merge. The merge of components s_{t_1} and s_{t_2} is implemented by the following procedure.

1. **Joining observations:** For $t = 1, ..., T$ do

   - If $s_t < max(s_{t_1}, s_{t_2})$, do $s_t^{mg} = s_t$.
   - If $s_t = max(s_{t_1}, s_{t_2})$, do $s_t^{mg} = min(s_{t_1}, s_{t_2})$.
   - If $s_t > max(s_{t_1}, s_{t_2})$, do $s_t^{mg} = s_t - 1$.

The probability of this new configuration $\mathbf{s}^{mg} = (s_1^{mg}, ..., s_T^{mg})$ is

$$Pr(\mathbf{s}^{mg}|\mathbf{s}) = 1. \tag{2.17}$$

2. **Drawing parameters of the new component:** Conditional on $\mathbf{S} = \mathbf{s}^{mg}$, draw a new set of parameters $x^{mg} = (K-1, \mathbf{P}^{mg}, \mathbf{p}_0^{mg}, \boldsymbol{\theta}^{mg})$ from their conditional *a posteriori* distributions. Moving from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^{mg}$ we only update $\boldsymbol{\theta}_{s_{t_*}^{mg}}$, the parameters of the component whose configuration changed.

The proposal distribution of a merge is given by

$$\begin{aligned} q(x^{mg}|x) = d_{s_{t_1} s_{t_2}} \pi \left(\boldsymbol{\theta}_{s_{t_*}^{mg}} | \mathbf{y}, \mathbf{s}^{mg}, K-1, \mathbf{P}^{mg}, \mathbf{p}_0^{mg}, \boldsymbol{\theta}^{mg}_{-\boldsymbol{\theta}_{s_{t_*}^{mg}}} \right) \pi \left(\mathbf{p}_0^{mg} | \mathbf{y}, \mathbf{s}^{mg}, K-1, \mathbf{P}^{mg}, \boldsymbol{\theta}^{mg} \right) \\ \times \prod_{j=1}^{K-1} \pi \left(\mathbf{p}_j^{mg} | \mathbf{y}, \mathbf{s}^{mg}, K-1, \mathbf{P}^{mg}_{-\mathbf{p}_j^{mg}}, \mathbf{p}_0^{mg}, \boldsymbol{\theta}^{mg} \right), \end{aligned} \tag{2.18}$$

where $\pi\left(\cdot|\cdot\right)$ is the conditional *a posteriori* distribution for each parameter.

The acceptance probability of merging is $\Psi(x^{mg}|x) = min(1, A^{mg})$, where $A^{mg} = 1/A^{sp}$.

Note that the probability of proposing a split from x is the same as being in state x^{mg} and proposing a split, $q(x^{sp}|x) = q(x|x^{mg})$, and analogously $q(x^{mg}|x) = q(x|x^{sp})$.

If we first split the state x, giving x^{sp}, and then combine the components s_{t_1} and s_{t_2} we can recover x and state x is likely to be recovered after a split process of x^{mg}. Also note that a proposed movement is built without defining any arbitrary deterministic function or sampling arbitrary random variables. The proposed step is a special case of reversible jump when parameters of the proposed model are drawn from the proposal distribution and the Jacobian is equal to 1. More details are provided in the appendices of this chapter.

2.3.2 Transition between independent and first-order mixture models

In this section we propose a move to update the dependence order of the model. Let $x^0 = (K, \mathbf{p}, \boldsymbol{\theta})$ be an independent mixture model of K components, where $\mathbf{p} = (p_1, ..., p_K)$ and $p_k = Pr(S_t = k)$ for $k = 1, ..., K$. Let $x^1 = (K, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta})$ be a first-order mixture model of K components as described in Section 2.2. Transitions between these models are drawn using the conditional *a posteriori* distributions of $\mathbf{p}$ or $\mathbf{p}_0$ and $\mathbf{P}$ which are the modified parameters when we move from x^0 and x^1 maintaining K fixed. The proposed move is evaluated through the Metropolis-Hastings procedure.

The acceptance probability for moving from x^1 to x^0 is $\Psi(x^0|x^1) = min(1, A^0)$, where

$$A^0 = \frac{L(\boldsymbol{\theta}, \mathbf{p}, K|\mathbf{y}, \mathbf{s})}{L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s})} \frac{\pi(\mathbf{p}|K)}{\pi(\mathbf{p}_0, \mathbf{P}|K)} \frac{q(x^1|x^0)}{q(x^0|x^1)}, \tag{2.19}$$

$L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s})$ is given by (2.4); $L(\boldsymbol{\theta}, \mathbf{p}, K|\mathbf{y}, \mathbf{s}) = \prod_{k=1}^{K} \left\{ p_k^{n_k} \left(\prod_{t=1}^{T} f_{Y_t} \left(y_t|\boldsymbol{\theta}_{s_t}\right)^{I(S_t=k)} \right) \right\}$ is the likelihood function of independent model and $n_k = \sum_{t=1}^{T} I(S_t = k)$ for $k = 1, ..., K$ is calculated using the current configuration of $\mathbf{S}$; $\pi(\mathbf{p}|K)$ is the *a priori* distribution for $\mathbf{p}$ (a Dirichlet$(\gamma_1, ..., \gamma_K)$); $\pi(\mathbf{p}_0, \mathbf{P}|K)$ is the *a priori* distribution for $\mathbf{p}_0$ and $\mathbf{P}$ defined in Section 2.3; $q(x^0|x^1)$ and $q(x^1|x^0)$ are given by the conditional *a posteriori* distribution of the corresponding model, $\pi(\mathbf{p}|\mathbf{y}, \mathbf{s}, K, \boldsymbol{\theta})$=Dirichlet$(\gamma_1 + n_1, ..., \gamma_K + n_K)$ and $\pi(\mathbf{p}_0|\mathbf{y}, \mathbf{s}, K, \mathbf{P}, \boldsymbol{\theta}) \prod_{j=1}^{K} \pi(\mathbf{p}_j|\mathbf{y}, \mathbf{s}, K, \mathbf{p}_0, \mathbf{P}_{-\mathbf{p}_j}, \boldsymbol{\theta})$, respectively.

The acceptance probability for moving from x^0 to x^1 is $\Psi(x^1|x^0) = min(1, A^1)$, where $A^1 = 1/A^0$.

Note that transitions between the independent and first-order mixture models are easily built and evaluated using only the current state of the MCMC. The proposed step is also a special case of reversible jump where the Jacobian is equal to 1. More details are available in the appendices of this chapter.

2.3.3 Algorithm

The DDRJ is specified by the following steps:

1. Initialize K and $\mathbf{s} = \{s_1, ..., s_T\}$ such that $s_t \in \{1, ..., K\}$, for $t = 1, ..., T$.

2. Sample $\boldsymbol{\theta}_k$, for $k = 1, ..., K$, from its conditional *a posteriori* distribution.

3. Sample $\mathbf{p}_j$, for $j = 0, ..., K$, from its conditional *a posteriori* distribution.

4. Data-driven split-merge: Update K
   For $b = 1, ..., B$ do

   (a) Sample two observations, t_1 and t_2, and decide between split or merge.

   (b) Build the candidate for $\mathbf{s}^*$ and sample candidate-values x^*.

   (c) Accept the proposal with probability $\Psi(x^*|x)$, where $*$ is either *sp* or *mg*.

      i. If a split is accepted, do $K^{(b)} = K^{(b-1)} + 1$ and consider x^{sp}.

      ii. If a merge is accepted, do $K^{(b)} = K^{(b-1)} - 1$ and consider x^{mg}.

      iii. If no move is accepted, do $K^{(b)} = K^{(b-1)}$ and consider x.

5. Update s_t, for $t = 1, ..., T$, from a multinomial$(1, (\delta_{t1}, ..., \delta_{tK^{(b)}}))$, where
   $\delta_{tk} = Pr(S_t = k|\mathbf{y}, \mathbf{s}_{-t}, K^{(b)}, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta})$.

6. Update $\boldsymbol{\theta}_k$, for $k = 1, ..., K^{(b)}$, from its conditional *a posteriori* distribution.

7. Update $\mathbf{p}_j$, for $j = 0, ..., K^{(b)}$, from its conditional *a posteriori* distribution.

8. Move between dependent and independent models with $K^{(b)}$ components as described in Section 2.3.2.

The *a priori* distribution and likelihood function are invariant under permutation of the component labels. See Dahl (2006) for methods details about non identifiable parameters in mixture models, also called *label switching*.

2.4 Mixture of dependent binomial distributions

Now we apply the method to a mixture of binomial distributions although it can be adapted to a mixture of other distribution. In the next section, we will use this particular case to analyze the number of diagnosed cases of diabetes.

Consider $Y_t|S_t = k \sim \text{binomial}(m_t, \theta_k)$, $t = 1, 2, ..., T$ and $k \in \{1, 2, ..., K\}$, where m_t is known and $0 < \theta_k < 1$.The likelihood function is

$$
\begin{aligned}
L(\theta, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s}, \mathbf{m}) &= \prod_{k=1}^{K} \left\{ \left(\prod_{j=0}^{K} p_{jk}^{n_{jk}} \right) \prod_{t=1}^{T} \left(\binom{m_t}{y_t} \theta_k^{y_t} (1-\theta_k)^{m_t - y_t} \right)^{I(S_t=k)} \right\} \\
&= \left\{ \prod_{t=1}^{T} \binom{m_t}{y_t} \right\} \left\{ \prod_{k=1}^{K} \theta_k^{\sum\limits_{t:s_t=k} y_t} (1-\theta_k)^{\sum\limits_{t:s_t=k} (m_t - y_t)} \left(\prod_{j=0}^{K} p_{jk}^{n_{jk}} \right) \right\}, \quad (2.20)
\end{aligned}
$$

where $\mathbf{m} = \{m_1, ..., m_T\}$ and $\boldsymbol{\theta} = \{\theta_1, ..., \theta_K\}$.

Assume $\text{beta}(\alpha_k, \beta_k)$ as *a priori* distribution for θ_k, $k = 1, ..., K$, with known hyperparameters $\alpha_k > 0$ and $\beta_k > 0$. θ_k and $\theta_{k'}$ are supposed to be independent for $k \neq k'$.

The joint *a posteriori* distribution for θ, $\mathbf{p}_0$, $\mathbf{P}$ and K is given by

$$
\begin{aligned}
\pi(\theta, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s}, \mathbf{m}) &\propto L(\theta, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s}, \mathbf{m}) \pi(\theta, \mathbf{p}_0, \mathbf{P}, K) \\
&\propto \left\{ \prod_{k=1}^{K} \theta_k^{\sum\limits_{t:s_t=k} y_t + \alpha_k - 1} (1-\theta_k)^{\sum\limits_{t:s_t=k} (m_t - y_t) + \beta_k - 1} \left(\prod_{j=0}^{K} p_{jk}^{n_{jk} + \gamma_{jk} - 1} \right) \right\} \quad (2.21)
\end{aligned}
$$

and the conditional *a posteriori* distribution for θ_k is
$\theta_k| \left(\theta_{-\theta_k}, \mathbf{p}_0, \mathbf{P}, K, \mathbf{y}, \mathbf{s}, \mathbf{m}\right) \sim \text{beta}\left(\sum\limits_{t:s_t=k} y_t + \alpha_k, \sum\limits_{t:s_t=k} (m_t - y_t) + \beta_k \right)$, for $k = 1, ..., K$.

The likelihoods ratio for the split proposal is

$$\frac{L(\boldsymbol{\theta}^{sp}, \mathbf{p}_0^{sp}, \mathbf{P}^{sp}, K+1|\mathbf{y}, \mathbf{s}^{sp}, \mathbf{m})}{L(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K|\mathbf{y}, \mathbf{s}, \mathbf{m})} = \frac{\prod_{i=1}^{2}\left(\theta_{s_{t_i}^{sp}}^{\sum\limits_{t:s_t^{sp}=s_{t_i}^{sp}} y_t}\left(1-\theta_{s_{t_i}^{sp}}\right)^{\sum\limits_{t:s_t^{sp}=s_{t_i}^{sp}} (m_t-y_t)}\right)}{\theta_{s_{t_*}}^{\sum\limits_{t:s_t=s_{t_*}} y_t}\left(1-\theta_{s_{t_*}}\right)^{\sum\limits_{t:s_t=s_{t_*}} (m_t-y_t)}} \times \frac{\prod\limits_{k=1}^{K+1}\left(\prod\limits_{j=0}^{K+1} p_{jk}^{sp\,n_{jk}^{sp}}\right)}{\prod\limits_{k=1}^{K}\left(\prod\limits_{j=0}^{K} p_{jk}^{n_{jk}}\right)}, \tag{2.22}$$

the *a priori* distributions ratio is

$$\frac{\pi(\boldsymbol{\theta}^{sp}, \mathbf{p}_0^{sp}, \mathbf{P}^{sp}, K+1)}{\pi(\boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)} = \frac{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1}\gamma_{jk}\right)}{\prod_{k=1}^{K+1}\Gamma\left(\gamma_{jk}\right)}\prod_{k=1}^{K+1} p_{jk}^{sp\,\gamma_{jk}-1}\right)\right)}{\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K}\gamma_{jk}\right)}{\prod_{k=1}^{K}\Gamma\left(\gamma_{jk}\right)}\prod_{k=1}^{K} p_{jk}^{\gamma_{jk}-1}\right)\right)} \times \frac{\prod\limits_{i=1}^{2}\left(\frac{1}{B\left(\alpha_{s_{t_i}^{sp}}, \beta_{s_{t_i}^{sp}}\right)}\theta_{s_{t_i}^{sp}}^{\alpha_{s_{t_i}^{sp}}-1}\left(1-\theta_{s_{t_i}^{sp}}\right)^{\beta_{s_{t_i}^{sp}}-1}\right)}{\frac{1}{B\left(\alpha_{s_{t_*}}, \beta_{s_{t_*}}\right)}\theta_{s_{t_*}}^{\alpha_{s_{t_*}}-1}\left(1-\theta_{s_{t_*}}\right)^{\beta_{s_{t_*}}-1}} \tag{2.23}$$

and the proposal distribution ratio is

$$\frac{q(x|x^{sp})}{q(x^{sp}|x)} = \frac{2n_{s_{t_1}^{sp}}n_{s_{t_2}^{sp}}}{n_{s_{t_*}}\left(n_{s_{t_*}}-1\right)\left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}}+n_{s_{t_2}^{sp}}-2}} \times \frac{\frac{1}{B\left(\sum\limits_{t:s_t=s_{t_*}} y_t+\alpha_{s_{t_*}}, \sum\limits_{t:s_t=s_{t_*}} (m_t-y_t)+\beta_{s_{t_*}}\right)}\theta_{s_{t_*}}^{\sum\limits_{t:s_t=s_{t_*}} y_t+\alpha_{s_{t_*}}-1}\left(1-\theta_{s_{t_*}}\right)^{\sum\limits_{t:s_t=s_{t_*}} (m_t-y_t)+\beta_{s_{t_*}}-1}}{\prod\limits_{i=1}^{2}\left(\frac{1}{B\left(\sum\limits_{t:s_t^{sp}=s_{t_i}^{sp}} y_t+\alpha_{s_{t_i}^{sp}}, \sum\limits_{t:s_t=s_{t_*}} (m_t-y_t)+\beta_{s_{t_i}^{sp}}\right)}\theta_{s_{t_i}^{sp}}^{\sum\limits_{t:s_t^{sp}=s_{t_i}^{sp}} y_t+\alpha_{s_{t_i}^{sp}}-1}\left(1-\theta_{s_{t_i}^{sp}}\right)^{\sum\limits_{t:s_t=s_{t_*}} (m_t-y_t)+\beta_{s_{t_i}^{sp}}-1}\right)} \times \frac{\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K} n_{jk}+\gamma_{jk}\right)}{\prod_{k=1}^{K}\Gamma\left(n_{jk}+\gamma_{jk}\right)}\prod_{k=1}^{K} p_{jk}^{n_{jk}+\gamma_{jk}-1}\right)\right)}{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1} n_{jk}^{sp}+\gamma_{jk}\right)}{\prod_{k=1}^{K+1}\Gamma\left(n_{jk}^{sp}+\gamma_{jk}\right)}\prod_{k=1}^{K+1} p_{jk}^{sp\,n_{jk}^{sp}+\gamma_{jk}-1}\right)\right)}. \tag{2.24}$$

The acceptance probability for the split moves is $\Psi(x^{sp}|x) = min(1, A^{sp})$, where A^{sp} is given by the product of equations (2.22), (2.23) and (2.24),

$$A^{sp} = \frac{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1}\gamma_{jk}\right)}{\prod_{k=1}^{K+1}\Gamma\left(\gamma_{jk}\right)}\right)\right)\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K}n_{jk}+\gamma_{jk}\right)}{\prod_{k=1}^{K}\Gamma\left(n_{jk}+\gamma_{jk}\right)}\right)\right)}{\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K}\gamma_{jk}\right)}{\prod_{k=1}^{K}\Gamma\left(\gamma_{jk}\right)}\right)\right)\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1}n_{jk}^{sp}+\gamma_{jk}\right)}{\prod_{k=1}^{K+1}\Gamma\left(n_{jk}^{sp}+\gamma_{jk}\right)}\right)\right)} \frac{B\left(\alpha_{s_{t*}}, \beta_{s_{t*}}\right)}{\prod_{i=1}^{2} B\left(\alpha_{s_{t_i}^{sp}}, \beta_{s_{t_i}^{sp}}\right)}$$
$$\times \frac{\prod_{i=1}^{2} B\left(\sum_{t:s_t^{sp}=s_{t_i}^{sp}} y_t + \alpha_{s_{t_i}^{sp}}, \sum_{t:s_t=s_{t*}} (m_t - y_t) + \beta_{s_{t_i}^{sp}}\right)}{B\left(\sum_{t:s_t=s_{t*}} y_t + \alpha_{s_{t*}}, \sum_{t:s_t=s_{t*}} (m_t - y_t) + \beta_{s_{t*}}\right)} \frac{2n_{s_{t_1}^{sp}} n_{s_{t_2}^{sp}}}{n_{s_{t*}}(n_{s_{t*}}-1)\left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}}+n_{s_{t_2}^{sp}}-2}}. \qquad (2.25)$$

Consider $\gamma_{jk} = 1$, $\alpha_k = 1$ and $\beta_k = 1$ for $k = 1, ..., K+1$ and $j = 0, 1, ..., K+1$ in order to obtain vague *a priori* distributions. The acceptance probability for the split moves is $\Psi(x^{sp}|x) = min(1, A^{sp})$, where

$$A^{sp} = \frac{K^{K+1}K!\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K}n_{jk}+1\right)}{\prod_{k=1}^{K}\Gamma\left(n_{jk}+1\right)}\right)\right)}{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1}n_{jk}^{sp}+1\right)}{\prod_{k=1}^{K+1}\Gamma\left(n_{jk}^{sp}+1\right)}\right)\right)} \frac{\prod_{i=1}^{2} B\left(\sum_{t:s_t^{sp}=s_{t_i}^{sp}} y_t + 1, \sum_{t:s_t=s_{t*}} (m_t - y_t) + 1\right)}{B\left(\sum_{t:s_t=s_{t*}} y_t + 1, \sum_{t:s_t=s_{t*}} (m_t - y_t) + 1\right)}$$
$$\times \frac{2n_{s_{t_1}^{sp}} n_{s_{t_2}^{sp}}}{n_{s_{t*}}(n_{s_{t*}}-1)\left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}}+n_{s_{t_2}^{sp}}-2}} \qquad (2.26)$$

and $B(\cdot,\cdot)$ denotes the beta function.

In order to deal with *label switching*, we relabel the components at each MCMC iteration such that $\theta_1 < \theta_2 < \cdots < \theta_K$.

2.5 Applications

We apply DDRJ and RJ for the first-order model proposed by Robert et al. (2000) in simulated data sets, including a well-studied data set from Beckett & Diaconis (1994), and the real USA diabetes incidence. We consider a mixture of binomial distributions and set hyperparameters $\gamma_{jk} = 1$, $\gamma_k = 1$, $\alpha_k = 1$ and $\beta_k = 1$ for $k = 1, ..., K+1$ and $j = 0, 1, ..., K+1$ in order to obtain vague *a priori* distributions. Simulations have not shown sensitivity of the proposed method for the choice of different vague *a priori* distributions.

To carry out the RJ procedure, we define deterministic functions of current parameters to increase or reduce the dimension of the current model trying to keep the same stationary distribution of $\mathbf{P}$. When the proposal matrix $\mathbf{P}^{sp}$ does not have the features of a transition matrix, the proposal is rejected automatically. Depending on the defined deterministic functions, the calculation of the Jacobian is not trivial. As the RJ scheme does not have a high rejection rate of transdimensional moves (see RJ acceptance probabilities in the appendices of this chapter), we do not consider the refinements commented in Section 2.1 to improve its transdimensional moves acceptance. See Zuanetti & Milan (2015) for details about the RJ scheme carried out in this section.

Despite DDRJ having the additional step to select the best dependence order of the model, the DDRJ and RJ computational times are similar. We implement DDRJ and RJ methods in R language. R is a free software environment for statistical computing and graphics and more details are found on the following homepage: `https://www.r-project.org`. The DDRJ R codes for estimating a mixture of binomial distributions are available in the appendices of this chapter.

2.5.1 Simulated data sets

We consider a mixture of $K = 4$ binomial distributions with success probability $\theta_1 = 0.15$, $\theta_2 = 0.25$, $\theta_3 = 0.50$ and $\theta_4 = 0.85$, respectively, in a sequence of $T = 120$ observations. We simulate 12 distinct data sets varying m_t and $\mathbf{P}$ to verify the performance of DDRJ and RJ in different situations. Despite DDRJ and RJ do not require that all m_t's be equal, we fix m_t =10, 50 or 100 for each data set and choose $\mathbf{P}_1 = \{p_{jk} = 1/K; \forall j, k\}$ (independent mixture model), $\mathbf{P}_2 = \{p_{kk} = 0.30, p_{jk} = 0.70/(K-1); k = 1, ..., K, j \neq k\}$ (low probability of mixture components are the same successively), $\mathbf{P}_3 = \{p_{kk} = 0.50, p_{jk} = 0.50/(K-1); k = 1, ..., K, j \neq k\}$ (moderate probability of mixture components are the same successively) or $\mathbf{P}_4 = \{p_{kk} = 0.75, p_{jk} = 0.25/(K-1); k = 1, ..., K, j \neq k\}$ (high probability of mixture components are the same successively) to sample $\mathbf{S}$.

We run $B = 55000$ iterations for DDRJ and RJ, discard the first 5000 iterations and consider one draw for every 10 elements of the sequence. For each situation and method, we simulate two sequences with different starting points to check the convergence of the algorithm. One chain is initialized with $K = 1$ and the other with $K = 10$ and an independent model in DDRJ scheme. The results of both sequences are very similar indicating convergence of the chains and that the methods are not sensitive to the starting point. Here we show the results of sequences with starting point $K = 1$. The RJ algorithm does not include the step of selecting the dependence structure of the model since using arbitrary deterministic transformations to move between dependent and independent models hinders the convergence of the RJ algorithm. The trace plots for all the chains show no concerns about convergence in both cases.

We verify the convergence of the DDRJ and RJ sequences by the trace plots of K, effective sample size (ESS) (Kass *et al*, 1998) and integrated autocorrelation time (IAT) of the sequences. ESS is the number of effectively independent draws from *a posteriori* distribution. Large discrepancy between the ESS and the simulation sample size indicates poor mixing, *i.e.*, the method needs a large number of steps to scan the parametric space. Table 2.1 shows ESS for RJ and DDRJ K sequences. We observe that ESS of DDRJ sequences is always higher than ESS of

RJ sequences indicating a better mixing of DDRJ. IAT is a MCMC diagnostic which estimates the number of autocorrelated samples, on average, required to produce one independent drawn sample. The method with the lowest IAT is the most efficient. The IAT values of RJ and DDRJ K sequences are shown in the appendices of this chapter. Acceptance rates of transdimensional moves are also shown in the appendices of this chapter.

Table 2.1: Effective sample size of K sequence in DDRJ and RJ chains.

Method	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
RJ	39	50	80	128	50	29	51	42	26	26	67	101
DDRJ	319	319	218	211	182	168	301	147	187	190	301	444

Table 2.3 shows DDRJ *a posteriori* probability for the dependence structure of the model. The *a posteriori* probability is the relative frequency of occurrence of each model in the simulated sequence. DDRJ correctly identifies the first-order model as the most suitable for the data sets simulated from $\mathbf{P}_3$ and $\mathbf{P}_4$ and the independent model for $\mathbf{P}_1$. For $\mathbf{P}_2$, DDRJ identifies the independent model as the most suitable and this may be explained by the closeness of the transition matrix rows.

Table 2.2: DDRJ *a posteriori* probability for dependence structure of the model. The highest probability of each situation is in boldface type.

Model	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
Independent	**0.77**	**0.88**	0.10	0.00	**0.96**	**0.93**	$<$ 0.01	0.00	**0.94**	**0.90**	0.00	0.00
First-order	0.23	0.12	**0.90**	**1.00**	0.04	0.07	$>$ **0.99**	**1.00**	0.06	0.10	**1.00**	**1.00**

Table 2.2 shows DDRJ *a posteriori* probability for K of the most suitable model. The *a posteriori* probability for K is the relative frequency of each value of K in the simulated sequence. The DDRJ has a good performance to identify $K = 4$ when $m_t \geq 50$ for all the considered transition matrices. For $m_t = 10$, the first and second components are estimated as a single component. In this case, the number of trials ($m_t = 10$) is not big enough to distinguish $\theta_1 = 0.15$ and $\theta_2 = 0.25$. We note that for independent models ($\mathbf{P}_1$) higher values of m_t are necessary to identify the real number of components.

Table 2.4 shows the *a posteriori* probability for K obtained by RJ. First-order RJ only identifies $K = 4$ when $m_t = 100$ and $\mathbf{P}_3$ or $\mathbf{P}_4$. In cases of independent models or small values of m_t, some values of K have similar *a posteriori* probability and we cannot clearly identify the selected model. Comparing Tables 2.2 and 2.4, it can be observed that DDRJ is more accurate to identify the true model than the RJ for the simulated cases.

Table 2.3: DDRJ *a posteriori* probability for the number of components K. Higher probabilities are in boldface type.

K	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
2	**0.44**	0.12	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	**0.44**	**0.67**	**0.63**	**0.75**	**0.46**	0.25	0.00	0.00	0.00	0.01	0.00	0.00
4	0.10	0.18	0.30	0.14	**0.43**	**0.53**	**0.80**	**0.84**	**0.76**	**0.67**	**0.90**	**0.96**
5	0.02	0.03	0.06	0.02	0.10	0.19	0.18	0.16	0.21	0.27	0.09	0.04
≥ 6	0.00	0.00	0.01	0.00	0.01	0.03	0.02	0.00	0.03	0.05	0.01	0.00

Table 2.4: RJ *a posteriori* probability for the number of components K. Higher probabilities are in boldface type.

K	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
2	0.04	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	**0.19**	0.08	0.11	**0.39**	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	**0.21**	**0.20**	**0.34**	**0.38**	0.20	0.13	**0.31**	**0.50**	0.00	0.03	**0.60**	**0.81**
5	**0.23**	**0.27**	**0.33**	0.16	**0.33**	**0.25**	**0.38**	**0.43**	0.03	0.18	0.27	0.17
6	**0.16**	**0.23**	0.14	0.05	**0.29**	**0.30**	0.21	0.05	0.18	**0.28**	0.11	0.02
≥ 7	0.17	0.21	0.08	0.01	0.11	**0.32**	0.10	0.02	**0.79**	**0.51**	0.02	0.00

DDRJ estimates and credibility intervals for $\boldsymbol{\theta}$ and $\mathbf{P}$ are well estimated in all cases. When $m_t = 100$ and $\mathbf{P}_3$ or $\mathbf{P}_4$, RJ and DDRJ estimates of $\boldsymbol{\theta}$ and $\mathbf{P}$ are very similar.

We also apply DDRJ to analyze independent and first-order mixtures of $K = 5$ multinomial distributions with 10 possible events. A description of the data sets and results are available in the appendices of this chapter. Under those high-dimension simulated situations, DDRJ also shows a good performance in selecting and estimating the best model.

2.5.2 Rolling thumbtacks

We apply DDRJ and first-order RJ methods in Beckett & Diaconis (1994) data set. This is a well-studied example with binomial observations. They generated binary strings from rolls of common thumbtacks and considered that a 1 was scored if the tack landed point up and a 0 was recorded if the tack landed point down. The actual data arose from 12 different tacks, 2 flickers and 10 surfaces. Each tack was flicked 9 times and Y_t represents the number of ups out of $m_t = 9$ flips for $t = 1, ..., 320$. A spectral analysis of the data rejected the ordinary iid model of rolling. As in Liu (1996) and Newton *et al.* (1998), we assume that conditioned on a certain tack, the results of the 9 different flips are independent and a binomial model is adequate.

Liu (1996) and Newton *et al.* (1998) analyze this data set using a nonparametric Bayesian model for independent observations and they identified bimodality in the predictive distribution. Two modes were 0.52 and 0.79. A possible explanation for this might be that the tack data were produced by two people with some systematic differences in their flipping. Liu (1996) comments that this characteristic would rarely be identified by a regular parametric analysis using beta-binomial *a priori* distribution.

We run DDRJ and first-order RJ $B = 55000$ iterations, discard the first 5000 iterations and consider one draw for every 10 elements of the sequence. The chains are initialized with $K = 1$ and independent models in DDRJ.

DDRJ *a posteriori* probabilities for the independent and first-order model are 0.348 and 0.652, respectively. Therefore the first-order model fits better to the data than the independent model assumed in the nonparametric Bayesian analysis. A possible explanation for this result might be that the data is ordered by flickers and the number of ups are dependent as Beckett & Diaconis (1994).

Table 2.5 shows the DDRJ and RJ *a posteriori* probability for the number of components K considering a first-order model. In concordance with Liu (1996), DDRJ identifies $K = 2$. The estimates of the success probabilities (*a posteriori* average) and their 95% credibility interval are $0.51(0.44, 0.57)$ and $0.79(0.73, 0.84)$. The RJ method does not clearly clearly the value of K since the *a posteriori* probability for $K = 2, 3$ and 4 are all nearly the same. The ESS of DDRJ and RJ chain is 92 and 57, respectively, which indicates a better mixing for the DDRJ method.

Table 2.5: DDRJ and RJ *a posteriori* probability for the number of components K.

K	DDRJ	RJ
2	0.69	0.22
3	0.25	0.28
4	0.05	0.27
≥ 5	0.01	0.23

2.5.3 Number of diagnosed cases of diabetes

The incidence of diabetes has increased in recent years. From 1980 through 2013, the incidence of diagnosed diabetes increased in adults aged 18-44 years, 45-64 years and 65-79 years and has become one of the greatest public health concern.

The data we are analyzing is available on the following website: `http://www.cdc.gov/diabetes/statistics/incidence/fig3.htm`. The sequence of the yearly number of diagnosed cases of diabetes in the USA per 1000 population, from 1980 through 2013 ($T = 34$), is a realization of $\mathbf{Y}$, where $Y_t|S_t = k \sim \text{binomial}(m_t = 1000, \theta_k)$, $t = 1, 2, ..., 34$ and $k \in \{1, 2, ..., K\}$, m_t is known and $0 < \theta_k < 1$. We assume the number of diagnosed cases has a binomial distribution instead of another distribution such as Poisson or a negative binomial

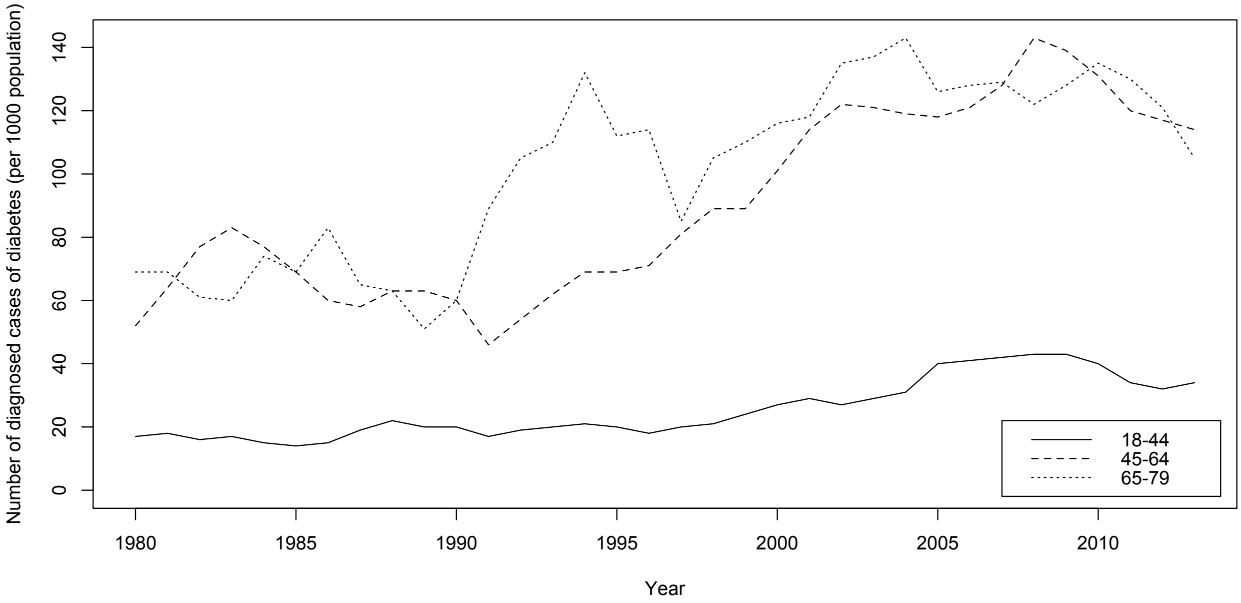


Figure 2.1: Number of diagnosed cases of diabetes by ages (per 1000 population).

since the number of trials is known ($m_t = 1000$) and we expect the variance of the number of diagnosed cases is lower than its mean inside a homogeneous period of time. Madden & Hughes (1995), Kranz (2003) and Rothman *et al.* (2008) show other cases where the binomial distribution or beta-binomial model is preferred to analyze the incidence of a disease.

Figure 2.1 shows the sequences of the number of diagnosed cases of diabetes by ages. It shows that from 1980 through 2013 the incidence of diagnosed diabetes increased for all adult ages. In the last three years, it has decreased, probably as a result of preventive policies. Ages ranges from 45-64 and 65-79 seem to have similar probabilities of diagnosing new cases of diabetes in the analyzed period.

We run DDRJ $B = 55000$ iterations, discard a burn in of the first 5000 iterations and select one every 10 values of the sequence for each age. The chains are initialized with $K = 1$ and the independent model. Convergence diagnostics identified no reason for concern.

The first-order mixture model fits better to the data than the independent mixture model. Table 2.6 shows the *a posteriori* probabilities for K by age, conditional on the first-order model. We observe the maximum probability at $K = 2$ and conclude that for the analyzed period, there are two different levels of probability of diagnosing diabetes for all ages. For ages in the 45-64 interval, we note that the model with $K = 3$ has a moderate frequency (0.40) indicating that this group is more heterogeneous than the other groups.

Table 2.6: The *a posteriori* probability for the number of components K.

K	$18-44$	$45-64$	$65-79$
2	0.99	0.60	0.92
3	0.01	0.40	0.08
4	0.00	< 0.004	< 0.001

The estimates of the two incidence probability and their 95% credibility intervals are presented in Table 2.7. We note the probability of diagnosing new cases of diabetes almost doubled for all age groups in the period. At the of $45-64$ and $65-79$, the diabetes incidence are similar. At the ages of 18-44, the incidences are lower than in the other ages.

Table 2.7: Estimates and 95% credibility intervals for diagnosed diabetes rates.

Rate	18-44	45-64	65-79
θ_1	$0.02(0.01-0.02)$	$0.07(0.06-0.09)$	$0.07(0.06-0.10)$
θ_2	$0.04(0.03-0.04)$	$0.12(0.12-0.13)$	$0.12(0.12-0.13)$

Analyzing the differences in trajectory of age subgroups, we estimate the expected value of $\mathbf{S}$. Figure 2.2 shows the estimates of *a posteriori* probability of the first binomial distribution by age groups. We observe that the model clearly detected a change in incidence in the analyzed period. For subgroups 18-44 and 45-65, the diabetes incidence increased after 2000. For the ages of 65-79, the incidence has been higher since 1992.

2.6 Use of the first-order mixture model in Genetics and Molecular Biology

We apply the proposed method to the number of diagnosed cases of diabetes. In addition, the first-order mixture model is widely used in Genetics and Molecular Biology. A DNA sequence $\mathbf{y} = \{y_1, y_2, ..., y_T\}$, for example, can be thought of as a realization of the observable sequence of random variables where $y_t \in \{a, c, g, t\} \equiv \{1, 2, 3, 4\}$, for $t = 1, 2, ..., T$, and where $Y_t|S_t = k \sim$ multinomial$(1, \boldsymbol{\theta}_k = (\theta_{k1}, \theta_{k2}, \theta_{k3}, \theta_{k4}))$. The letters represent the four nucleic acids or bases: adenine, cytosine, guanine and thymine.

Suppose there are K types of homogeneous segments in this DNA sequence. The nonobservable sequence of these spatial oriented homogeneous segments is a realization of the nonobservable Markov chain $\mathbf{S} = \{S_1, S_2, ..., S_T\}$, where $S_t \in \{1, 2, ..., K\}$, for $t = 1, 2, ..., T$. A DNA sequence can be modeled as a first-order multinomial mixture model and the methodology proposed in this section can be applied to select and estimate the best model. Details and applications of using a first-order mixture model to analyze DNA sequences are found in Churchill (1992), Churchill (1989), Muri (1998), Boys & Henderson (2004), Boys & Henderson (2002), Boys *et al.* (2000), Zuanetti (2006) and Zuanetti & Milan (To appear a).

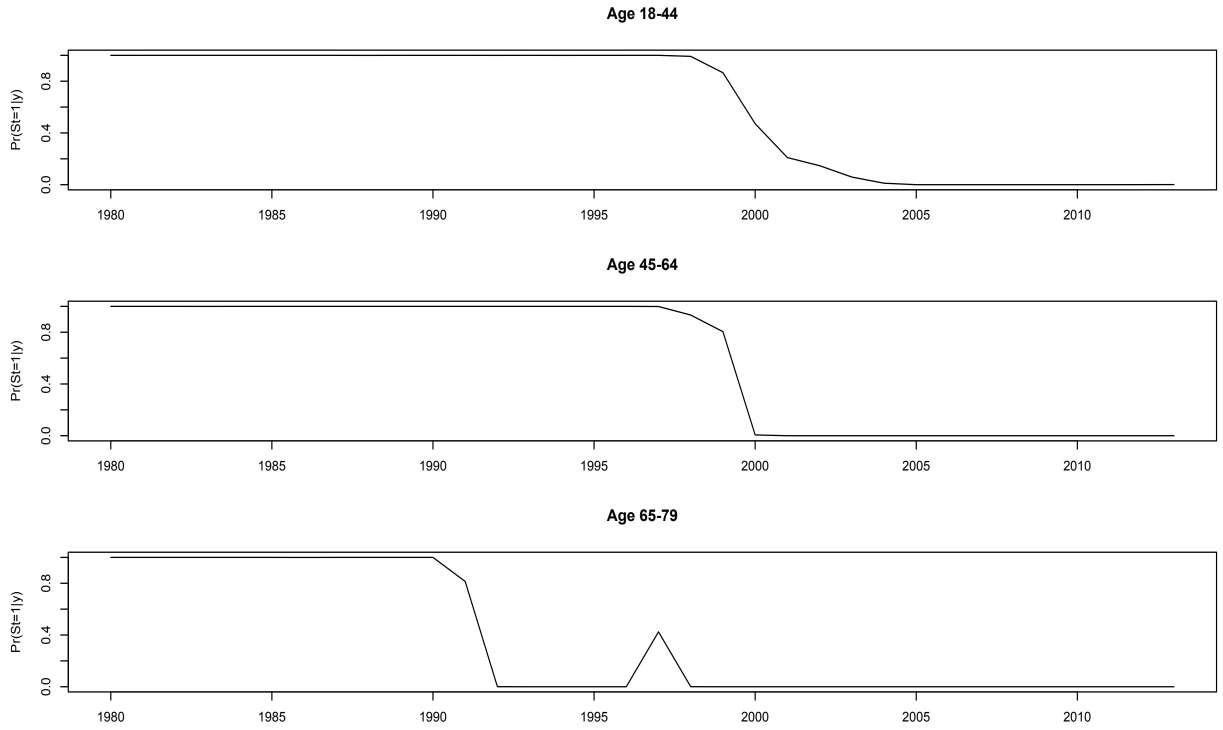


Figure 2.2: The *a posteriori* probability estimate of $P(S_t = 1|\mathbf{y}, \mathbf{s}_{-t}, \boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)$
.

The first-order mixture model is also used to input missing genotype in QTL mapping (Broman, 2006) and identity-by-descent (IBD) estimation along a chromosome (Abney, 2008; Brown *et al*, 2012; Druet & Farnir, 2011).

2.7 Discussion

We describe a generalization of the mixture model which accommodates the independent (usual) and first-order dependent mixture model (HMM) in one framework. We propose the data-driven reversible jump as a tool for model selection and estimation of the model.

Simulations show better performance of DDRJ to identify the most suitable model and estimate its parameters in situations where usual methods do not select the best model.

DDRJ also shows good results analyzing Beckett & Diaconis (1994) data set where other beta-binomial parametric methods (RJ, for example) cannot work very well. In addition to properly estimate the number of components, DDRJ identifies the first-order model as the best model to describe Beckett & Diaconis (1994) data set.

We apply the generalized mixture model to the yearly USA diabetes incidence data. We observe the incidence increasing after 2000 for 18-64 ages. For the ages of 65-79, the incidence has increased from 1992.

One of the most interesting aspects of the proposed method is that it is applicable for selecting and estimating a wide variety of models, the simple usual mixture model (independent model) and the hidden Markov model (first-order mixture model), including cases of time series and change point models. The method can also accommodate more than one distribution for the observable variable, all participating in the model selection procedure.

Here we only use conjugate *a priori* distributions. However the method works well for nonconjugate distributions. In these cases, it may be necessary changing some Gibbs sampling to Metropolis-Hastings steps to simulate values from the conditional *a posteriori* distributions of parameters.

2.8 Appendices

2.8.1 Validity of split-merge acceptance probability

Consider a split move from state $x = (K, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta})$ to $x^{sp} = (K+1, \mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}^{sp})$. Let $\boldsymbol{\lambda} = \left(\mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta}_{s_{t_*}}\right)$, $\mathbf{u} = \left(\mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}_{s_{t_1}^{sp}}, \boldsymbol{\theta}_{s_{t_2}^{sp}}\right)$ be auxiliary variables of the transition $x \to x^{sp}$, $\boldsymbol{\lambda}^{sp} = \left(\mathbf{P}^{sp}, \mathbf{p}_0^{sp}, \boldsymbol{\theta}_{s_{t_1}^{sp}}, \boldsymbol{\theta}_{s_{t_2}^{sp}}\right)$ and $\mathbf{u}^{sp} = \left(\mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta}_{s_{t_*}}\right)$ be auxiliary variables of the transition $x^{sp} \to x$ which represents a merge move.

In the way we propose the DDRJ algorithm to update the number of components K, the transition $x \to x^{sp}$ involves a deterministic map $h(\boldsymbol{\lambda}, \mathbf{u}) = (\mathbf{u}^{sp}, \boldsymbol{\lambda}^{sp})$ where the proposal density of $\mathbf{u} = \boldsymbol{\lambda}^{sp}$ does not depend on $\boldsymbol{\lambda}$ and $h(\cdot, \cdot)$ is one-to-one function with unity Jacobian. Therefore, the acceptance probability of the split move is defined as (2.7) and the proposed DDRJ method to update the number of components K is a special case of reversible jump algorithm. The proposed chain is ergodic and its convergence to the desirable invariant distribution is guaranteed.

2.8.2 Validity of dependence order acceptance probability

Consider a transdimensional move from an independent model $x^0 = (K, \mathbf{p}, \boldsymbol{\theta})$ to a first-order model $x^1 = (K, \mathbf{P}, \mathbf{p}_0, \boldsymbol{\theta})$. Let $\mathbf{u}^0 = (\mathbf{P}, \mathbf{p}_0)$ be auxiliary variables of the transition $x^0 \to x^1$ and $\mathbf{u}^1 = \mathbf{p}$ be auxiliary variables of the transition $x^1 \to x^0$.

In the way we propose the DDRJ algorithm to update the dependence order of the model, the transition $x^0 \to x^1$ involves a deterministic map $g(\mathbf{p}, \mathbf{u}^0) = (\mathbf{u}^1, (\mathbf{P}, \mathbf{p}_0))$ where the proposal density of $\mathbf{u}^0 = (\mathbf{P}, \mathbf{p}_0))$ does not depend on $\mathbf{p}$ and $g(\cdot, \cdot)$ is one-to-one function with unity Jacobian. Therefore, the acceptance probability of the transition $x^0 \to x^1$ is defined as (2.19) and the proposed scheme is another special case of reversible jump algorithm. The convergence of proposed chain to the desirable invariant distribution is guaranteed.

2.8.3 Additional mixing and statistical convergence diagnostics of simulated data sets

Table 2.8 shows integrated autocorrelation time (IAT) measure for RJ and DDRJ K sequences. DDRJ shows lower IAT than RJ in all tested situations confirming the better mixing of DDRJ sequences.

Table 2.8: IAT of K sequence in DDRJ and RJ chains.

Method	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
RJ	64	49	31	19	49	86	49	59	96	96	37	24
DDRJ	7	7	11	11	13	14	8	17	13	13	8	5

Table 2.9 and 2.10 shows the acceptance rates of split-merge proposals and dependence order, respectively.

Table 2.9: Acceptance rate of split-merge proposals.

Method	m_t											
	10				50				100			
	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
RJ	0.918	0.834	0.824	0.927	0.520	0.374	0.411	0.187	0.150	0.285	0.318	0.194
DDRJ	0.014	0.013	0.008	0.005	0.008	0.008	0.004	0.002	0.005	0.006	0.002	0.002

Table 2.10: Acceptance rate of dependence order proposals in DDRJ chains.

m_t											
10				50				100			
$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$
0.21	0.091	0.081	0.001	0.044	0.046	0.001	0.001	0.044	0.077	0.001	0.001

2.8.4 Generalized mixture of multinomial distributions

We apply the proposed MCMC method to a mixture of multinomial distributions in order to analyze the performance of DDRJ in selecting and estimating higher-dimensional mixture models.

Assume $\mathbf{Y}_t = (Y_{t1}, ..., Y_{tL})|S_t = k \sim \text{multinomial}(m_t, \boldsymbol{\theta}_k = (\theta_{k1}, ..., \theta_{kL}))$, for $t = 1, 2, ..., T$ and $k \in \{1, 2, ..., K\}$ and where $m_t = \sum_{l=1}^{L} y_{tl}$ and $\sum_{l=1}^{L} \theta_{tl} = 1$. We consider a conjugate Dirichlet$(\alpha_{k1}, \cdots, \alpha_{kL})$ as *a priori* distribution for $\boldsymbol{\theta}_k$, $k = 1, ..., K$, with known hyperparameters $\alpha_{kl} > 0$. $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_{k'}$ are supposed to be independent for $k \neq k'$.

In order to simulate data sets, we consider a mixture of $K = 5$ multinomial distributions with events probabilities $\boldsymbol{\theta}_1 = (0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10)$, $\boldsymbol{\theta}_2 = (0.05, 0.15, 0.05, 0.15, 0.05, 0.15, 0.05, 0.15, 0.05, 0.15)$, $\boldsymbol{\theta}_3 = (0.15, 0.05, 0.15, 0.05, 0.15, 0.05, 0.15, 0.05, 0.15, 0.05)$, $\boldsymbol{\theta}_4 = (0.025, 0.025, 0.025, 0.025, 0.40, 0.40, 0.025, 0.025, 0.025, 0.025)$ and $\boldsymbol{\theta}_5 = (0.10, 0.25, 0.05, 0.0375, 0.0375, 0.05, 0.0375, 0.30, 0.0375, 0.10)$ respectively, in a sequence of $T = 500$ observations. The events probabilities of first three components are similar and usually it is hard to identify all of them.

We simulate 4 distinct data sets combining two values of m_t and two transition matrices $\mathbf{P}$ to verify the performance of DDRJ in different situations of high dimension. We fix $m_t = 50$ or 100 for each data set and choose $\mathbf{P}_1 = \{p_{jk} = 1/K; \forall j, k\}$ (independent mixture model) and $\mathbf{P}_3 = \{p_{kk} = 0.50, p_{jk} = 0.50/(K-1); k = 1, ..., K, j \neq k\}$ (first-order mixture model with moderate probability of mixture components are the same successively) to sample $\mathbf{S}$. We set hyperparameters $\gamma_{jk} = 1$, $\gamma_k = 1$ and $\alpha_{kl} = 1$ for $k = 1, ..., K+1$, $j = 0, 1, ..., K+1$ and $l = 1, ..., L$ which represents vague *a priori* distributions.

Considering these values of hyperparameters, the acceptance probability for the split moves is $\Psi(x^{sp}|x) = min(1, A^{sp})$, where

$$A^{sp} = \frac{K^{K+1}K!(L-1)!\left(\prod_{j=0}^{K}\left(\frac{\Gamma\left(\sum_{k=1}^{K} n_{jk}+1\right)}{\prod_{k=1}^{K}\Gamma\left(n_{jk}+1\right)}\right)\right)}{\left(\prod_{j=0}^{K+1}\left(\frac{\Gamma\left(\sum_{k=1}^{K+1} n_{jk}^{sp}+1\right)}{\prod_{k=1}^{K+1}\Gamma\left(n_{jk}^{sp}+1\right)}\right)\right)} \frac{\frac{\Gamma\left(\sum_{l=1}^{L} m_{s_{t_*}}+1\right)}{\prod_{l=1}^{L}\Gamma\left(m_{s_{t_*}}+1\right)}}{\prod_{i=1}^{2}\frac{\Gamma\left(\sum_{l=1}^{L} m_{s_{t_i}^{sp}}+1\right)}{\prod_{l=1}^{L}\Gamma\left(m_{s_{t_i}^{sp}}+1\right)}}$$

$$\times \frac{2n_{s_{t_1}^{sp}}n_{s_{t_2}^{sp}}}{n_{s_{t_*}}\left(n_{s_{t_*}}-1\right)\left(\frac{1}{2}\right)^{n_{s_{t_1}^{sp}}+n_{s_{t_2}^{sp}}-2}}. \tag{2.27}$$

We run DDRJ $B = 30000$ iterations, discard the first 5000 iterations and consider one draw for every 5 elements of the sequence. We start each MCMC chain with $K = 1$ and an independent mixture model. We evaluate the convergence of the chains through trace plots.

The *a priori* distribution and likelihood function are invariant under permutation of the component labels. In order to deal with *label switching*, we relabel the components at each MCMC iteration such that $\theta_{11} \leq \theta_{21} \leq \cdots \leq \theta_{K1}$, $\theta_{12} \leq \theta_{22} \leq \cdots \leq \theta_{K2}$ and so on.

Table 2.11 shows DDRJ *a posteriori* probability for the dependence structure of the model. *A posteriori* probability is calculated as the relative frequency of each dependence order in the sequence. The DDRJ correctly identifies the first-order model as more suitable for all data set simulated from $\mathbf{P}_3$ and independent model for $\mathbf{P}_1$.

Table 2.11: The *a posteriori* probability for dependence structure of the model obtained by DDRJ.

Model	m_t			
	50		100	
	$\mathbf{P}_1$	$\mathbf{P}_3$	$\mathbf{P}_1$	$\mathbf{P}_3$
Independent	1.00	0.00	1.00	0.00
First-order	0.00	1.00	0.00	1.00

Table 2.12 shows DDRJ *a posteriori* probability for K of the most suitable model in each situation. *A posteriori* probability for K is calculated as the relative frequency of each value of K in the sequence. We observe, under simulated conditions, DDRJ has a good performance to identify $K = 5$ when $m_t = 100$. When $m_t = 50$, the number of trials are not enough for DDRJ

identifies the component 1 whose events probabilities are the average of the probabilities of similar components 2 and 3. Components 2 and 3 (which also have similar events probabilities), 4 and 5 are very well identified.

Table 2.12: The *a posteriori* probability for the number of components K obtained by DDRJ.

K	m_t			
	50		100	
	$\mathbf{P}_1$	$\mathbf{P}_3$	$\mathbf{P}_1$	$\mathbf{P}_3$
4	1.00	1.00	0.00	0.246
5	0.00	0.00	1.00	0.754

The misclassification table of $\mathbf{s}$ for each situation are shown in Tables 2.13, 2.14, 2.15 and 2.16. The estimated s_t, for $t = 1, ..., T$, is the component which maximizes $P(S_t|\mathbf{y}, \mathbf{s}_{-t}, \boldsymbol{\theta}, \mathbf{p}, K)$ in an independent model or $P(S_t|\mathbf{y}, \mathbf{s}_{-t}, \boldsymbol{\theta}, \mathbf{p}_0, \mathbf{P}, K)$ in a first-order model. When the components' label of estimated s_t is switched compared with true estimated s_t, we relabel estimated s_t and choose the labels' permutation which minimize misclassification rate. We observe that members of component 1 are allocated in components 2 and 3 when $m_t = 50$ which it is expected since events probabilities of component 1 are close to events probabilities of components 2 and 3. Members of components 2, 3, 4 and 5 when $m_t = 50$ and of all the components when $m_t = 100$ are very well identified.

Table 2.13: Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 50$ and $\mathbf{P}_1$.

True $\mathbf{s}$	Estimated $\mathbf{s}$			
	1	2	3	4
1	28	71	0	1
2	89	0	0	1
3	0	99	0	0
4	0	0	118	0
5	0	0	0	93

Table 2.14: Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 50$ and $\mathbf{P}_3$.

True $\mathbf{s}$	Estimated $\mathbf{s}$			
	1	2	3	4
1	57	34	0	0
2	107	0	0	0
3	0	100	0	0
4	0	0	118	0
5	1	0	0	83

Table 2.15: Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 100$ and $\mathbf{P}_1$.

True $\mathbf{s}$	Estimated $\mathbf{s}$				
	1	2	3	4	5
1	99	0	1	0	0
2	0	90	0	0	0
3	1	0	98	0	0
4	0	0	0	118	0
5	0	0	0	0	93

Table 2.16: Misclassification table of true and estimated $\mathbf{s}$ when $m_t = 100$ and $\mathbf{P}_3$.

True $\mathbf{s}$	Estimated $\mathbf{s}$				
	1	2	3	4	5
1	90	0	1	0	0
2	0	107	0	0	0
3	0	0	100	0	0
4	0	0	0	118	0
5	0	0	0	0	84

Here we don't show estimates and credibility intervals for parameters $\boldsymbol{\theta}$, $\mathbf{p}$, $\mathbf{p}_0$ and $\mathbf{P}$ since we have, depending on situation, at least 49 parameters. However, estimates are very close to true values and credibility intervals are short.

2.8.5 R codes to carry out DDRJ

In this section, we show DDRJ R codes for a simulated data set.

```
#
# mixture of K binomials distributions
#
################### useful functions
#
rDiric<-function(a){
  X<-rgamma(length(a),a,1)
  Y<-X/sum(X)
  Y}
#
rmultinomial<-function(p){
  u<-runif(1)
  P<-cumsum(p)
  val<-sum(P<u)+1
  val}
```

```
#
somaYt<-function(Y,C,k){
  Ta<-length(Y)
  somaYij<-c(rep(0,k))
  for (j in 1:k){
    for (i in 1:Ta){
      if(C[i]==j) somaYij[j]<-somaYij[j]+Y[i]}}
  somaYij}
#
contank<-function(C,k){
  Ta<-length(C)
  nj<-rep(0,k)
    for (j in 1:k){
      for (i in 1:Ta){
        if(C[i]==j) nj[j]<-nj[j]+1}}
  nj}
#
contanjk<-function(C,k){
  Ta<-length(C)
  nji<-matrix(0,k,k)
    for (j in 1:k){
      for (i in 1:k){
        for (l in 1:(Ta-1)){
          if(C[l]==j & C[l+1]==i) nji[j,i]<-nji[j,i]+1}}}
  nji}
#
contan0k<-function(C1,k){
  n0<-rep(0,k)
  n0[C1]<-1
  n0}
#
posttetak<-function(alfa,beta,somayi,somaniyi){
  tetak<-rbeta(1,(alfa+somayi),(beta+somaniyi))
  tetak}
#
postpj<-function(gama,njk){
  gamav<-rep(gama,length(njk))
  pj<-rDiric(gama+njk)
  pj}
#
postSj<-function(P,teta,yt,nt,sta,stp){
  K<-nrow(P)
  probst<-numeric(K)
  for (i in 1:K) probst[i]<-exp(log(P[i,stp])+log(P[sta,i])+log(dbinom(yt, nt,
teta[i])))
  probst<-probst/sum(probst)
  st<-rmultinomial(probst)}
#
postSjind<-function(P,teta,yt,nt){
  K<-length(P)
  probst<-numeric(K)
  for (i in 1:K) probst[i]<-exp(log(P[i])+log(dbinom(yt, nt, teta[i])))
  probst<-probst/sum(probst)
  st<-rmultinomial(probst)}
#
postS1<-function(P,P0,teta,y1,n1,stp){
  K<-nrow(P)
  probst<-numeric(K)
  for (i in 1:K) probst[i]<-exp(log(P[i,stp])+log(P0[i])+log(dbinom(y1, n1,
teta[i])))
  probst<-probst/sum(probst)
  st<-rmultinomial(probst)}
#
postST<-function(P,teta,yT,nT,sta){
  probst<-numeric()
  K<-nrow(P)
  for (i in 1:K) probst[i]<-exp(log(P[sta,i])+log(dbinom(yT, nT, teta[i])))
  probst<-probst/sum(probst)
  st<-rmultinomial(probst)}
#
propind<-function(katual,vetorY, vetorc, vetorNt, alfa, beta, gama, teta){
  Nij0<-matrix(contank(vetorc,katual),nrow=katual,ncol=katual,byrow=TRUE)
  vet.prob<-postpj(gama,Nij0[1,])
  MT<-matrix(vet.prob,katual,katual,byrow=TRUE)
  P0<-vet.prob
  T<-length(vetorY)
  Snovo<-numeric(T)
  for (t in (1:T)) Snovo[t]<-postSjind(MT[1,],teta,vetorY[t],vetorNt[t])
  list(MT,Snovo,teta,P0,0)}
#
propdepend<-function(katual,vetorY, vetorc, vetorNt, alfa, beta, gama, teta){
  Nij0<-contanjk(vetorc,katual)
  N1j0<-contan0k(vetorc[1],katual)
  MT<-matrix(0,katual,katual)
  for (j in (1:katual)) MT[j,]<-postpj(gama,Nij0[j,])
  P0<-postpj(gama,N1j0)
  T<-length(vetorY)
  Snovo<-numeric(T)
  Snovo[1]<-postS1(MT,P0,teta,vetorY[1],vetorNt[1],vetorc[2])
  for (t in (2:(T-1))) Snovo[t]<-postSj(MT,teta,vetorY[t],
vetorNt[t],Snovo[t-1],vetorc[t+1])
  Snovo[T]<-postST(MT,teta,vetorY[T],vetorNt[T],Snovo[T-1])
  list(MT,Snovo,teta,P0,1)}
#
probaceitdepend<-function(alfa,beta,gama,vetorY,vetorNt,vetorCind,
vetorCpo,teta,P0,Ppo,Pind){
  #
  ### acceptance probability of being in a first-order
  ### model and transit to an independent model
  #
  veroind<-0
  veropo<-0
  T<-length(vetorY)
  K<-length(teta)
  Nij1<-contanjk(vetorCpo,K)
  N1j1<-contan0k(vetorCpo[1],K)
  Nijt1<-rbind(N1j1,Nij1)
  Nk0<-contank(vetorCind,K)
  for (t in 1:T){
    veroind<-veroind+dbinom(vetorY[t],vetorNt[t],teta[vetorCind[t]],log=TRUE)
    veropo<-veropo+dbinom(vetorY[t],vetorNt[t],teta[vetorCpo[t]],log=TRUE)}
  veroind<-veroind+sum(Nk0*log(Pind))
  for (i in 1:nrow(Nijt1)) veropo<-veropo+sum(Nijt1[i,]*log(rbind(P0,Ppo)[i,]))
  vero<-veroind-veropo
  #
  prio<--K*lgamma(K)
  #
  Nij0<-contanjk(vetorCind,K)
  N1j0<-contan0k(vetorCind[1],K)
  Nijt0<-rbind(N1j0,Nij0)+gama
  Nk1<-contank(vetorCpo,K)+gama
  b1<-lgamma(sum(Nk1))
  b2<-sum(lgamma(Nk1))
  b3<-b4<-0
  for (i in 1:nrow(Nijt0)){
    b3<-b3+lgamma(sum(Nijt0[i,]))
    b4<-b4+sum(lgamma(Nijt0[i,]))}
  post1<-b3-b4-b1+b2
  #
  Nijt0<-Nijt0-gama
  Nk1<-Nk1-gama
  a1<-sum(Nk1*log(Pind))
  a2<-0
  for (i in 1:nrow(Nijt0)) a2<-a2+sum(Nijt0[i,]*log(rbind(P0,Ppo)[i,]))
  post2<-a2-a1
  #
  postSpo<-0
  probst<-numeric(K)
  for (i in 1:K) probst[i]<-exp(log(P0[i])+log(Ppo[i,vetorCind[2]])+
dbinom(vetorY[1],vetorNt[1],teta[i],log=TRUE))
  probst<-probst/sum(probst)
  postSpo<-postSpo+log(probst[vetorCpo[1]])
  for (t in 2:(T-1)){
    for (i in 1:K) probst[i]<-exp(log(Ppo[vetorCpo[t-1],i])+log(
Ppo[i,vetorCind[t+1]])+dbinom(vetorY[t],vetorNt[t],teta[i],log=TRUE))
    probst<-probst/sum(probst)
    postSpo<-postSpo+log(probst[vetorCpo[t]])}
  for (i in 1:K) probst[i]<-exp(log(Ppo[vetorCpo[T-1],i])+dbinom(vetorY[T],
vetorNt[T],teta[i],log=TRUE))
  probst<-probst/sum(probst)
  postSpo<-postSpo+log(probst[vetorCpo[T]])
  #
  postSind<-0
  probst<-numeric(K)
  for (t in 1:T){
    for (i in 1:K) probst[i]<-exp(log(Pind[i])+dbinom(vetorY[t],
 vetorNt[t],teta[i],log=TRUE))
    probst<-probst/sum(probst)
    postSind<-postSind+log(probst[vetorCind[t]])}
  #
  post3<-postSpo-postSind
  #
  PA<-exp(vero+prio+post1+post2+post3)
  PA}
#
propmerge<-function(katual,vetorY, vetorc, vetorNt, alfa,
beta, gama, pos1, pos2, teta, depend){
  #
  #
  ############## merge both components
  componentes<-sort(c(vetorc[pos1],vetorc[pos2]))
  vetorcnovo<-vetorc
  Ta<-length(vetorc)
  for (i in 1:Ta){
    if (vetorc[i]==componentes[2]) vetorcnovo[i]<-componentes[1]
    if (vetorc[i]>componentes[2]) vetorcnovo[i]<-vetorc[i]-1}
  K<-katual-1
  #
  ############ sample theta for the new component
  a<-sum(vetorY[vetorcnovo==componentes[1]])
  b<-sum(vetorNt[vetorcnovo==componentes[1]])
  tetanovo<-teta
  tetanovo[componentes[1]]<-posttetak(alfa,beta,a,(b-a))
  for (j in (componentes[2]:katual)) tetanovo[j]<-teta[j+1]
  tetanovo<-tetanovo[1:K]
  #
  ################### sample P e P0
  if (depend==1){
    N<-contanjk(vetorcnovo,K)
    N0<-contan0k(vetorcnovo[1],K)
    MTnovo<-matrix(0,K,K)
    for (j in (1:K)) MTnovo[j,]<-postpj(gama,N[j,])
    P0novo<-postpj(gama,N0)} else{
    N0<-contank(vetorcnovo,K)
    N<-matrix(N0,nrow=K,ncol=K,byrow=TRUE)
    vet.prob<-postpj(gama,N[1,])
    MTnovo<-matrix(vet.prob,K,K,byrow=TRUE)
    P0novo<-vet.prob}
  #
  list(K,vetorcnovo,tetanovo,MTnovo,P0novo,a,b,N,N0)}
#
propsplit<-function(katual, vetorY, vetorc, vetorNt, alfa, beta,
gama, pos1, pos2, teta, depend){
  #
  K<-katual+1
  #
  ################ split one component in two
  componente<-vetorc[pos1]
  vetorcnovo<-vetorc
  Ta<-length(vetorc)
  for (i in 1:Ta){
    if (vetorc[i]==componente){
      aux<-runif(1)
      if (aux<0.5) vetorcnovo[i]<-componente else vetorcnovo[i]<-K}}
  vetorcnovo[pos1]<-componente
  vetorcnovo[pos2]<-K
  #
  ############## sample theta for the new components
  tetanovo<-teta
  a<-sum(vetorY[vetorcnovo==componente])
  b<-sum(vetorNt[vetorcnovo==componente])
  tetanovo[componente]<-posttetak(alfa,beta,a,(b-a))
  a1<-sum(vetorY[vetorcnovo==K])
  b1<-sum(vetorNt[vetorcnovo==K])
  tetanovo<-c(tetanovo,posttetak(alfa,beta,a1,(b1-a1)))
  #
  ################ sample P e P0
  if (depend==1){
    N<-contanjk(vetorcnovo,K)
    N0<-contan0k(vetorcnovo[1],K)
    MTnovo<-matrix(0,K,K)
    for (j in (1:K)) MTnovo[j,]<-postpj(gama,N[j,])
    P0novo<-postpj(gama,N0)} else {
    N0<-contank(vetorcnovo,K)
    N<-matrix(N0,nrow=K,ncol=K,byrow=TRUE)
    vet.prob<-postpj(gama,N[1,])
    MTnovo<-matrix(vet.prob,K,K,byrow=TRUE)
    P0novo<-vet.prob}
  #
  list(K,vetorcnovo,tetanovo,MTnovo,P0novo,a,b,a1,b1,N,N0)}
#
probaceit<-function(alfa,beta,sytsi,sntytsi,sytsisp,sntytsisp,
sytsjsp,sntytsjsp,gama,N,Nsp,nsisp,nsjsp,nsi){
  #
  ### acceptance probability of a split proposal
  #
  a<-(-lbeta(alfa+sytsi,beta+sntytsi))+lbeta(alfa+sytsisp,beta+
sntytsisp)+lbeta(alfa+sytsjsp,beta+sntytsjsp)
  #
  b1<-b2<-b3<-b4<-0
  for(j in 1:nrow(Nsp)){
    a1<-a2<-a3<-a4<-0
    for (i in 1:ncol(Nsp)){
      a1<-a1+gama
      a2<-a2+gama+Nsp[j,i]
      a3<-a3+lgamma(gama)
      a4<-a4+lgamma(gama+Nsp[j,i])}
    b1<-b1+lgamma(a1)-a3
    b2<-b2+lgamma(a2)-a4}
  for(j in 1:nrow(N)){
    c1<-c2<-c3<-c4<-0
    for (i in 1:ncol(N)){
      c1<-c1+gama
      c2<-c2+gama+N[j,i]
      c3<-c3+lgamma(gama)
      c4<-c4+lgamma(gama+N[j,i])}
    b3<-b3+lgamma(c1)-c3
    b4<-b4+lgamma(c2)-c4}
  b<-b1-b3+b4-b2
  #
  c<-log(2*nsisp*nsjsp)-(log(nsi*(nsi-1))+(nsisp+nsjsp-2)*log(1/2))
#
  PA<-exp(a+b+c)
  PA}
#
# perform DDRJ to a mixture of 4 binomials distributions
#
#################
#  Simulation 1. Nt=100 and p_{kk}=0.30
#################
set.seed(1111)
#
Kverd<-4
T<-120
Pbin<-c(0.15,0.25,0.50,0.85)
Y<-numeric()
minNt<-100
maxNt<-100
Nt<-round(runif(T, min = minNt, max = maxNt))
Ptrans<-matrix(c(0.30,0.70/3,0.70/3,0.70/3,0.70/3,0.30,0.70/3,0.70/3,
0.70/3,0.70/3,0.30,0.70/3,0.70/3,0.70/3,0.70/3,0.30),4,4,byrow=TRUE)
Sverd<-numeric()
Sverd[1]<-1
for (i in 2:T) Sverd[i]<-rmultinomial(Ptrans[Sverd[i-1],])
for (i in 1:length(Sverd)) Y[i]<-rbinom(1,Nt[i],Pbin[Sverd[i]])
#
alfa<-1
beta<-1
gama<-1
#
############# starting points
#
S<-rep(1,T)
K<-max(S)
```

```
dependencia<-0
#
Ktotal<-numeric()
dependtotal<-numeric()
indSpMgtotal<-numeric()
indrejtotal<-numeric()
indrejdeptotal<-numeric()
vetordep<-numeric()
vetorK<-numeric()
vetorteta<-numeric()
vetorS<-numeric()
vetorP<-numeric()
#
a0<-somaYt(Y,S,K)
b0<-somaYt(Nt,S,K)
teta<-numeric()
for (j in (1:K)) teta[j]<-posttetak(alfa,beta,a0[j],(b0[j]-a0[j]))
#
if (dependencia==1){
  Nij0<-contanjk(S,K)
  N1j0<-contan0k(S[1],K)
  MT<-matrix(0,K,K)
  for (j in (1:K)) MT[j,]<-postpj(gama,Nij0[j,])
  P0<-postpj(gama,N1j0)}
if (dependencia==0){
  N1j0<-contank(S,K)
  Nij0<-matrix(N1j0,nrow=K,ncol=K,byrow=TRUE)
  vet.prob<-postpj(gama,Nij0[1,])
  MT<-matrix(vet.prob,K,K,byrow=TRUE)
  P0<-vet.prob}
#
amostrasfin<-5000
burnin<-5000
saltos<-10
AmostrasTotal<-burnin+amostrasfin*saltos
#
library(compiler)
enableJIT(3)
```

```
#
for (int in (1:AmostrasTotal)){
  cat('\n', int)
  ############# dependence order proposal
  #
  if (K>1){
  if (dependencia==1){
    candidato<-propind(K,Y,S,Nt,alfa,beta,gama,teta)
    probava<-probaceitdepend(alfa,beta,gama,Y,Nt,candidato[[2]],
S,teta,P0,MT,candidato[[4]])}
  if (dependencia==0){
    candidato<-propdepend(K,Y,S,Nt,alfa,beta,gama,teta)
    probava<-1/probaceitdepend(alfa,beta,gama,Y,Nt,S,candidato[[2]],
teta,candidato[[4]],candidato[[1]],MT[1,])}
  #
  #### evaluate the acceptance of dependence proposal
  #
  aux2<-runif(1)
  if (aux2>=probava){
    indrejdeptotal[int]<-1
    dependtotal[int]<-dependencia}
  if (aux2<probava){
    indrejdeptotal[int]<-0
    dependencia<-candidato[[5]]
    dependtotal[int]<-dependencia
    MT<-candidato[[1]]
    P0<-candidato[[4]]
    S<-candidato[[2]]
  #
    for (atu in 1:10){ # update model 10 times after accepting the
    # dependence proposal
    Snovo<-numeric()
    if (dependencia==1){
      Snovo[1]<-postS1(MT,P0,teta,Y[1],Nt[1],S[2])
      for (t in (2:(T-1))) Snovo[t]<-postSj(MT,teta,Y[t],Nt[t],Snovo[t-1],S[t+1])
      Snovo[T]<-postST(MT,teta,Y[T],Nt[T],Snovo[T-1])} else {
      for (t in (1:T)) Snovo[t]<-postSjind(MT[1,],teta,Y[t],Nt[t])}
    S<-Snovo
    #
    a0<-somaYt(Y,S,K)
    b0<-somaYt(Nt,S,K)
    teta<-numeric()
    for (j in (1:K)) teta[j]<-posttetak(alfa,beta,a0[j],(b0[j]-a0[j]))
    #
    if (dependencia==1){
      Nij0<-contanjk(S,K)
      N1j0<-contan0k(S[1],K)
      MT<-matrix(0,K,K)
      for (j in (1:K)) MT[j,]<-postpj(gama,Nij0[j,])
      P0<-postpj(gama,N1j0)} else {
      N1j0<-contank(S,K)
      Nij0<-matrix(N1j0,K,K,byrow=TRUE)
      P0<-postpj(gama,Nij0[1,])
      MT<-matrix(P0,K,K,byrow=TRUE)}}}}
  #
  ############## proposal of split-merge step
  #
  posicao<-sort(sample(seq(1,T), 2, replace = FALSE))
  pos1<-posicao[1]
  pos2<-posicao[2]
  #
  if(S[pos1]==S[pos2]) indSpMgtotal[int]<-0 else indSpMgtotal[int]<-1
  # indSpMgtotal[int]=0 is split, se indSpMgtotal[int]=1 is merge
  #
  if (indSpMgtotal[int]==1) candidato<-propmerge(K,Y,S,Nt,alfa,beta,gama,pos1,
pos2,teta,dependencia) else
candidato<-propsplit(K,Y,S,Nt,alfa,beta,gama,pos1,pos2,teta,dependencia)
  #
  ########### evaluate the acceptance of split-merge proposal
  #
  if (indSpMgtotal[int]==0){
    sytsi<-a0[S[pos1]]
    sntytsi<-(b0[S[pos1]]-a0[S[pos1]])
    sytsisp<-candidato[[6]]
    sntytsisp<-(candidato[[7]]-candidato[[6]])
    sytsjsp<-candidato[[8]]
    sntytsjsp<-(candidato[[9]]-candidato[[8]])
    if (dependencia==1){
      Nij0<-contanjk(S,K)
      N1j0<-contan0k(S[1],K)
      matrizN<-rbind(N1j0,Nij0)
      matrizNsp<-rbind(candidato[[11]],candidato[[10]])} else {
      matrizN<-matrix(contank(S,K),nrow=1)
      matrizNsp<-matrix(candidato[[11]],nrow=1)}
    nsisp<-sum(matrizNsp[,S[pos1]])
    nsjsp<-sum(matrizNsp[,candidato[[1]]])
    nsi<-sum(matrizN[,S[pos1]])
    PACEI<-min(1,probaceit(alfa,beta,sytsi,sntytsi,sytsisp,sntytsisp,sytsjsp,
sntytsjsp,gama,matrizN,matrizNsp,nsisp,nsjsp,nsi))}
  #
  if (indSpMgtotal[int]==1){
    sytsi<-candidato[[6]]
    sntytsi<-(candidato[[7]]-candidato[[6]])
    sytsisp<-a0[S[pos1]]
    sntytsisp<-(b0[S[pos1]]-a0[S[pos1]])
    sytsjsp<-a0[S[pos2]]
    sntytsjsp<-(b0[S[pos2]]-a0[S[pos2]])
    if (dependencia==1){
      Nij0<-contanjk(S,K)
      N1j0<-contan0k(S[1],K)
      matrizN<-rbind(candidato[[9]],candidato[[8]])
      matrizNsp<-rbind(N1j0,Nij0)} else {
      matrizN<-matrix(candidato[[9]],nrow=1)
      matrizNsp<-matrix(contank(S,K),nrow=1)}
    comp1<-sort(c(S[pos1],S[pos2]))[1]
    nsi<-sum(matrizN[,comp1])
    nsisp<-sum(matrizNsp[,S[pos1]])
    nsjsp<-sum(matrizNsp[,S[pos2]])
    PACEI<-min(1,1/(probaceit(alfa,beta,sytsi,sntytsi,sytsisp,sntytsisp,sytsjsp,
sntytsjsp,gama,matrizN,matrizNsp,nsisp,nsjsp,nsi)))}
  #
  ######## Gibbs steps to update other parameters of the model
  #
  aux2<-runif(1)
```

```
if (aux2<PACEI){
  Ktotal[int]<-candidato[[1]]
  indrejtotal[int]<-0
  MT<-candidato[[4]]
  teta<-candidato[[3]]
  P0<-candidato[[5]]
  S<-candidato[[2]]
  K<-candidato[[1]]
  #
  Snovo<-numeric()
  if (dependencia==1){
    Snovo[1]<-postS1(MT,P0,teta,Y[1],Nt[1],S[2])
    for (t in (2:(T-1))) Snovo[t]<-postSj(MT,teta,Y[t],Nt[t],Snovo[t-1],S[t+1])
    Snovo[T]<-postST(MT,teta,Y[T],Nt[T],Snovo[T-1])} else {
    for (t in (1:T)) Snovo[t]<-postSjind(MT[1,],teta,Y[t],Nt[t])}
  S<-Snovo
  #
  a0<-somaYt(Y,S,K)
  b0<-somaYt(Nt,S,K)
  teta<-numeric()
  for (j in (1:K)) teta[j]<-posttetak(alfa,beta,a0[j],(b0[j]-a0[j]))
  #
  if (dependencia==1){
    Nij0<-contanjk(S,K)
    N1j0<-contan0k(S[1],K)
    MT<-matrix(0,K,K)
    for (j in (1:K)) MT[j,]<-postpj(gama,Nij0[j,])
    P0<-postpj(gama,N1j0)} else {
    N1j0<-contank(S,K)
    Nij0<-matrix(N1j0,K,K,byrow=TRUE)
    P0<-postpj(gama,Nij0[1,])
    MT<-matrix(P0,K,K,byrow=TRUE)}}
#
if (aux2>=PACEI){
  indrejtotal[int]<-1
  Ktotal[int]<-K
  #
  Snovo<-numeric()
```

```
    if (dependencia==1){
      Snovo[1]<-postS1(MT,P0,teta,Y[1],Nt[1],S[2])
      for (t in (2:(T-1))) Snovo[t]<-postSj(MT,teta,Y[t],Nt[t],Snovo[t-1],S[t+1])
      Snovo[T]<-postST(MT,teta,Y[T],Nt[T],Snovo[T-1])} else {
      for (t in (1:T)) Snovo[t]<-postSjind(MT[1,],teta,Y[t],Nt[t])}
    S<-Snovo
    #
    a0<-somaYt(Y,S,K)
    b0<-somaYt(Nt,S,K)
    teta<-numeric()
    for (j in (1:K)) teta[j]<-posttetak(alfa,beta,a0[j],(b0[j]-a0[j]))
    #
    if (dependencia==1){
      Nij0<-contanjk(S,K)
      N1j0<-contan0k(S[1],K)
      MT<-matrix(0,K,K)
      for (j in (1:K)) MT[j,]<-postpj(gama,Nij0[j,])
      P0<-postpj(gama,N1j0)} else {
      N1j0<-contank(S,K)
      Nij0<-matrix(N1j0,K,K,byrow=TRUE)
      P0<-postpj(gama,Nij0[1,])
      MT<-matrix(P0,K,K,byrow=TRUE)}}
  #
  ######### record the samples after burn-in and jumps
  #
  if (int>burnin & int%%saltos==0){
    vetordep<-c(vetordep,dependencia)
    vetorK<-c(vetorK,K)
    vetorteta<-c(vetorteta,teta)
    vetorS<-rbind(vetorS,S)
    mtrans<-rbind(P0,MT)
    a1<-numeric()
    for (i in 1:ncol(mtrans)) a1<-c(a1,mtrans[,i])
    vetorP<-c(vetorP,a1)}
  }
#
########## export the final files
#
```

```
Kmaxobs<-max(vetorK)
matrizteta<-matrix(0,amostrasfin,Kmaxobs)
matrizPtrans<-matrix(0,amostrasfin,Kmaxobs+(Kmaxobs*Kmaxobs))
cont1<-1
cont2<-1
for (i in (1:amostrasfin)){
  for (l in (1:vetorK[i])){
    matrizteta[i,l]<-vetorteta[cont1]
    cont1<-cont1+1}
  for (j in (1:(vetorK[i]+vetorK[i]*vetorK[i]))){
    matrizPtrans[i,j]<-vetorP[cont2]
    cont2<-cont2+1}}
#
soma<-numeric()
for (i in 1:nrow(matrizPtrans)) soma[i]<-sum(matrizPtrans[i,])
round(soma,2)
#
cat(dependtotal,file="/Users/Daiane/Downloads/deptotsp2_100_mat1.txt",append=T)
cat(Ktotal,file="/Users/Daiane/Downloads/Ktotsp2_100_mat1.txt",append=T)
cat(vetordep,file="/Users/Daiane/Downloads/depsp2_100_mat1.txt",append=T)
cat(indrejtotal,file="/Users/Daiane/Downloads/indrejsp2_100_mat1.txt",append=T)
cat(indrejdeptotal,file="/Users/Daiane/Downloads/indrejdepsp2_100_mat1.txt",
append=T)
cat(matrizteta,file="/Users/Daiane/Downloads/tetasp2_100_mat1.txt",append=T)
cat(matrizPtrans,file="/Users/Daiane/Downloads/Psp2_100_mat1.txt",append=T)
cat(vetorS,file="/Users/Daiane/Downloads/Ssp2_100_mat1.txt",append=T)
cat(vetorK,file="/Users/Daiane/Downloads/Ksp2_100_mat1.txt",append=T)
#
```

Chapter 3

QTL mapping as a mixture model [1]

In this chapter, we characterize the multiple QTLs mapping as a mixture model and propose a birth-death-merge data-driven reversible jump (DDRJ) to estimate a model with unknown number of QTLs. We compare the performance of the proposed methodology, usual reversible jump (RJ) and multiple-interval mapping (MIM) using simulated and real data sets.

Compared with RJ, DDRJ shows a better performance to estimate the number of QTLs and their locations on the genome mainly when the QTL effect is moderate, basically as a result of better mixing for transdimensional moves. The inclusion of a merge step of consecutive QTLs in DDRJ is efficient, under tested conditions, to avoid splitting a true QTL's effects with false QTLs and, consequently, selecting a wrong model. DDRJ is also more precise to estimate the QTLs location than MIM in which the number of QTLs need to be specified in advance.

As DDRJ is more efficient to identify and characterize QTLs with smaller effect, this method also appears to be useful and bring contributions to identify SNPs (single nucleotide polymorphism) which usually have small effect on phenotype.

3.1 Introduction

Geneticists and molecular biologists have aimed at locating regions associated with quantitative traits in a chromosome. These chromosomal regions are known as quantitative trait *loci* (QTL) and their location and effects on the phenotypic traits are estimated by genetic markers. The most popular genetic markers are SSR (simple sequence repeats) and SNP (single nucleotide polymorphism), their location is specified by the linkage map and their genotype is known.

[1] This chapter is based on the manuscript "Data-driven reversible jump for QTL mapping" (Zuanetti & Milan, 2016).

A phenotype is usually modeled as a linear function of the additive and dominance effects of the QTL genotypes and several methods have been developed for the localization and characterization of QTLs. The standard estimation method in experimental crosses is the interval mapping (IM) presented by Lander & Botstein (1989) and Haley & Knott (1992). Lander & Botstein (1989) propose use of EM algorithm (Dempster *et al*, 1977), assuming a single putative QTL at each location on the genome and comparing the hypothesis of a single QTL to the null hypothesis of no segregation QTLs by the logarithm of the odds ratio (LOD score). However, the estimate of the QTL effects can be influenced by the effect of other possible QTLs in adjacent regions since this effect is not controlled in the model and nonexisting or ghost QTLs can be identified. A ghost QTL appears when two or more QTLs are linked in coupling (meaning that their effects have the same sign) and the interval mapping gives a maximum LOD score at a location between the two QTLs (Broman & Speed, 1999).

Jansen (1993), Jansen & Stam (1994) and Zeng (1994) propose the composite interval mapping (CIM) to control the effect of QTLs located in adjacent regions and avoid the identification of ghost QTLs. They propose to include in the single putative QTL regression model a subset of markers as cofactors. Kao *et al.* (1999) propose the multiple-interval mapping (MIM) which consider the effect of all possible QTLs and epistatic effect between them in a single model. This model, with a fixed number of QTLs, is estimated by EM algorithm and the number of QTLs is selected by model selection methods as AIC (Akaike information criterion), BIC (Bayesian information criterion), among others.

Bayesian methods for QTL mapping are interesting tools since they allow to select and estimate the model jointly. Earlier Bayesian approaches were proposed by Stephens & Smith (1993) and Satagopan *et al.* (1996). The authors estimate the locations and effect of a prespecified number of QTLs. In practice, however, the number of QTLs is unknown and must be estimated. Satagopan & Yandell (1996) and Stephens & Fisch (1998) propose variants of reversible jump Markov chain Monte Carlo to estimate it and the remaining parameters of the model jointly. An important characteristic in the chain generated in MCMC is that it mixes well, *i.e.*, that it moves around the parameter space rather easily and quickly finds its stationary distribution. Forming good Markov chain and monitoring their behavior is a delicate and sophisticated work (Broman & Speed, 1999).

Over the past decade, different ways to generate proposal parameters in MCMC have been suggested to facilitate the moves between models and accelerate the convergence of the original RJ algorithm. Green & Mira (2001) propose an algorithm that, on rejection, a second attempt to move is made. Regarding the inclusion of a new QTL, Yi & Xu (2002) suggest generating its effects (additive and dominance) from the conditional *a posteriori* distribution. Yi *et al.* (2005) propose updating the location of a specific QTL and its genotypes together. As QTL's location and genotype are correlated, the acceptance probability of a new QTL's location is higher if its genotype is updated jointly.

To accelerate the search procedure of the correct number of QTLs, K, more suitable and efficient dimensional change candidates must be generated. For this purpose, we propose a birth-death-merge data-driven reversible jump (DDRJ) for multiple QTL mapping. It simulates a more likely location for a new QTL using the available data, chooses a QTL to be excluded according to its importance in the current model or merges the effects of two consecutive QTLs if their genotype are correlated. Consequently, candidates are more likely to be accepted and the space of possible models are easily explored. Jain & Neal (2004), Jain & Neal (2007) and Saraiva & Milan (2012) show that data-driven methods are effective in simplifying the methodology and improving the chain mixing.

The merge movement of consecutive QTLs is efficient under tested conditions to avoid identification of false QTLs. Usually, as close QTLs have similar estimated genotype, the effects of a true QTL are split between two QTLs and bias the estimate of the number of QTLs and their effects. Split QTLs can be seen as the opposite problem of ghost QTLs.

The proposed method also has the advantage of providing intervalar estimates about the uncertainty of estimates. Usual methods generally provide only point estimates or asymptotic confidence intervals for big samples.

The chapter is organized as follows: Section 3.2 presents a model for quantitative traits and discusses the likelihood function; Section 3.3 addresses the Bayesian approach for the model including the DDRJ procedure to estimate the number of QTLs; Section 3.4 analyzes the performance of the DDRJ and compares it with the RJ and MIM performance in simulated and real data sets. Finally, Section 3.5 draws discussions and Section 3.6 shows the appendices.

3.2 Model for quantitative traits

Let $\mathbf{y} = (y_1, y_2, ..., y_n)$ be a quantitative trait of n individuals from an F2 population. Assume this phenotype has been affected by K QTLs located at positions $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_K)$, $\lambda_k < \lambda_{k+1}$ for $k = 1, ..., K-1$, between m different genotyped markers with a known linkage map.

Phenotype y_i for the i-th individual can be modeled by

$$y_i = \mu + \sum_{k=1}^{K} \alpha_k Q_{ik} + \sum_{k=1}^{K} \delta_k (1 - |Q_{ik}|) + \varepsilon_i, \tag{3.1}$$

where μ is the average of expected values of genotypes AA and aa, α_k is the additive effect of the k-th QTL, δ_k is the dominance effect of k-th QTL, Q_{ik} represents the genotype of k-th QTL of the i-th individual coded as $-1, 0$ or 1 for aa, Aa or AA, respectively, $k = 1, ..., K$ and $i = 1, 2, ..., n$, $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$ is the random error, and ε_i and $\varepsilon_{i'}$ are supposed to be independent for $i \neq i'$.

The phenotype can also be affected by environmental covariates and interactions among

QTLs or between covariates and QTLs. The model defined by eq. (3.1) does not consider these effects, but extensions (modeling environmental covariates as fixed effects, for example) are straightforward.

The data set consist of $\mathbf{y} = (y_1, y_2, ..., y_n)$ - the observations regarding the quantitative trait of n individuals, $\mathbf{M}_{(n\times m)}$ - the markers genotype coded as $-1, 0$ or 1 for aa, Aa or AA, respectively, and $\mathbf{D} = \{D_1, D_2, ..., D_m\}$ - the distances (in centiMorgans - cM) between each marker and the first marker, where $D_1 = 0$.

We assume there is at most one QTL between two consecutive markers, therefore $K < m$, and the QTL's genotype is explained only by flanking markers, *i.e.*, $Q_{ik}|M_{ir_k}, M_{il_k}$ and $Q_{ik'}|M_{ir_{k'}}, M_{il_{k'}}$ are independent for $k \neq k'$, where M_{ir_k} is the genotype of the marker to the right of the k-th QTL for the i-th individual and M_{il_k} is the genotype of the marker to the left of the k-th QTL for the i-th individual.

The joint probability distribution of $\mathbf{y}$ and $\mathbf{Q}$, where $\mathbf{Q} = \{Q_{ik}\}$ is the matrix of the K QTLs genotype for the n individuals is

$$f_{\mathbf{Y},\mathbf{Q}|\mathbf{M},\mathbf{D}}(\mathbf{y},\mathbf{q}) = \prod_{i=1}^{n} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i, \mathbf{D}\right), \tag{3.2}$$

where $Pr\left(Q_{i1} = q_{i1}, ..., Q_{iK} = q_{iK}|\mathbf{M}_i, \mathbf{D}\right) = \prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}\right)$, for $i = 1, ..., n$, $\sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}\right) = 1$, for $q_{ik} = -1, 0, 1$, and f is the conditional normal density for Y_i.

As $\mathbf{Q}$ are nonobservable variables, the marginal probability distribution of $\mathbf{y}$ is

$$\begin{aligned} f_{\mathbf{Y}|\mathbf{M},\mathbf{D}}(\mathbf{y}) &= \sum_{\mathbf{q}} f_{\mathbf{Y},\mathbf{Q}|\mathbf{M},\mathbf{D}}(\mathbf{y},\mathbf{q}) \\ &= \prod_{i=1}^{n} \sum_{q_{i1}} \cdots \sum_{q_{iK}} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(Q_{i1} = q_{i1}, ..., Q_{iK} = q_{iK}|\mathbf{M}_i, \mathbf{D}\right), \end{aligned} \tag{3.3}$$

where the sum over $\mathbf{q}$ is over the 3^K possible QTLs genotypes for i-th individual and $\sum_{q_{i1}} \cdots \sum_{q_{iK}} Pr\left(Q_{i1} = q_{i1}, ..., Q_{iK} = q_{iK}|\mathbf{M}_i, \mathbf{D}\right) = \prod_{k=1}^{K} \sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}\right) = 1$. Eq. (3.3) characterizes variable Y_i, $i = 1, ..., n$, as a mixture of 3^K distributions.

In practice, the number of QTLs K is unknown and the parameters of the model are $\boldsymbol{\theta} = (K, \boldsymbol{\lambda}, \mu, \boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K), \boldsymbol{\delta} = (\delta_1, ..., \delta_K), \sigma^2)$. The likelihood function of $\boldsymbol{\theta}$ given $\mathbf{y}$ and $\mathbf{Q} = \mathbf{q}$ is

$$L(\boldsymbol{\theta}|\mathbf{y},\mathbf{q}) = \left(2\pi\sigma^2\right)^{-n/2} exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^2\right\} \prod_{i=1}^{n}\prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}\right), \tag{3.4}$$

where $\epsilon_i = y_i - \mu - \sum_{k=1}^{K} \alpha_k q_{ik} - \sum_{k=1}^{K} \delta_k (1 - |q_{ik}|)$ is the residual of the i-th observation and

$Pr\left(Q_{ik}=q_{ik}|M_{ir_k},M_{il_k},\mathbf{D}\right)$ is the conditional probability of QTL genotype given the flanking marker genotypes as defined by Stephens & Fisch (1998). Such a probability is a function of recombination fractions between the k-th QTL and its flanking markers calculated by the Haldane distance function. Note that Q_{ik}, $i=1,...,n$ and $k=1,...,K$, is nonobservable and must be estimated.

Without loosing the generality and for simplicity, consider the models with one and two QTLs defined, respectively, as

$$y_i=\mu+\alpha_1 Q_{i1}+\delta_1(1-|Q_{i1}|)+\varepsilon_i\ (M_1)\text{ and}$$
$$y_i=\mu+\alpha_1^{'}Q_{i1}^{'}+\alpha_2^{'}Q_{i2}^{'}+\delta_1^{'}(1-|Q_{i1}^{'}|)+\delta_2^{'}(1-|Q_{i2}^{'}|)+\varepsilon_i\ (M_2),$$

for $i=1,...,n$. Observe if $Q_{i1}^{'}=Q_{i2}^{'}=Q_{i1}$ for all or almost all individuals, $\alpha_1^{'}+\alpha_2^{'}=\alpha_1$ and $\delta_1^{'}+\delta_2^{'}=\delta_1$, the models M_1 and M_2 are equally or almost equally likely and can be hard to select the correct model in this situation. The genotype of two *loci* has a high probability of being equal when they are close on the same chromosome and the model is wrongly estimated if the effect of two or more true close QTLs are merged in only one QTL or if the effect of one true QTL is split with one or more false close QTLs. We note in our simulated data sets, some of them are shown in Section 3.4, and using multiple QTLs methods to estimate the model that often methods split the effect of one true QTL with one or more false QTLs. Conventional methodologies for QTL mapping often do not deal well with this problem.

3.3 Bayesian approach

The usual Bayesian methodology for models with unknown K is the RJ proposed by Green (1995). This method consists in running Metropolis-Hastings steps that either accepts or rejects different moves, like "birth" or "death" of a QTL. These steps enable transitions from the current model to models of higher or lower dimensions.

Parameters $\boldsymbol{\lambda}|K$, $\boldsymbol{\alpha}|K$, $\boldsymbol{\delta}|K$, μ, σ^2 and elements of $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ are supposed to be independent and the joint *a priori* density for $\boldsymbol{\theta}$ is written as

$$\pi(\boldsymbol{\theta})=\pi(K)\pi(\boldsymbol{\lambda}|K)\left(\prod_{k=1}^{K}\pi(\alpha_k)\pi(\delta_k)\right)\pi(\mu)\pi(\sigma^2). \tag{3.5}$$

Particularly, we consider

1. $K\sim$ Uniform$(0,1,...,m-1)$;

2. $\alpha_k\sim$ Normal$(\nu_\alpha,\sigma_\alpha^2)$, $k=1,...,K$, where ν_α and $\sigma_\alpha^2>0$ are known hyper-parameters;

3. $\delta_k\sim$ Normal$(\nu_\delta,\sigma_\delta^2)$, $k=1,...,K$, where ν_δ and $\sigma_\delta^2>0$ are known hyper-parameters;

4. $\mu\sim$ Normal(ν_μ,σ_μ^2), where ν_μ and $\sigma_\mu^2>0$ are known hyper-parameters;

5. $\sigma^2 \sim$ Inverse-gamma(η_a, η_b), where $\eta_a > 0$ and $\eta_b > 0$ are known hyper-parameters; and

6. $\pi(\boldsymbol{\lambda}|K) = \pi(\lambda_1, ..., \lambda_K|K) = \pi(\lambda_1|K)\pi(\lambda_2|\lambda_1, K)\ldots\pi(\lambda_K|\lambda_{K-1}, K)$. If there is no *a priori* information about the QTL's location, each location is assumed uniformly distributed over the possible *loci*.

   As we have assumed any marker interval contains at most one QTL, we can define

   - $\lambda_1|K \sim$ Uniform$(D_1, D_{m-(K-1)})$, therefore we guarantee there are more $(K-1)$ marker intervals to allocate the remaining $(K-1)$ QTLs;
   - $\lambda_2|\lambda_1, K \sim$ Uniform$(D_{r_1}, D_{m-(K-2)})$, therefore we guarantee there are more $(K-2)$ marker intervals to allocate the remaining $(K-2)$ QTLs. D_{r_1} is the element of $\mathbf{D}$ that represents the position of the marker to the right of 1-st QTL;
   - $\lambda_k|\lambda_{k-1}, K$ distribution, for $k = 3, ..., K-1$, is defined as in the above item and;
   - $\lambda_K|\lambda_{K-1}, K \sim$ Uniform$(D_{r_{K-1}}, D_m)$, where $D_{r_{K-1}}$ is the element of $\mathbf{D}$ that represents the position of the marker to the right of $(k-1)$-th QTL.

Combining the likelihood function in equation (3.4) with the *a priori* distributions, we obtain the conditional *a posteriori* distributions of $\mu|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\mu})$, $\sigma^2|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\sigma^2})$, $\alpha_k|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\alpha_k})$, $\delta_k|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\delta_k})$, provided in the appendices of this chapter, and $\lambda_k|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\lambda_k})$, $k = 1, ..., K$.

The nonobservable genotype q_{ik}, $i = 1, ..., n$ and $k = 1, ..., K$, is simulated and updated by its conditional *a posteriori* distribution given by

$$\begin{aligned} Pr(Q_{ik} = q_{ik}|\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D}) &\propto Pr(Q_{ik} = q_{ik}, y_i|\mathbf{q}_{-q_{ik}}, M_{ir_k}, M_{il_k}, \mathbf{D}) \\ &= f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}), \end{aligned} \qquad (3.6)$$

for $q_{ik} \in \{-1, 0, 1\}$ and where $f_{Y_i|\mathbf{q}_i}(y_i)$ is the Normal$\Big(\mu + \sum_{k=1}^{K}\alpha_k q_{ik} + \sum_{k=1}^{K}\delta_k(1-|q_{ik}|), \sigma^2\Big)$ density function.

From equation (3.6), $Q_{ik}|(\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D}) \sim$ Multinomial$(1, (p_{ik-1}, p_{ik0}, p_{ik1}))$, where

$$p_{ikj} = \frac{f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j|M_{ir_k}, M_{il_k}, \mathbf{D})}{\sum_j f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j|M_{ir_k}, M_{il_k}, \mathbf{D})},$$

$j = -1, 0, 1$.

Parameters μ, σ^2, α_k, δ_k and nonobservable values q_{ik}, $i = 1, ..., n$ and $k = 1, ..., K$, are updated by Gibbs sampling steps and λ_k is updated jointly with $\mathbf{q}_k$ by Metropolis-Hastings steps in which $\lambda_k^{'}$ is sampled from a Uniform(D_{l_k}, D_{r_k}) distribution and the block $(\lambda_k^{'}, \mathbf{q}_k^{'})$ is accepted according to probability $\Psi((\lambda_k^{'}, \mathbf{q}_k^{'})|(\lambda_k, \mathbf{q}_k)) = min(1, A)$, where

$$\begin{aligned} A = & \frac{exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^{'2}\right\}}{exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^{2}\right\}} \frac{\prod_{i=1}^{n} Pr\left(Q_{ik} = q_{ik}^{'}|M_{ir_k}, M_{il_k}, \mathbf{D}\right)}{\prod_{i=1}^{n} Pr\left(Q_{ik} = q_{ik}|M_{ir_k}, M_{il_k}, \mathbf{D}\right)} \\ & \times \frac{\prod_{i=1}^{n} Pr(Q_{ik} = q_{ik}|\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D})}{\prod_{i=1}^{n} Pr(Q_{ik} = q_{ik}^{'}|\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D})}, \end{aligned} \tag{3.7}$$

$\epsilon_i = y_i - \mu - \sum_{k=1}^{K}\alpha_k q_{ik} - \sum_{k=1}^{K}\delta_k(1-|q_{ik}|)$ is the residual of the i-th individual, $i = 1, ..., n$, and $\epsilon_i^{'}$ is calculated using $\mathbf{q}_{-\mathbf{q}_k}$ and $\mathbf{q}_k^{'}$.

3.3.1 DDRJ

The movements that change K are called birth (b), death (d) or merge (mg) moves when a new QTL is included in the model or, conversely, one QTL is excluded from the current model or the effect of two QTLs are summed into a single QTL. The birth, death and merge moves are implemented by Metropolis-Hastings steps and either increase or reduce the number of QTL by one at each step.

Consider $x = (\mathbf{q}, \boldsymbol{\theta})$ the current state of MCMC procedure with K QTLs and $x^{'} = (\mathbf{q}^{'}, \boldsymbol{\theta}^{'})$ the proposed movement, where $'$ means either a birth (b), a death (d) or a merge (mg) of QTLs. Therefore, $K^{'} = K + 1$ if a birth movement is proposed or $K^{'} = K - 1$ if a death or a merge movement is proposed. This move is accepted according to Metropolis-Hastings probability $\Psi(x^{'}|x) = min(1, A^{'})$, where

$$A^{'} = \frac{L(\boldsymbol{\theta}^{'}|\mathbf{y}, \mathbf{q}^{'})}{L(\boldsymbol{\theta}|\mathbf{y}, \mathbf{q})} \frac{\pi(\boldsymbol{\theta}^{'})}{\pi(\boldsymbol{\theta})} \frac{q(x|x^{'})}{q(x^{'}|x)}, \tag{3.8}$$

and $q(\cdot|\cdot)$ is the transition function, described below.

At each step, we choose a movement to increase or reduce the number of QTLs as follows:

1. If $0 < K < m - 1$, a birth or a death is randomly chosen, according to their probability. Here, we assume $Pr(b|K) = 1/2$ and $Pr(d|K) = 1/2$;

2. If $K = 0$, a birth is chosen, *i.e.*, $Pr(b|K) = 1$; and

3. If $K = m - 1$, a death is chosen, *i.e.*, $Pr(d|K) = 1$.

Birth proposal

When a birth movement is chosen, a location is selected for the new QTL in a marker interval that has no QTL and its genotype and effect parameters must be defined. The selection of a location through a Uniform distribution can be inefficient, mainly if we have a large number of marker intervals.

If there is a strong association between a marker and a trait, it is reasonable suppose there is a QTL nearby that marker. Therefore, the association between markers and trait can be used to guide the search for new QTLs in the estimation process. As each marker can be seen as a factor with three levels affecting differently the phenotype mean or the residual mean of the current model, we use the Kruskal-Wallis test statistic to measure this association. The F-statistics in a one-way analysis of variance could also be used. Higher values indicate the residual mean is different for the distinct levels of the marker and there is a higher chance of a QTL close to it whose effect is not considered in the current model. Values close to zero indicate the residual mean is the same for all levels of the marker and its contribution to explain the quantitative trait is not relevant or its effect is already considered in the model.

The complete birth step is built as follows:

1. Select a marker to allocate the new QTL from a Multinomial$(1, (pb_1, ..., pb_m))$, where $pb_j = \frac{KW_j}{\sum_{j=1}^{m} KW_j}$, $j = 1, ..., m$, and KW_j is the statistics of the Kruskal-Wallis test from residuals of the current model and j-th marker genotype, defined as

   $$KW_j = (n-1)\frac{\sum_{l=1}^{3} n_l(\bar{r}_{l\cdot} - \bar{r})^2}{\sum_{l=1}^{3}\sum_{i=1}^{n_l}(r_{li} - \bar{r})^2},$$

   where n_l is the number of individuals in l-th group and the three groups are specified by the genotype of j-th marker, r_{li} is the rank (among all individuals) of i-th individual from l-th group, $\bar{r}_l = \sum_{i=1}^{n_l} r_{li}/n_l$ and $\bar{r} = 0.5(n+1)$ is the average of all the r_{li}.

   Note that markers which most affect the residual mean are more likely to be chosen;

2. Assume j^*-th marker has been chosen, $j^* \neq 1$ and $j^* \neq m$, and suppose there is no QTL between (j^*-1) and (j^*+1)-th markers. The new QTL can be located in $[D_{j^*-1}, D_{j^*+1}]$ and λ_{K+1} is defined as $D_{j^*-1} + (D_{j^*+1} - D_{j^*-1}) * Z$, where $Z \sim$ Beta$(a, 1)$ and a is calculated according to

   $$E[Z] = \frac{\sum_{j=(j^*-1)}^{(j^*+1)} \frac{D_j - D_{j^*-1}}{D_{j^*+1} - D_{j^*+1}} KW_j}{\sum_{j=(j^*-1)}^{(j^*+1)} KW_j}, \; i.e., \; a = \frac{E[Z]}{1-E[Z]}.$$

   Consequently, the expected value of λ_{K+1} is the average of j^*-th marker and its flanking markers' position weighted by their effect on the residual mean of the current model and the new QTL is more likely to be close to the marker that is most relevant effect on the residual mean. Note Beta$(a, 1)$ distribution is Uniform$(0, 1)$ when M_{j^*-1}, M_{j^*} and M_{j^*+1} have the same effect on the residual mean and j^*-th marker is in the middle of $[D_{j^*-1}, D_{j^*+1}]$.

   If $j^* = 1$, $j^* = m$, $[D_{j^*-1}, D_{j^*}]$ or $[D_{j^*}, D_{j^*+1}]$ already contains a QTL, the new QTL will be located in $[D_1, D_2]$, $[D_{m-1}, D_m]$, $[D_{j^*}, D_{j^*+1}]$ or $[D_{j^*-1}, D_{j^*}]$, respectively, and its position is simulated as in step 2, considering only two markers and not three;

3. Sample genotype of the new QTL for all individuals, $\mathbf{q}_{K+1}$, from $Pr\left(Q_{iK+1} = q_{iK+1} | M_{ir_{K+1}}, M_{il_{K+1}}, \mathbf{D}\right)$;

4. Sample α_{K+1} from its conditional *a posteriori* distribution considering $\mathbf{q}^b = (\mathbf{q}, \mathbf{q}_{K+1})$ and $\delta_{K+1} = 0$;

5. Sample δ_{K+1} from its conditional *a posteriori* distribution considering $\mathbf{q}^b$ and $\boldsymbol{\alpha}^b = (\boldsymbol{\alpha}, \alpha_{K+1})$;

6. Sample μ^b from its conditional *a posteriori* distribution considering $\mathbf{q}^b$, $\boldsymbol{\alpha}^b$ and $\boldsymbol{\delta}^b = (\boldsymbol{\delta}, \delta_{K+1})$ and;

7. Sample σ^{2^b} from its conditional *a posteriori* distribution considering $\mathbf{q}^b$, $\boldsymbol{\alpha}^b$, $\boldsymbol{\delta}^b$ and μ^b.

Therefore, we have a new set of QTL genotypes and parameters $x^b = (\mathbf{q}^b, \boldsymbol{\theta}^b)$. This transition proposal is denoted by $x^b|x$ and its probability is

$$\begin{aligned} q(x^b|x) = {} & Pr(b|K) pb_{j^*} f_Z(z) \prod_{i=1}^{n} \left(Pr\left(Q_{iK+1} = q_{iK+1} | M_{ir_{K+1}}, M_{il_{K+1}}, \mathbf{D}\right)\right) \\ & \times \pi\left(\alpha_{K+1} | \mathbf{y}, \mathbf{q^b}, \boldsymbol{\theta}_{-K}, K+1, \lambda_{K+1}, \delta_{K+1}\right) \pi\left(\delta_{K+1} | \mathbf{y}, \mathbf{q}^b, \boldsymbol{\theta}_{-K}, K+1, \lambda_{K+1}, \alpha_{K+1}\right) \\ & \times \pi\left(\mu^b | \mathbf{y}, \mathbf{q^b}, \boldsymbol{\theta}^b_{-(\mu^b, \sigma^{2^b})}, \sigma^2\right) \pi\left(\sigma^{2^b} | \mathbf{y}, \mathbf{q^b}, \boldsymbol{\theta}^b_{-\sigma^{2^b}}\right), \end{aligned} \tag{3.9}$$

where $\pi\left(\cdot|\cdot\right)$ is the conditional *a posteriori* distribution for each parameter used to sample the candidate-values. The acceptance probability for the birth move is $\Psi(x^b|x) = min(1, A^b)$, where A^b is given by equation (3.8). The probability of the transition proposal denoted by $x|x^b$ is

$$q(x|x^b) = Pr(d|K+1) pd_{K+1} \pi\left(\mu | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-(\mu, \sigma^2)}, \sigma^{2^b}\right) \pi\left(\sigma^2 | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\sigma^2}\right). \tag{3.10}$$

Death proposal

Since a death move has been selected, we choose a QTL from the current model to be deleted.

As Q_{ik} assumes only values -1, 0 and 1 and $(1 - |Q_{ik}|)$ assumes only 0 and 1, for $i = 1, ..., n$ and $k = 1, ..., K$, the current absolute value of α_k and δ_k shows the importance and significance of the k-th QTL, *i.e.*, higher absolute values of α_k or δ_k indicate the k-th QTL is more relevant to explain the phenotype. The current values of these parameters are useful for the choice of the QTL to be excluded without changing significantly the predictive power of the model.

Instead of selecting a QTL to be excluded from a Uniform$(1, ..., K)$, we select it from a Multinomial$(1, (pd_1, ..., pd_k))$, where $pd_k = \frac{\frac{1}{|\alpha_k| + |\delta_k|}}{\sum_{k=1}^{K} \frac{1}{|\alpha_k| + |\delta_k|}}$, for $k = 1, ..., K$, *i.e.*, QTLs that exert the strongest effects and are the most relevant to the model are less likely to be selected and deleted. Therefore, the acceptance probability of the death movement is improved.

The complete death step is as follows:

1. Select the QTL to be excluded from Multinomial$(1, (pd_1, ..., pd_K))$, the k^*-th QTL;
2. Delete $\mathbf{q}_k^*$, λ_k^*, α_k^* and δ_k^* from $\mathbf{q}$, $\boldsymbol{\lambda}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$, respectively;
3. Sample μ^d from its conditional *a posteriori* distribution considering only $K-1$ QTLs and;
4. Sample σ^{2^d} from its conditional *a posteriori* distribution considering the reduced model.

We have a new set of QTL's genotypes and parameters $x^d = (\mathbf{q}^d, \boldsymbol{\theta}^d = (K-1, \boldsymbol{\lambda}^d, \boldsymbol{\alpha}^d, \boldsymbol{\delta}^d, \mu^d, \sigma^{2^d}))$. This transition proposal is denoted by $x^d|x$ and its probability is

$$q(x^d|x) = Pr(d|K) pd_{k^*} \pi \left(\mu^d | \mathbf{y}, \mathbf{q}^d, \boldsymbol{\theta}^d_{-(\mu^d, \sigma^{2^d})}, \sigma^2\right) \pi \left(\sigma^{2^d} | \mathbf{y}, \mathbf{q}^d, \boldsymbol{\theta}^d_{-\sigma^{2^d}}\right), \tag{3.11}$$

where $\pi(\cdot|\cdot)$ is the conditional *a posteriori* distribution of each parameter used to generate the candidate-values.

The acceptance probability for the death movement is $\Psi(x^d|x) = min(1, A^d)$, where $A^d = 1/A^b$ with some suitable substitutions. The probability of transition proposal denoted by $x|x^d$ is defined as

$$\begin{aligned} q(x|x^d) = {} & Pr(b|K-1) \left(pb_{l_{k^*}} f_Z \left(\frac{\lambda_{k^*} - D_{l_{k^*}-1}}{D_{l_{k^*}+1} - D_{l_{k^*}-1}} \right) + pb_{r_{k^*}} f_Z \left(\frac{\lambda_{k^*} - D_{r_{k^*}-1}}{D_{r_{k^*}+1} - D_{r_{k^*}-1}} \right) \right) \\ & \times \prod_{i=1}^{n} \left(Pr\left(Q_{ik^*} = q_{ik^*} | M_{ir_{k^*}}, M_{il_{k^*}}, \mathbf{D}, \lambda_{k^*} \right) \right) \\ & \times \pi\left(\alpha_{k^*} | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}^d_{-(K-1)}, K, \lambda_{k^*}, \delta_{k^*}\right) \pi\left(\delta_{k^*} | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}^d_{-(K-1)}, K, \lambda_{k^*}, \alpha_{k^*}\right) \\ & \times \pi\left(\mu | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-(\mu, \sigma^2)}, \sigma^{2^d}\right) \pi\left(\sigma^2 | \mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\sigma^2}\right), \end{aligned} \tag{3.12}$$

where l_{k^*} is the marker on the left of k^*-th QTL and r_{k^*} is the marker on the right of k^*-th QTL.

Note that if we first choose a birth movement in state x, giving x^b, and then choose the death of $(K+1)$-th QTL, we can recover x and state x is likely to be recovered after a birth process of x^d. If the candidate movement is not accepted, the chain remains in the current model, the value of K does not change and the remaining parameters of the model are updated by Metropolis-Hastings or Gibbs steps.

Merge proposal

Instead of proposing a data-driven with only birth and death steps, we also include a merge movement in the procedure since the model can be wrongly estimated if the effect of a true QTL is split between two or more false QTLs. The split of a QTL may happen if a QTL appears very close to an existent QTL and, as their genotype are very similar, both are in the model and split the additive and dominance effect which would be of only one QTL. The death of one of these QTLs is not generally accepted since the effects of both QTLs are relevant to explain the phenotype variability. The merge moves of two consecutive QTLs is usually accepted and effective to avoid split QTLs since the effects of the QTL that is removed from the model is added to the effect of an adjacent QTL and the predictive power of the model does not change significantly.

For merging two QTLs we must choose a pair of consecutive QTLs to be merged and choose one QTL to be removed from the model. Its effects are added to the effect of the other QTL. We propose to build a data-driven merge candidate as follows:

1. Select a pair of consecutive QTLs to be merged from Multinomial$(1, (pmg_{12}, pmg_{23}, ..., pmg_{(K-1)K}))$, where $pmg_{kj} = \frac{V_{kj}}{\sum_{k=1}^{K-1}\sum_{j=k+1}^{K} V_{kj}}$, $k = 1, ..., K-1$ and $j = k+1, ..., K$, V_{kj} is the Cramér's V measure of association between the genotype of k-th and j-th QTLs. Note that pairs of successive QTLs with more associated genotypes have higher probability to be merged since the split happens between QTLs with similar genotype. Suppose the pair of QTLs k^* and k^*+1 has been selected;

2. Choose the k^*-th or (k^*+1)-th to be excluded from the current model, according to $pd_k = \frac{\frac{1}{|\alpha_k|+|\delta_k|}}{\sum_{k=k^*}^{k^*+1}\frac{1}{|\alpha_k|+|\delta_k|}}$, $k = k^*, k^*+1$. Consider (k^*+1)-th has been chosen to be excluded;

3. Delete $\mathbf{q}_{k^*+1}$, λ_{k^*+1}, α_{k^*+1} and δ_{k^*+1} from $\mathbf{q}$, $\boldsymbol{\lambda}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$, respectively;

4. Update α_{k^*}, δ_{k^*}, μ and σ^2, successively, from their conditional *a posteriori* distribution considering $\mathbf{q}^{mg}$, $\boldsymbol{\alpha}^{mg}$ and $\boldsymbol{\delta}^{mg}$ with $k-1$ QTLs.

Instead of adding the value of α_{k^*+1} and δ_{k^*+1} to α_{k^*} and δ_{k^*}, respectively, we propose to update α_{k^*} and δ_{k^*} from their conditional *a posteriori* probability using the reduced model. It is equivalent since we remove the effects of (k^*+1)-th QTL from the current model to update α_{k^*} and δ_{k^*} and simplify the calculation of merge acceptance probability since is not necessary to define deterministic transformations to reduce the dimension of the model.

We have a new set of QTL's genotypes and parameters $x^{mg} = (\mathbf{q}^{mg}, \boldsymbol{\theta}^{mg} = (K-1, \boldsymbol{\lambda}^{mg}, \boldsymbol{\alpha}^{mg}, \boldsymbol{\delta}^{mg}, \mu^{mg}, \sigma^{2mg}))$. This transition proposal is denoted by $x^{mg}|x$ and its probability is

$$\begin{aligned}
q(x^{mg}|x) = {} & pmg_{k^*(k^*+1)} pd_{k^*+1} \pi\left(\alpha_{k^*}|\mathbf{y}, \mathbf{q}^{mg}, K-1, \boldsymbol{\lambda}^{mg}, \boldsymbol{\alpha}^{mg}_{-\alpha_{k^*}}, \boldsymbol{\delta}^{xmg}, \mu, \sigma^2\right) \\
& \times \pi\left(\delta_{k^*}|\mathbf{y}, \mathbf{q}^{mg}, K-1, \boldsymbol{\lambda}^{mg}, \boldsymbol{\alpha}^{mg}, \boldsymbol{\delta}^{mg}_{-\delta_{k^*}}, \mu, \sigma^2\right) \pi\left(\mu^{mg}|\mathbf{y}, \mathbf{q}^{\mathbf{mg}}, \boldsymbol{\theta}^{mg}_{-(\mu^{mg}, \sigma^{2mg})}, \sigma^2\right) \\
& \times \pi\left(\sigma^{2mg}|\mathbf{y}, \mathbf{q}^{\mathbf{mg}}, \boldsymbol{\theta}^{mg}_{-\sigma^{2mg}}\right), \qquad (3.13)
\end{aligned}$$

where $\pi\left(\cdot|\cdot\right)$ is the conditional *a posteriori* distribution of each parameter used to sample the candidate-values.

The acceptance probability for the merge movement is $\Psi(x^{mg}|x) = min(1, A^{mg})$, where A^{mg} is defined by eq. (3.8). The probability of transition proposal denoted by $x|x^{mg}$ which represents a split of k^*-th QTL is defined as

$$\begin{aligned} q(x|x^{mg}) = & \left(pb_{l_{k^*+1}} f_Z\left(\frac{\lambda_{k^*+1} - D_{l_{k^*+1}-1}}{D_{l_{k^*+1}+1} - D_{l_{k^*+1}-1}}\right) + pb_{r_{k^*+1}} f_Z\left(\frac{\lambda_{k^*+1} - D_{r_{k^*+1}-1}}{D_{r_{k^*+1}+1} - D_{r_{k^*+1}-1}}\right)\right) \\ & \times \prod_{i=1}^{n}\left(Pr\left(Q_{ik^*+1} = q_{ik^*+1}|M_{ir_{k^*+1}}, M_{il_{k^*+1}}, \mathbf{D}, \lambda_{k^*+1}\right)\right) \\ & \times \pi\left(\alpha_{k^*+1}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}^{mg}_{-(K-1)}, K, \lambda_{k^*+1}, \delta_{k^*+1} = 0\right)\pi\left(\delta_{k^*+1}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}^{mg}_{-(K-1)}, K, \lambda_{k^*+1}, \alpha_{k^*+1}\right) \\ & \times \pi\left(\alpha_{k^*}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\alpha_{k^*}}\right)\pi\left(\delta_{k^*}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\delta_{k^*}}\right) \\ & \times \pi\left(\mu|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-(\mu,\sigma^2)},\sigma^{2^{mg}}\right)\pi\left(\sigma^2|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\sigma^2}\right), \end{aligned} \tag{3.14}$$

where l_{k^*+1} is the marker on the left of (k^*+1)-th QTL and r_{k^*+1} is the marker on the right of (k^*+1)-th QTL.

Since we include the QTL merge move only to avoid split QTLs, we do not include a QTL split step in this procedure. However, a split step could be easily included in the algorithm using the transition function of a split movement $q(x^{sp}|x) = q(x|x^{mg})$ defined in eq. (3.14).

Algorithm

The birth-death-merge DDRJ is specified as follows:

1. Initialize a configuration for $\boldsymbol{\theta}$ and $\mathbf{q}$.

2. For l-th iteration, $l = 1, ..., L$, do:

   (a) Choose a death or birth movement.

   (b) Generate the candidate-values of $x^{'}$.

   (c) Accept the proposal with probability $\Psi(x^{'}|x)$, where $'$ means either b or d.

      i. If a birth movement has been accepted, do $K^{(l)} = K^{(l-1)} + 1$ and consider x^b.

      ii. If a death movement has been accepted, do $K^{(l)} = K^{(l-1)} - 1$ and consider x^d.

      iii. If no movement has been accepted, do $K^{(l)} = K^{(l-1)}$ and consider x.

   (d) If $K^{(l)} \geq 2$, generate and evaluate the acceptance of a QTLs merge candidate. If a merge movement has been accepted, do $K^{(l)} = K^{(l)} - 1$ and consider x^{mg}.

   (e) Update λ_k, $k = 1, ..., K^{(l)}$.

   (f) Update q_{ik}, $i = 1, ..., n$ and $k = 1, ..., K^{(l)}$, from its conditional *a posteriori* distribution.

(g) Update α_k and δ_k, $k = 1, ..., K^{(l)}$, from their conditional *a posteriori* distributions.

(h) Update μ from its conditional *a posteriori* distribution.

(i) Update σ^2 from its conditional *a posteriori* distribution.

This algorithm is implemented in R language and the codes are available in the appendices of this chapter. R is a free software environment for statistical computing and graphics and more details are found in its homepage `https://www.r-project.org`'.

3.4 **Applications**

We apply the proposed method to simulated and real data sets and compare the performance of the RJ, DDRJ and MIM methodologies. Although the computational efficiency is an important feature of the methods, we focus in analyzing and comparing their performance in selecting and estimating the correct model. We set hyper-parameters $\nu_\alpha = \nu_\delta = \nu_\mu = 0$, $\sigma^2_\alpha = \sigma^2_\delta = \sigma^2_\mu = 100$ and $\eta_a = \eta_b = 0.1$. This set up provides *a priori* distributions with large variability and weak information about the parameters.

3.4.1 Simulated data sets

We simulate a high dimension linkage map with 450 *loci* which are allocated on a large chromosome of 450 cM (average distance between the *locus* is 1 cM) and their genotype for an F2 population of 300 individuals by QTL Cartographer 2.5 software available on `http://statgen.ncsu.edu/qtlcart/WQTLCart.htm` (Basten *et al*, 1997). We choose $K = 5$ *loci* located at $\boldsymbol{\lambda} = \{15.0, 82.4, 299.8, 363.1, 391.1\}$ to be the QTLs and simulate the phenotype using $\boldsymbol{\alpha} = (-0.60, 0.90, 0.25, -0.40, 0.40)$, $\boldsymbol{\delta} = (0.30, 0.05, -0.25, 0.15, -0.15)$, $\mu = 20$ and three values of σ $(0.5, 1.0, 1.5)$. The effect of first and second QTLs are stronger and are easily identified, fourth and fifth QTLs have opposite effects, and the effect of third QTL is the weakest.

We run RJ and DDRJ chains $L = 55000$ iterations, discard the first 5000 iterations and take one for every 10 iterations. The chains are initialized with $K = 0$. Convergence is verified using trace plots.

Figures 3.1, 3.2 and 3.3 show the RJ and DDRJ trace plots of K for $\sigma = 0.5$, 1.0 and 1.5, respectively. We observe DDRJ chains show better mixing since they easily move around the models space throughout the chain as a consequence of better proposal candidates. The RJ chain moves with greater difficulty among the possible models and it can get stuck in a specific model for longer periods even if it is a wrong model. When $\sigma = 0.5$, we observe a very poor mixing of the RJ chain since it gets stuck for long periods (in the beginning and end of the chain) in model with $K = 3$ (wrong model). When $\sigma = 1.0$, the RJ chain moves easily around the models space in the beginning of the chain but not in its end.

We also analyze the mixing of the chains by their effective sample size (ESS), (Kass *et al*,

Table 3.1: ESS of K sequences.

	RJ	DDRJ
$\sigma = 0.5$	3	357
$\sigma = 1.0$	159	330
$\sigma = 1.5$	445	894

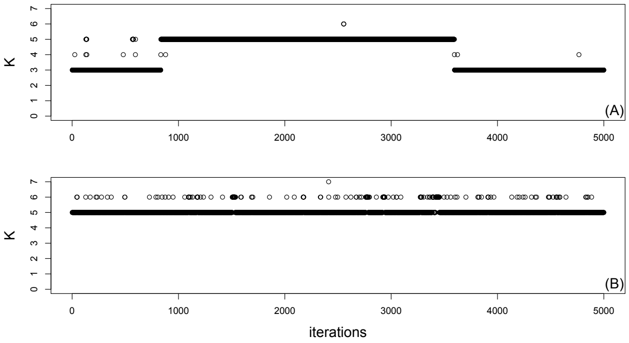


Figure 3.1: Trace plot of K for $\sigma = 0.5$: (A) RJ sequence and (B) DDRJ sequence.

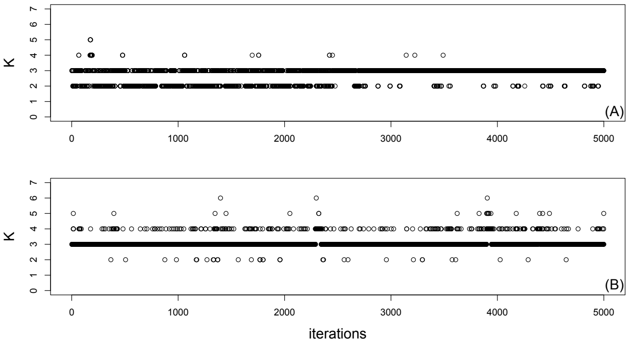


Figure 3.2: Trace plot of K for $\sigma = 1.0$: (A) RJ sequence and (B) DDRJ sequence.

1998) which is the number of effectively independent draws from the *a posteriori* distribution. A large discrepancy between the ESS and the simulation sample size indicates poor mixing. Table 3.1 shows the ESS for the RJ and DDRJ K sequences and we observe the DDRJ ESS is higher than the RJ ESS which confirms a better mixing of DDRJ chains. We observe a very poor mixing of RJ chain mainly for $\sigma = 0.5$. The DDRJ and RJ ESS of the remaining parameters of the models are shown in Table 3.7 and DDRJ ESSs are in most cases higher than RJ ESSs.

Table 3.2 shows *a posteriori* probabilities for K calculated as the relative frequency of each value of K in the sequence. The highest *a posteriori* probability estimate for each situation is in boldface type and the argument that maximizes this probability is the estimate of K. In situations where the genetic effects of QTL are strong compared with the size of the error variability ($\sigma = 0.5$) both methodologies estimate correctly $K = 5$. However, as a result of

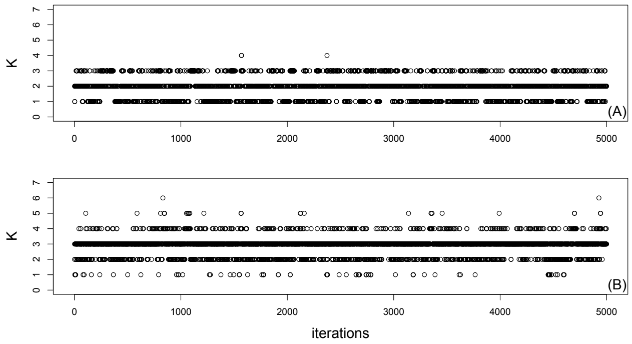


Figure 3.3: Trace plot of K for $\sigma = 1.5$: (A) RJ sequence and (B) DDRJ sequence.

Table 3.2: The *a posteriori* probability for K.

K	σ					
	0.5		1.0		1.5	
	RJ	DDRJ	RJ	DDRJ	RJ	DDRJ
1	0.000	0.000	0.000	0.000	0.308	0.016
2	0.000	0.000	0.268	0.007	**0.490**	0.191
3	0.443	0.000	**0.724**	**0.914**	0.201	**0.706**
4	0.002	0.000	0.007	0.075	0.001	0.081
5	**0.554**	**0.971**	0.001	0.004	0.000	0.005
6	0.001	0.028	0.000	0.000	0.000	< 0.001
7	0.000	0.001	0.000	0.000	0.000	0.000

weak mixing, the RJ chain gets stuck in $K = 3$ for long periods and tends to underestimate the *a posteriori* probability of K. Since $\sigma = 0.5$ represents a small variability of the random error and, consequently, the effect of QTL is more evident, the choice of the correct model should be precise. When $\sigma \geqslant 1.0$, the opposite fourth and fifth QTLs, although they have higher addictive effect than the third QTL, are not identified by RJ and DDRJ since their effect cancel each other. For $\sigma = 1.5$, the RJ procedure estimates only $K = 2$ and shows greater difficulties in locating the QTLs.

Table 3.3 shows the estimates (*a posteriori* average) of parameters and their 95% credibility interval. The estimates of both methodologies are similar when $\sigma = 0.5$ and close to the true values. The DDRJ point estimates of additive and dominance effect of fourth and fifth QTLs are closer to the true simulated parameters than the RJ estimates. Zero belongs to RJ credibility interval of δ_5. The additive and dominance effects of third QTL are the worst estimate in both methods. When $\sigma = 1.0$, RJ and DDRJ estimates for model with $K = 3$ QTLs are similar and the additive and dominance effects estimates of third QTL is also the worst estimate in both methods. For $\sigma = 1.5$, RJ shows a low performance to estimate the number of QTLs and the parameters associated with them. The RJ point estimates are different from the parameters and interval estimates are large.

Table 3.3: The *a posteriori* estimates of the models parameters.

Real value	$\sigma = 0.5$		$\sigma = 1.0$		$\sigma = 1.5$	
	RJ	DDRJ	RJ	DDRJ	RJ	DDRJ
$\lambda_1 = 15.0$	16.3 (16.0;16.8)	13.8 (13.4;14.0)	16.4 (16.0;17.0)	16.5 (13.4;17.9)	45.3 (19.7;80.4)	15.4 (13.0;23.7)
$\lambda_2 = 82.4$	82.6 (81.9;83.2)	83.5 (83.4;83.9)	80.9 (77.4;87.2)	82.8 (81.8;83.4)	178.5 (81.4;302.6)	82.7 (77.7;87.8)
$\lambda_3 = 299.8$	295.4 (294.8;295.8)	299.1 (295.9;302.7)	296.4 (291.2;302.9)	296.0 (292.6;302.4)		294.9 (289.4;303.1)
$\lambda_4 = 363.1$	363.1 (362.2;364.0)	361.6 (361.1;362.1)				
$\lambda_5 = 391.1$	387.8 (386.2;402.6)	389.7 (389.2;390.1.6)				
$\mu = 20.0$	19.92 (19.79;20.05)	19.99 (18.86;20.12)	19.94 (19.70;20.18)	19.93 (19.71;20.16)	20.00 (19.70;20.29)	19.92 (19.65;20.20)
$\alpha_1 = -0.60$	-0.60 (-0.68;-0.52)	-0.59 (-0.67;-0.51)	-0.62 (-0.79;-0.44)	-0.59 (-0.75;-0.41)	0.03 (-0.82;0.98)	-0.59 (-0.80;-0.28)
$\alpha_2 = -0.90$	0.93 (0.84;1.01)	0.91 (0.83;0.99)	0.91 (0.74;1.09)	0.97 (0.80;1.13)	0.78 (0.39;1.14)	0.96 (0.75;1.18)
$\alpha_3 = 0.25$	0.36 (0.27;0.45)	0.37 (0.29;0.46)	0.44 (0.29;0.62)	0.45 (0.28;0.61)		0.57 (0.27;0.78)
$\alpha_4 = -0.40$	-0.37 (-0.49;-0.25)	-0.41 (-0.51;-0.30)				
$\alpha_5 = 0.40$	0.33 (0.21;0.44)	0.38 (0.28;0.48)				
$\delta_1 = 0.30$	0.24 (0.13;0.35)	0.24 (0.13;0.36)	0.14 (-0.08;0.38)	0.13 (-0.10;0.37)	0.07 (-0.32;0.44)	0.10 (-0.19;0.39)
$\delta_2 = 0.05$	0.02 (-0.10;0.13)	0.01 (-0.10;0.12)	-0.02 (-0.32;0.25)	0.03 (-0.20;0.26)	-0.15 (-0.50;0.21)	-0.04 (-0.37;0.26)
$\delta_3 = -0.25$	-0.15 (-0.27;-0.02)	-0.16 (-0.28;-0.03)	-0.05 (-0.31;0.23)	-0.05 (-0.32;0.19)		0.04 (-0.29;0.37)
$\delta_4 = 0.15$	0.17 (0.04;0.31)	0.15 (0.03;0.28)				
$\delta_5 = -0.15$	-0.11 (-0.24;0.03)	-0.15 (-0.27;-0.02)				
σ	0.50 (0.46;0.54)	0.49 (0.45;0.53)	1.02 (0.93;1.10)	0.98 (0.91;1.06)	1.49 (1.38;1.62)	1.44 (1.33;1.57)

We also analyze the simulated data sets using MIM method available in QTL Cartographer. The main model selection criterion available in QTL Cartographer to select the number of QTLs is BIC= $-2\log(L(\widehat{\boldsymbol{\theta}}|\mathbf{y},\mathbf{q})) + pc(n)$, where $\widehat{\boldsymbol{\theta}}$ is the maximum-likelihood estimator of $\boldsymbol{\theta}$, p is the number of free parameters to be estimated and $c(n) = \log(n)$. Other definitions of $c(n)$ are used and available in QTL Cartographer such as $c(n) = 2$ (AIC), $c(n) = 2\log(\log(n))$, $c(n) = 2\log(n)$, $c(n) = 3\log(n)$ and $c(n) = 10X\log(n)$, where we define $X = 0.01$. We choose MIM forward search method to estimate the initial model and test the six model selection criteria to optimize QTLs positions, search for new QTLs and test existing QTLs. We report the results of $c(n) = \log(n)$ which shows the best results for the simulated data sets.

The MIM method combined with BIC model selection methodologies and optimization procedures of QTLs location and effect estimates $K = 6, 3, 3$ for $\sigma = 0.5, 1.0$ and 1.5, respectively. Table 3.4 shows the MIM estimates of the remaining parameters of the models. The method identifies one nonexisting QTL at 9.0 cM when $\sigma = 0.5$ and the additive and dominance effects of the second QTL are biased. We observe that if we sum the estimates of additive and dominance effects of first and second QTLs, we have estimates closer to additive and dominance effect of the QTL located at 15.0 cM, that is, the effects of the true QTL estimated at 14 cM are split with a false QTL identified at 9 cM. When $\sigma = 1.0$ and 1.5, the opposite fourth and fifth QTLs are not identified and the DDRJ estimates of the remaining parameters, especially estimates associated with the third QTL that has weaker effects, are better than MIM estimates. We do not have confidence interval considering the uncertainty of the estimates.

Table 3.4: MIM estimates of the parameters.

Parameter	Real value	$\sigma = 0.5$	$\sigma = 1.0$	$\sigma = 1.5$
$\boldsymbol{\lambda}$	$(15.0, 82.4, 299.8, 363.1, 391.1)$	$(9.0, 14.0, 83.4, 298.8, 363.1, 390.1)$	$(14.0, 83.4, 293.8)$	$(14.0, 83.4, 293.8)$
$\boldsymbol{\alpha}$	$(-0.60, 0.90, 0.25, -0.40, 0.40)$	$(0.24, -0.80, 0.89, 0.40, -0.43, 0.40)$	$(-0.58, 0.96, 0.47)$	$(-0.59, 0.98, 0.61)$
$\boldsymbol{\delta}$	$(0.30, 0.05, -0.25, 0.15, -0.15)$	$(-0.21, 0.42, -0.01, -0.19, 0.18, -0.13)$	$(0.18, 0.01, -0.001)$	$(0.15, -0.02, 0.07)$

Unlike BIC ($c(n) = \log(n)$), we stop AIC, BIC-like criterion with $c(n) = 2\log(\log(n))$ and $c(n) = 0.1\log(n)$ estimation when they wrongly identifies $K = 12, 9$ and 9 significant QTLs for $\sigma = 0.5, 1.0$ and 1.5 located at $\widehat{\boldsymbol{\lambda}} = \{9.0, 14.0, 83.4, 86.4, 91.5, 298.8, 339.8, 351.2, 360.2, 363.1, 388.2, 390.1\}$, $\widehat{\boldsymbol{\lambda}} = \{10.0, 14.0, 83.4, 293.8, 301.8, 309.8, 337.8, 388.1, 390.1\}$ and $\widehat{\boldsymbol{\lambda}} = \{3.0, 9.0, 14.0, 83.4, 86.4, 91.5, 293.8, 338.8, 410.1, 390.1\}$, respectively. BIC-like criterion with $c(n) = 2\log(n)$ estimates $K = 3$ significant QTLs located at $\widehat{\boldsymbol{\lambda}} = \{14.0, 83.4, 293.8\}$ for all values of σ and BIC-like criterion with $c(n) = 3\log(n)$ estimates $K = 3$ QTLs located at $\widehat{\boldsymbol{\lambda}} = \{14.0, 83.4, 296.8\}$ for $\sigma = 0.5$, $K = 2$ QTLs located at $\widehat{\boldsymbol{\lambda}} = \{15.0, 83.4\}$ for $\sigma = 1.0$ and $K = 1$ significant QTL located at $\widehat{\lambda} = 83.5$ for $\sigma = 1.5$. Therefore, we observe MIM method combined with BIC model selection is sensitive to $c(n)$ choice, the method overestimates or underestimates the number of QTLs. If the data were not simulated and we did not known the correct model, we could estimate the model by the six MIM model selection criteria and select the estimated model that was the most frequent between all criteria. In this case, we would choose, for all values of σ, the model estimated by AIC, BIC-like criterion with $c(n) = 2\log(\log(n))$ and $c(n) = 0.1\log(n)$ which is the worst estimated model.

3.4.2 Bone mineral density data set

We apply RJ and DDRJ to the bone mineral density data set (Wergedal *et al*, 2006). It consists of 661 female F2 mice derived from matings of F1 individuals belonging from NZB/B1NJ x RF/J parents. This cross is designed to identify the genetic *loci* regulating femur mechanical properties, geometric properties and bone mineral density (BMD). The data have 94 genetic markers located in 19 chromosomes. NZB, RF and heterozygous markers are coded as 1, −1 and 0, respectively. The data was downloaded from site `http://qtlarchive.org/db/q?pg=projlist`.

Twenty-three phenotypes were measured in all individuals. However, we analyze only the total femur volumetric BMD in milligrams per cubic centimeter. The trait was log-transformed before analysis to be comparable with Wergedal *et al.* (2006) and Cox *et al.* (2009) results. We remove 6 individuals with missing genotype for all markers and use the hidden Markov model (HMM) to input missing genotypes for remaining individuals. For details about inference in HMM see the Chapter 2 of this thesis.

We run $L = 110000$ RJ iterations, discard the first 10000 and take one for every 10 iterations. We run $L = 55000$ DDRJ iterations, discard the first 5000 and take one for every 10 iterations. The sequences are initialized with $K = 0$ and, in DDRJ, we update the birth candidate 10 times before evaluating its acceptance, as proposed by Green & Mira (2001). We analyze the convergence and conclude the number of iterations is sufficient for reliable results.

Table 3.5 shows the *a posteriori* DDRJ probability (relative frequency) for K in each chromosome whose value is a evidence of a QTL presence. The *a posteriori* probability of the model with one QTL is 0.67 in chromosome 7, 0.42 in chromosome 11, 0.38 in chromosome

19, 0.33 in chromosome 9 and 0.25 in chromosome 1, which represents strong evidence of a QTL in chromosome 7 since $K = 1$ is the argument that maximizes the *a posteriori* probability of K and moderate in chromosomes 1, 9, 11 and 19 since, despite the maximum *a posteriori* probability is not for $K = 1$, it is > 0.25. In chromosomes 10, 12, 17 and 18, the probability of a QTL is not negligible. Depending on the cost and researcher interest, these *loci* can be studied in more details. Therefore, we identify at least $K = 5$ QTLs regulating bone mineral density.

Table 3.6 shows estimates and 95% credibility intervals for QTLs' locations (cM) and additive and dominance effects in chromosomes 1, 7, 9, 10, 11, 12, 17, 18 and 19. Additive and dominance effects explain how QTLs genotype are associated to the bone mineral density and their estimates are small (close to zero) because of the scale of the log(BMD). Although the chance of a QTL in chromosomes 10, 17 and 19 is not negligible, zero belongs to their additive and dominance effects 95% credibility interval. Therefore, DDRJ identifies relevant QTLs at chromosomes 1, 7, 9, 11, 12 and 18.

Table 3.5: DDRJ *a posteriori* probability for K in each chromosome.

K	Chromosome									
	1	2	3	4	5	6	7	8	9	10
0	0.60	0.93	0.90	0.86	0.95	0.92	0.30	0.85	0.63	0.76
1	0.25	0.06	0.09	0.11	0.04	0.06	0.67	0.12	0.33	0.17
2	0.13	0.01	0.01	0.03	0.01	0.02	0.03	0.03	0.03	0.06
≥ 3	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01

K	Chromosome									
	11	12	13	14	15	16	17	18	19	
0	0.47	0.79	0.88	0.91	0.92	0.94	0.76	0.82	0.59	
1	0.42	0.18	0.11	0.08	0.07	0.05	0.21	0.16	0.38	
2	0.10	0.03	0.01	0.01	0.01	0.01	0.02	0.02	0.03	
≥ 3	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	

We also analyze this data by a RJ and MIM forward search method combined with BIC model selection ($c(n) = \log(n)$) which shows better results in simulated data sets. We observe only RJ low *a posteriori* probabilities 0.0006, 0.0009 and 0.027 for one QTL in chromosomes 7, 9 and 11, respectively. MIM identifies one QTL in chromosomes 1, 7, 9, 11 and 12 located at 88, 65, 70, 34 and 28 cM, respectively. The MIM point estimates of additive and dominance effects are $\widehat{\boldsymbol{\alpha}} = (0.009, 0.009, 0.012, -0.014, 0.009)$ and $\widehat{\boldsymbol{\delta}} = (0.008, 0.016, -0.005, -0.004, 0.004)$. The MIM effect estimates are close to DDRJ estimates; however, we do not have information about MIM estimates uncertainty. Wergedal *et al.* (2006) use a three-stage strategy and LOD score to identify $K = 5$ QTLs located in chromosomes 3, 7, 10, 11 and 18 at 10, 65, 65, 40 and 50 cM positions, respectively.

Table 3.6: DDRJ estimates and 95% credibility intervals of parameters.

Chromosome	λ	α	δ
1	84.1 (52.8;99.9)	0.008 (0.001;0.013)	0.009 (-0.001;0.002)
7	63.5 (48.3;68.6)	0.009 (0.003;0.014)	0.015 (0.006;0.023)
9	64.4 (45.4;70.8)	0.011 (0.006;0.017)	-0.006 (-0.015;0.005)
10	60.4 (47.0;64.7)	0.003 (-0.003;0.009)	0.003 (-0.006;0.010)
11	32.5 (21.9;43.1)	-0.013(-0.019;-0.008)	-0.002 (-0.011;0.007)
12	30.7 (5.8;57.6)	0.007 (0.001;0.013)	0.001 (-0.012;0.015)
17	35.2 (18.0;54.2)	0.002 (-0.004;0.008)	-0.009 (-0.017;0.001)
18	44.9 (30.7;55.7)	-0.008 (-0.014;-0.003)	0.005 (-0.009;0.015)
19	43.5 (28.5;51.4)	-0.0002 (-0.005;0.005)	-0.005 (-0.014;0.005)

If we use the DDRJ *a posteriori* probability of K as evidence of QTL presence, we observe DDRJ, MIM and Wergedal methodologies identify QTLs in chromosomes 7 and 11; DDRJ and MIM identify more three QTLs in chromosomes 1, 9 and 12; and DDRJ and Werdegal method identifies another QTL in chromosome 18. The Werdegal method also identifies one QTL in chromosome 3 and 10 whose credibility interval of the additive effect and dominance contains zero. Therefore, for this data set, DDRJ methodology identifies QTLs with strong and weak effect in BMD that are not identified by other QTL mapping methods.

3.5 Discussion

We propose a birth-death-merge data-driven reversible jump (DDRJ) for QTL mapping in an F2 population with unknown number of QTLs. We compare the performance of the proposed method with traditional reversible jump (RJ) and multiple-interval mapping (MIM) combined with model selection method and optimization procedures which are the most popular methodologies for QTL mapping in experimental crosses. Although the computational efficiency is an important feature of the methods, we focus in analyzing and comparing their performance in identifying significant QTL regions.

DDRJ shows a better performance to identify and estimate QTLs mainly when their effects are moderate and RJ does not identify them. The better performance of DDRJ occurs because it facilitates the moves around the models space and improves the chain mixing as a consequence of better proposals in transdimensional moves. Unlike DDRJ, the RJ method moves with greater difficulty between the possible models and it can get stuck in a specific model for longer periods even if it is a wrong model. Compared with MIM combined with model selection methods, DDRJ also shows better performance in identifying QTL regions and provides uncertainty information for all the estimates through credibility intervals. For simulated data sets, MIM shows sensitivity to the choice of model selection criterion and, depending on the criterion choice, the method overestimates or underestimates the number of QTLs. As QTLs single effect are not so high in practice, mainly the effect of SNP QTLs (Yang *et al*, 2010), the proposed methodology appears to be useful and brings contributions to identification and characterization of QTLs.

The DDRJ *a posteriori* probability of K is evidence of QTL presence and, even when this value is not maximum for $K > 0$, it allows us to specify regions which can be further explored by genetic researchers. The application in real data set illustrates an example where DDRJ identifies QTLs with strong, moderate and weak effect on the phenotype that are not identified by RJ, MIM or other QTL mapping methods.

The inclusion of merge moves in DDRJ is efficient under analyzed data sets to avoid the split of a true QTL effect with one or more false QTLs. The conventional methodologies usually deal with ghost QTL which appears between two or more QTLs linked in coupling and is generally more significant than the true QTLs. The problem presented here is the opposite of that of ghost QTLs since the true QTLs share their importance with one or more false QTLs. Ghost QTLs are usually avoided by multiple-QTL mapping methods and merge moves included in DDRJ reduced the chance of split QTLs. Since we include the QTL merge move only to avoid split QTLs, we do not include a QTL split step in this procedure.

The amplitude of DDRJ credibility interval of QTLs' location is large when error variability is higher. To improve the DDRJ performance, we can estimate the genotype of a QTL using more than the two flanking markers or use nonconjugate samplers and analyze the results in future works. The proposed data-driven method can be extended to generalized linear models and identifies QTLs that affect binary or discrete phenotypes or for QTL mapping in pedigree data in which the individuals' genotype is correlated if they are relatives and improve SNPs mapping methods which have smaller single effect on the phenotype.

3.6 Appendices

3.6.1 Conditional *a posteriori* distribution of parameters

Combining the likelihood function with the *a priori* distributions, we obtain the conditional *a posteriori* distribution of $\mu|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\mu})$, $\alpha_k|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\alpha_k})$, $\delta_k|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\delta_k})$, for $k = 1, ..., K$, and $\sigma^2|\,(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\sigma^2})$.

Let $a_i = \sum_{k=1}^{K} \alpha_k q_{ik}$ and $d_i = \sum_{k=1}^{K} \delta_k (1 - |q_{ik}|)$, for $i = 1, 2, ..., n$. The conditional *a* *posteriori* distribution of μ is given by

$$\begin{aligned}
\pi\left(\mu|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\mu}\right) &\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu-a_i-d_i)^2-\frac{1}{2\sigma^2_\mu}(\mu-\nu_\mu)^2\right\}\\
&\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(-2y_i\mu+\mu^2+2\mu a_i+2\mu d_i)-\frac{1}{2\sigma^2_\mu}(\mu^2-2\mu\nu_\mu)\right\}\\
&= exp\left\{-\frac{1}{2\sigma^2}\left(n\mu^2-2\mu\sum_{i=1}^{n}(y_i-a_i-d_i)+\frac{\sigma^2\mu^2}{\sigma^2_\mu}-\frac{2\sigma^2\mu\nu_\mu}{\sigma^2_\mu}\right)\right\}\\
&= exp\left\{-\frac{1}{2\sigma^2}\left(\left(\frac{n\sigma^2_\mu+\sigma^2}{\sigma^2_\mu}\right)\mu^2-2\mu\left(\sum_{i=1}^{n}(y_i-a_i-d_i)+\frac{\sigma^2\nu_\mu}{\sigma^2_\mu}\right)\right)\right\}\\
&= exp\left\{-\frac{n\sigma^2_\mu+\sigma^2}{2\sigma^2\sigma^2_\mu}\left(\mu^2-2\mu\frac{\sigma^2_\mu}{n\sigma^2_\mu+\sigma^2}\left(\sum_{i=1}^{n}(y_i-a_i-d_i)+\frac{\sigma^2\nu_\mu}{\sigma^2_\mu}\right)\right)\right\}\\
&= exp\left\{-\frac{1}{2\left(\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}\right)^{-1}}\left(\mu^2-2\mu\left(\frac{\sum_{i=1}^{n}(y_i-a_i-d_i)}{n+\frac{\sigma^2}{\sigma^2_\mu}}+\frac{\nu_\mu}{\frac{n\sigma^2_\mu}{\sigma^2}+1}\right)\right)\right\}\\
&= exp\left\{-\frac{1}{2\left(\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}\right)^{-1}}\left(\mu^2-2\mu\left(\frac{\frac{\sum_{i=1}^{n}(y_i-a_i-d_i)}{\sigma^2}}{\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}}+\frac{\frac{\nu_\mu}{\sigma^2_\mu}}{\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}}\right)\right)\right\},
\end{aligned}\tag{3.15}$$

which is the density function of a Normal$\left(\frac{\frac{\sum_{i=1}^{n}\left(y_i-\sum_{k=1}^{K}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)}{\sigma^2}+\frac{\nu_\mu}{\sigma^2_\mu}}{\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}},\frac{1}{\frac{n}{\sigma^2}+\frac{1}{\sigma^2_\mu}}\right)$ distribution.

The conditional *a posteriori* distribution of α_{k^*}, for $k^* = 1, 2, ...K$, is defined as

$$\begin{aligned}
\pi\left(\alpha_{k^*}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\alpha_{k^*}}\right) &\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i-\mu-\sum_{k=1}^{K}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)^2-\frac{1}{2\sigma_\alpha^2}(\alpha_{k^*}-\nu_\alpha)^2\right\}\\
&\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(-2y_i\alpha_{k^*}q_{ik^*}+2\mu\alpha_{k^*}q_{ik^*}+\left(\sum_{k=1}^{K}\alpha_k q_{ik}\right)^2+2\alpha_{k^*}q_{ik^*}\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)\right\}\\
&\times exp\left\{-\frac{1}{2\sigma_\alpha^2}(\alpha_{k^*}^2-2\alpha_{k^*}\nu_\alpha)\right\}\\
&\propto exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(\alpha_{k^*}^2 q_{ik^*}^2+2\alpha_{k^*}q_{ik^*}\sum_{k\neq k^*}\alpha_k q_{ik}-2\alpha_{k^*}q_{ik^*}\left(y_i-\mu-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)\right)\right\}\\
&\times exp\left\{-\frac{1}{2\sigma_\alpha^2}(\alpha_{k^*}^2-2\alpha_{k^*}\nu_\alpha)\right\}\\
&=exp\left\{-\frac{1}{2\sigma^2}\left(\alpha_{k^*}^2\sum_{i=1}^{n}q_{ik^*}^2-2\alpha_{k^*}\sum_{i=1}^{n}q_{ik^*}\left(y_i-\mu-\sum_{k\neq k^*}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)+\frac{\sigma^2\alpha_{k^*}^2}{\sigma_\alpha^2}-\frac{2\sigma^2\alpha_{k^*}\nu_\alpha}{\sigma_\alpha^2}\right)\right\}\\
&=exp\left\{-\frac{1}{2\sigma^2}\left(\alpha_{k^*}^2\left(\sum_{i=1}^{n}q_{ik^*}^2+\frac{\sigma^2}{\sigma_\alpha^2}\right)-2\alpha_{k^*}\left(\sum_{i=1}^{n}q_{ik^*}\left(y_i-\mu-\sum_{k\neq k^*}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)+\frac{\sigma^2\nu_\alpha}{\sigma_\alpha^2}\right)\right)\right\}\\
&=exp\left\{-\frac{1}{2}\left(\frac{\sum_{i=1}^{n}q_{ik^*}^2}{\sigma^2}+\frac{1}{\sigma_\alpha^2}\right)\left(\alpha_{k^*}^2-2\alpha_{k^*}\frac{\sum_{i=1}^{n}q_{ik^*}\left(y_i-\mu-\sum_{k\neq k^*}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)+\frac{\sigma^2\nu_\alpha}{\sigma_\alpha^2}}{\sum_{i=1}^{n}q_{ik^*}^2+\frac{\sigma^2}{\sigma_\alpha^2}}\right)\right\}\\
&=exp\left\{-\frac{1}{2}\left(\frac{\sum_{i=1}^{n}q_{ik^*}^2}{\sigma^2}+\frac{1}{\sigma_\alpha^2}\right)\left(\alpha_{k^*}^2-2\alpha_{k^*}\frac{\frac{\sum_{i=1}^{n}q_{ik^*}\left(y_i-\mu-\sum_{k\neq k^*}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)}{\sigma^2}+\frac{\nu_\alpha}{\sigma_\alpha^2}}{\frac{\sum_{i=1}^{n}q_{ik^*}^2}{\sigma^2}+\frac{1}{\sigma_\alpha^2}}\right)\right\} \qquad (3.16)
\end{aligned}$$

which is the density function of a Normal $\left(\frac{\frac{\sum_{i=1}^{n}q_{ik^*}\left(y_i-\mu-\sum_{k\neq k^*}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)}{\sigma^2}+\frac{\nu_\alpha}{\sigma_\alpha^2}}{\frac{\sum_{i=1}^{n}q_{ik^*}^2}{\sigma^2}+\frac{1}{\sigma_\alpha^2}}, \frac{1}{\frac{\sum_{i=1}^{n}q_{ik^*}^2}{\sigma^2}+\frac{1}{\sigma_\alpha^2}}\right)$ distribution.

The conditional *a posteriori* distribution of δ_{k^*}, for $k^* = 1, 2, ...K$, is given by

$$
\begin{aligned}
\pi\Big(\delta_{k^*}|\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\delta_{k^*}}\Big) & \propto exp\Big\{-\tfrac{1}{2\sigma^2}\textstyle\sum_{i=1}^{n}\Big(y_i-\mu-\sum_{k=1}^{K}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\Big)^2-\tfrac{1}{2\sigma_\delta^2}(\delta_{k^*}-\nu_\delta)^2\Big\} \\
& \propto exp\Big\{-\tfrac{1}{2\sigma^2}\textstyle\sum_{i=1}^{n}\Big(-2y_i\delta_{k^*}(1-|q_{ik^*}|)+2\mu\delta_{k^*}(1-|q_{ik^*}|)+\Big(\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\Big)^2+2\delta_{k^*}(1-|q_{ik^*}|)\sum_{k=1}^{K}\alpha_k q_{ik}\Big)\Big\} \\
& \times exp\Big\{-\tfrac{1}{2\sigma_\delta^2}(\delta_{k^*}^2-2\delta_{k^*}\nu_\delta)\Big\} \\
& \propto exp\Big\{-\tfrac{1}{2\sigma^2}\textstyle\sum_{i=1}^{n}\Big(\delta_{k^*}^2(1-|q_{ik^*}|)^2+2\delta_{k^*}(1-|q_{ik^*}|)\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)-2\delta_{k^*}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k=1}^{K}\alpha_k q_{ik}\Big)\Big)\Big\} \\
& \times exp\Big\{-\tfrac{1}{2\sigma_\delta^2}(\delta_{k^*}^2-2\delta_{k^*}\nu_\delta)\Big\} \\
& = exp\Big\{-\tfrac{1}{2\sigma^2}\Big(\delta_{k^*}^2\textstyle\sum_{i=1}^{n}(1-|q_{ik^*}|)^2-2\delta_{k^*}\sum_{i=1}^{n}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)-\sum_{k=1}^{K}\alpha_k q_{ik}\Big)\Big)\Big\} \\
& \times exp\Big\{-\tfrac{1}{2\sigma^2}\Big(\frac{\sigma^2\delta_{k^*}^2}{\sigma_\delta^2}-\frac{2\sigma^2\delta_{k^*}\nu_\delta}{\sigma_\delta^2}\Big)\Big\} \\
& = exp\Big\{-\tfrac{1}{2\sigma^2}\Big(\delta_{k^*}^2\Big(\textstyle\sum_{i=1}^{n}(1-|q_{ik^*}|)^2+\frac{\sigma^2}{\sigma_\delta^2}\Big)\Big)\Big\} \\
& \times exp\Big\{-\tfrac{1}{2\sigma^2}\Big(-2\delta_{k^*}\Big(\textstyle\sum_{i=1}^{n}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)-\sum_{k=1}^{K}\alpha_k q_{ik}\Big)\Big)+\frac{\sigma^2\nu_\delta}{\sigma_\delta^2}\Big)\Big\} \\
& = exp\left\{-\tfrac{1}{2}\left(\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2}+\frac{1}{\sigma_\delta^2}\right)\left(\delta_{k^*}^2-2\delta_{k^*}\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)-\sum_{k=1}^{K}\alpha_k q_{ik}\Big)+\frac{\sigma^2\nu_\delta}{\sigma_\delta^2}}{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2+\frac{\sigma^2}{\sigma_\delta^2}}\right)\right\} \\
& = exp\left\{-\tfrac{1}{2}\left(\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2}+\frac{1}{\sigma_\delta^2}\right)\left(\delta_{k^*}^2-2\delta_{k^*}\frac{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)-\sum_{k=1}^{K}\alpha_k q_{ik}\Big)}{\sigma^2}+\frac{\nu_\delta}{\sigma_\delta^2}}{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2}+\frac{1}{\sigma_\delta^2}}\right)\right\},
\end{aligned}
\tag{3.17}
$$

which is the density function of a

$$\text{Normal}\left(\frac{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)\Big(y_i-\mu-\sum_{k=1}^{K}\alpha_k q_{ik}-\sum_{k\neq k^*}\delta_k(1-|q_{ik}|)\Big)}{\sigma^2}+\frac{\nu_\delta}{\sigma_\delta^2}}{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2}+\frac{1}{\sigma_\delta^2}},\frac{1}{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2}+\frac{1}{\sigma_\delta^2}}\right) \text{ distribution.}$$

The conditional *a posteriori* distribution of σ^2 is

$$\sigma^2|\,(\mathbf{y},\mathbf{q},\boldsymbol{\theta}_{-\sigma^2}) \sim \text{Inverse-gamma}\left(\frac{n}{2}+\eta_a, \frac{\sum_{i=1}^{n}\Big(y_i-\sum_{k=1}^{K}\alpha_k q_{ik}-\sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\Big)^2}{2}+\eta_b\right).$$

3.6.2 Additional information about the RJ and DDRJ chains of simulated data sets

Table 3.7 shows the effective sample size (ESS) of the RJ and DDRJ chains for all parameters of the estimated models for simulated data sets. We observe DDRJ ESS is higher than RJ ESS in most cases which confirms a better mixing of DDRJ chains.

Table 3.7: Effective sample size of RJ and DDRJ chains.

Parameter	$\sigma = 0.5$		$\sigma = 1.0$		$\sigma = 1.5$	
	RJ	DDRJ	RJ	DDRJ	RJ	DDRJ
λ_1	2774	4395	2727	1853	216	417
λ_2	2195	4037	192	4569	226	388
λ_3	2186	3	94	61		1762
λ_4	1621	4752				
λ_5	16	4752				
μ	2774	897	155	1252	1605	2399
α_1	2603	4752	272	4569	230	512
δ_1	2774	4752	3621	607	686	1959
α_2	2561	5004	3621	4569	416	1382
δ_2	2774	4752	27	4569	1528	286
α_3	2483	4752	592	2746		369
δ_3	2774	62	262	548		1168
α_4	301	4396				
δ_4	794	898				
α_5	284	5782				
δ_5	217	333				
σ	1591	5166	3807	3459	2266	3364

3.6.3 R codes to carry out DDRJ

In this section, we show DDRJ R codes for a simulated data set.

```
#
###################################
# Functions
###################################
#
################
### sample a multinomial value, where p is the success probability vector
#
rDiscreta<-function(p){
 u<-runif(1)
 P<-cumsum(p)
 val<-sum(P<u)+1
 val}
#
################
### compute the recombination rate through Haldane function for a specific
### genetic distance (dist) in Morgan (M)
```

```
#
Haldane<-function(dist){
 recomb<-0.5*(1-exp(-2*dist))
 recomb}
#
################
### compute the probability of selecting a specific marker using Kruskal-wallis
### and sample a marker
#
prob.selec.marc<-function(dados,res1,qtls,loc.marc){
  # dados = matriz com fenÛtipo na primeira coluna e genÛtipo dos marcadores
  # nas demais colunas
  # qtls = positions of QTLs in current model
  krusk<-numeric()
  for (i in 2:ncol(dados)) krusk[i-1]<-kruskal.test(res1~dados[,i])[[1]]
  if (sum(qtls<=loc.marc[2])>0) krusk[1]<-0
  for (i in 2:(length(loc.marc)-1)) if ((sum(qtls>=loc.marc[i-1] &
qtls<=loc.marc[i])>0)&(sum(qtls>=loc.marc[i] & qtls<=loc.marc[i+1])>0))
krusk[i]<-0
  if (sum(qtls>=loc.marc[length(loc.marc)-1])>0) krusk[length(loc.marc)]<-0
  prob<-krusk/sum(krusk)
  marc<-rDiscreta(prob)
  list(marc,log(prob[marc]),krusk)}
#
################
### compute the probability of excluding a specific marker from the model
### and sample a marker to be excluded
#
prob.excl.marc<-function(num.QTLs,vet.coef){
 efeitos<-numeric()
 for (i in 1:num.QTLs) efeitos[i]<-1/sum(c(abs(vet.coef[(2*i),1]),
abs(vet.coef[((2*i)+1),1])))
 proba<-efeitos/sum(efeitos)
 qtl<-rDiscreta(proba)
 list(qtl,log(proba[qtl]))}
#
################
### sample the position of the new QTL around the chosen marker
```

```
#
pos.qtl<-function(qtls,loc.marc,marc,krusk){
 if (marc==1 | sum(qtls>=loc.marc[marc-1] & qtls<=loc.marc[marc])>0){
  marc.ini<-marc
  marc.fim<-marc+1} else {
  if (marc==length(loc.marc) | sum(qtls>=loc.marc[marc] &
qtls<=loc.marc[marc+1])>0){
   marc.ini<-marc-1
   marc.fim<-marc} else {
   if ((sum(qtls>=loc.marc[marc-1] & qtls<=loc.marc[marc])==0)&
(sum(qtls>=loc.marc[marc]
& qtls<=loc.marc[marc+1])==0)){
    marc.ini<-marc-1
    marc.fim<-marc+1}}}
 marc.pos<-(loc.marc[marc.ini:marc.fim]-loc.marc[marc.ini])/
(loc.marc[marc.fim]-loc.marc[marc.ini])
 estas<-krusk[marc.ini:marc.fim]
 mi.pos<-(t(marc.pos)%*%estas)/sum(estas)
 parA<-mi.pos/(1-mi.pos)
 parB<-1
 uger<-rbeta(1,parA,parB)
 loc.qtl<-loc.marc[marc.ini]+(loc.marc[marc.fim]-loc.marc[marc.ini])*uger
 dens<-dbeta(uger,parA,parB,log = TRUE)
 list(loc.qtl,dens)}
#
dens.pos.qtl<-function(qtls,loc.marc,marc,krusk){
 if (marc==1 | sum(qtls>=loc.marc[marc-1] & qtls<=loc.marc[marc])>0){
  marc.ini<-marc
  marc.fim<-marc+1} else {
  if (marc==length(loc.marc) | sum(qtls>=loc.marc[marc] &
qtls<=loc.marc[marc+1])>0){
   marc.ini<-marc-1
   marc.fim<-marc} else {
   if ((sum(qtls>=loc.marc[marc-1] & qtls<=loc.marc[marc])==0)&
(sum(qtls>=loc.marc[marc] & qtls<=loc.marc[marc+1])==0)){
    marc.ini<-marc-1
    marc.fim<-marc+1}}}
 marc.pos<-(loc.marc[marc.ini:marc.fim]-loc.marc[marc.ini])/
(loc.marc[marc.fim]-loc.marc[marc.ini])
 estas<-krusk[marc.ini:marc.fim]
 mi.pos<-(t(marc.pos)%*%estas)/sum(estas)
 parA<-mi.pos/(1-mi.pos)
 parB<-1
 uger<-(QTLmg-loc.marc[marc.ini])/(loc.marc[marc.fim]-loc.marc[marc.ini])
 dens<-dbeta(uger,parA,parB)
 dens}
#
################
### compute the a priori density of QTLs location
#
priori.loc.qtls<-function(num.qtls,loc.marc){
 loc.qtls<-numeric()
 num.marc<-length(loc.marc)
 prob.loc<-0
 if (num.qtls>0){
 for (i in 1:num.qtls){
   loc.qtls[i]<-runif(1,min=loc.marc[sum(loc.marc<=loc.qtls[i-1])+1],
max=loc.marc[num.marc-(num.qtls-i)])
   prob.loc<-prob.loc+dunif(loc.qtls[i],min=loc.marc[sum(loc.marc<=
loc.qtls[i-1])+1], max=loc.marc[num.marc-(num.qtls-i)],log=TRUE)}}
 list(loc.qtls,prob.loc)}
#
dens.priori.loc.qtls<-function(loc.qtls,loc.marc){
 num.qtls<-length(loc.qtls)
 num.marc<-length(loc.marc)
 prob.loc<-0
 if (num.qtls>0){
 for (i in 1:num.qtls){
   prob.loc<-prob.loc+dunif(loc.qtls[i],min=loc.marc[sum(loc.marc<=
loc.qtls[i-1])+1],max=loc.marc[num.marc-(num.qtls-i)],log=TRUE)}}
 prob.loc}
#
################
### compute the probability of QTL genotype based on the flanking
### markers genotype
#
```

```
# gen = 1 if dominant homozygous
# gen = 0 if heterozygous
# gen = -1 if recessive homozygous
gen.igual<-function(recomb,gen1) if (gen1==0) {(recomb*recomb)+(1-recomb)**2}
else {(1-recomb)**2}
gen.dif1<-function(recomb) 2*(1-recomb)*recomb
gen.dif2<-function(recomb) recomb*recomb
#
calc.prob.gen<-function(gen1,gen2,recomb){
 if (abs(gen1-gen2)==0) {prob<-gen.igual(recomb,gen1)} else {
  if (abs(gen1-gen2)==1) {if (gen1==0) {prob<-gen.dif1(recomb)/2} else
{prob<-gen.dif1(recomb)}}
  else {prob<-gen.dif2(recomb)}}
 prob}
#
################
### sample from the a posteriori distribution of sigma2
#
poster.sigma2<-function(neta.a,neta.b,residuos){
 alpha<-(length(residuos)/2)+neta.a
 beta<-(sum(residuos^2)/2)+neta.b
 sigma2<-1/(rgamma(1,alpha,beta))
 dens<-dgamma((1/sigma2),alpha,beta,log = TRUE)
 list(sigma2,dens)}
#
################
### sample from the a posteriori distribution of mu
#
poster.mi<-function(media.mi,sigma2.mi,sigma2,residuos,mi.anterior){
 res.mi<-residuos+mi.anterior
 quo<-(length(residuos)/sigma2)+(1/sigma2.mi)
 media<-((sum(res.mi)/sigma2)+(media.mi/sigma2.mi))/quo
 variancia<-1/quo
 mi<-rnorm(1,media,sqrt(variancia))
 dens<-dnorm(mi,media,sqrt(variancia),log = TRUE)
 list(mi,dens)}
#
################
```

```
### sample from the a posteriori distribution of additive effect alphaj
#
poster.alpha<-function(media.alpha,sigma2.alpha,sigma2,residuos,
alpha.anterior,gen.QTL){
 res.alpha<-residuos+(alpha.anterior*gen.QTL)
 quo<-(sum(gen.QTL^2)/sigma2)+(1/sigma2.alpha)
 media<-((sum(gen.QTL*res.alpha)/sigma2)+(media.alpha/sigma2.alpha))/quo
 variancia<-1/quo
 alpha<-rnorm(1,media,sqrt(variancia))
 dens<-dnorm(alpha,media,sqrt(variancia),log = TRUE)
 list(alpha,dens)}
#
################
### sample from the a posteriori distribution of dominance effect deltaj
#
poster.delta<-function(media.delta,sigma2.delta,sigma2,residuos,
delta.anterior,gen.QTL.dom){
 res.delta<-residuos+(delta.anterior*gen.QTL.dom)
 quo<-(sum(gen.QTL.dom^2)/sigma2)+(1/sigma2.delta)
 media<-((sum(gen.QTL.dom*res.delta)/sigma2)+(media.delta/sigma2.delta))/quo
 variancia<-1/quo
 delta<-rnorm(1,media,sqrt(variancia))
 dens<-dnorm(delta,media,sqrt(variancia),log = TRUE)
 list(delta,dens)}
#
################
### choose a death or birth move
#
dec.sp.mg<-function(num.QTLs,num.marc){
 if (num.QTLs==0) {psplit<-1; pmerge<-0} else {if (num.QTLs==(num.marc-1))
{psplit<-0;pmerge<-1} else {psplit<-pmerge<-1/2}}
 prob<-c(psplit,pmerge)
 ind.sp.mg<-rDiscreta(prob)
 list(ind.sp.mg,log(prob))}
#
#################
### sample a birth QTL candidate and compute its transition function
#
```

```
gera.inclusao.QTL<-function(dados,residuos,mat.delinea,vet.coef,pos.qtls,
loc.marc,sigma2.vig,alpha.vig,delta.vig,media.mi,sigma2.mi,media.alpha,
sigma2.alpha,media.delta,sigma2.delta,neta.a,neta.b){
 marcador<-prob.selec.marc(dados,residuos,pos.qtls,loc.marc)
 qtl<-pos.qtl(pos.qtls,loc.marc,marcador[[1]],marcador[[3]])
 #
 dQTL<-qtl[[1]]
 Marc1<-sum(loc.marc<=dQTL)
 Marc2<-length(loc.marc)-(sum(loc.marc>=dQTL)-1)
 dM1<-loc.marc[Marc1]
 dM2<-loc.marc[Marc2]
 r12<-Haldane(abs(dM2-dM1))
 r1<-Haldane(abs(dQTL-dM1))
 r2<-Haldane(abs(dQTL-dM2))
 #
 matriz.prob.gen.QTL<-numeric()
 probQTL<-numeric()
 for (j in 1:nrow(dados)){
  gen<-c(-1,0,1)
  for (i in 1:3) probQTL[i]<-(calc.prob.gen(dados[j,(Marc1+1)],gen[i],r1)*
calc.prob.gen(gen[i],dados[j,(Marc2+1)],r2))/calc.prob.gen(
dados[j,(Marc1+1)],dados[j,(Marc2+1)],r12)
 matriz.prob.gen.QTL<-rbind(matriz.prob.gen.QTL,probQTL)}
 #
 dados.QTL<-matrix(0,nrow(dados),1)
 for (i in 1:nrow(dados)) dados.QTL[i,1]<-gen[rDiscreta(
matriz.prob.gen.QTL[i,])]
 mat.delinea<-cbind(mat.delinea,dados.QTL)
 #
 alpha<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,residuos,alpha.vig,
dados.QTL[,1])
 vet.coef<-rbind(vet.coef,alpha[[1]])
 predito<-mat.delinea%*%vet.coef
 residuos<-dados[,1]-predito
 #
 dados.QTL<-cbind(dados.QTL,(1-abs(dados.QTL[,1])))
 delta<-poster.delta(media.delta,sigma2.delta,sigma2.vig,residuos,delta.vig,
dados.QTL[,2])
```

```
 mat.delinea<-cbind(mat.delinea,dados.QTL[,2])
 vet.coef<-rbind(vet.coef,delta[[1]])
 predito<-mat.delinea%*%vet.coef
 residuos<-dados[,1]-predito
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef[1,1])
 vet.coef[1,1]<-mi[[1]]
 predito<-mat.delinea%*%vet.coef
 residuos<-dados[,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 #
 list(qtl[[1]],mat.delinea,vet.coef,sigma2[[1]],marcador[[2]],qtl[[2]],
matriz.prob.gen.QTL,alpha[[2]],delta[[2]],mi[[2]],sigma2[[2]],
c(pos.qtls,qtl[[1]]))}
#
#################
### sample a death QTL candidate and compute its transition function
#
gera.exclusao.QTL<-function(pos.qtls,vet.coef,mat.delinea,sigma2.vig,
media.mi,sigma2.mi,neta.a,neta.b){
 num.QTLs<-length(pos.qtls)
 qtl<-prob.excl.marc(num.QTLs,vet.coef)
 #
 pos.qtls<-pos.qtls[-qtl[[1]]]
 mat.delinea<-matrix(mat.delinea[,-c((2*qtl[[1]]),(2*qtl[[1]]+1))],
nrow=nrow(dados))
 vet.coef<-matrix(vet.coef[-c((2*qtl[[1]]),(2*qtl[[1]]+1))],ncol=1)
 predito<-mat.delinea%*%vet.coef
 residuos<-dados[,1]-predito
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef[1,1])
 vet.coef[1,1]<-mi[[1]]
 predito<-mat.delinea%*%vet.coef
 residuos<-dados[,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 #
 list(qtl[[1]],mat.delinea,vet.coef,sigma2[[1]],qtl[[2]],mi[[2]],sigma2[[2]],
sigma2[[2]],sigma2[[2]],sigma2[[2]],sigma2[[2]],pos.qtls)}
```

```
#
#################
### sample a merge QTL candidate and compute its transition function
#
gera.juncao.QTL<-function(dados,num.QTLs,mat.delinea,vet.coef,pos.qtls,
media.alpha,sigma2.alpha,sigma2.vig,media.delta,sigma2.delta,media.mi,
sigma2.mi,neta.a,neta.b){
 cramer<-numeric(num.QTLs-1)
 for (i in 1:(num.QTLs-1)) cramer[i]<-abs(cv.test(mat.delinea[,2*i],
mat.delinea[,2*(i+1)]))
 prob_par<-cramer/sum(cramer)
 junta<-rDiscreta(prob_par)
 par_QTL<-c(junta,junta+1)
 efeitos<-c(1/sum(abs(vet.coef[2*junta,1]),abs(vet.coef[2*junta+1,1])),
1/sum(abs(vet.coef[2*(junta+1),1]),abs(vet.coef[2*(junta+1)+1,1])))
 prob<-efeitos/sum(efeitos)
 gera<-rDiscreta(prob) # escolhe o QTL que vai sumir
 qtl<-par_QTL[gera]
 #
 pos.qtls.c<-pos.qtls[-qtl]
 mat.delinea.c<-matrix(mat.delinea[,-c((2*qtl),(2*qtl+1))],nrow=row(dados))
 vet.coef.c<-matrix(vet.coef[-c((2*qtl),(2*qtl+1))],ncol=1)
 predito<-mat.delinea.c%*%vet.coef.c
 residuos<-dados[,1]-predito
 #
 alpha<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,residuos,
vet.coef.c[(2*par_QTL[[1]]),1],mat.delinea.c[,(2*par_QTL[[1]])])
 vet.coef.c[(2*par_QTL[[1]]),1]<-alpha[[1]]
 residuos<-dados[,1]-(mat.delinea.c%*%vet.coef.c)
 delta<-poster.delta(media.delta,sigma2.delta,sigma2.vig,residuos,
vet.coef.c[(2*par_QTL[[1]])+1,1],mat.delinea.c[,(2*par_QTL[[1]])+1])
 vet.coef.c[(2*par_QTL[[1]])+1,1]<-delta[[1]]
 residuos<-dados[,1]-(mat.delinea.c%*%vet.coef.c)
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef.c[1,1])
 vet.coef.c[1,1]<-mi[[1]]
 predito<-mat.delinea.c%*%vet.coef.c
 residuos<-dados[,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 prob_merge<-log(prob[gera])+log(prob_par[junta])
 #
 list(qtl,mat.delinea.c,vet.coef.c,sigma2[[1]],prob_merge,mi[[2]],sigma2[[2]],
alpha[[2]],delta[[2]],par_QTL[which(par_QTL!=qtl)],par_QTL,pos.qtls.c)}
#
#################
### compute the acceptance probability of a birth move
#
prob.aceitacao<-function(residuossp,residuos,sigma2,sigma2sp,misp,mi,
media.mi,sigma2.mi,neta.a,neta.b,alphasp,media.alpha,sigma2.alpha,
deltasp,media.delta,sigma2.delta,loc.marc,num.QTLs,num.QTLssp,psplit,
pmerge,pmarc,plambdasp,plambdamg,postmi,postsigma2,postalphasp,
postdeltasp,postmisp,postsigma2sp,pri.pos.sp,pri.pos){
 vero<-sum(dnorm(residuossp,0,sqrt(sigma2sp),log=TRUE))-
sum(dnorm(residuos,0,sqrt(sigma2),log=TRUE))
 priori<-dnorm(misp,media.mi,sqrt(sigma2.mi),log=TRUE)-
dnorm(mi,media.mi,sqrt(sigma2.mi),log=TRUE)+
         dgamma((1/sigma2sp),neta.a,neta.b,log=TRUE)-
dgamma((1/sigma2),neta.a,neta.b,log=TRUE)+
         dnorm(alphasp,media.alpha,sqrt(sigma2.alpha),log=TRUE)+
dnorm(deltasp,media.delta,sqrt(sigma2.delta),log=TRUE)+
         pri.pos.sp-pri.pos
 trans<-pmerge+plambdamg+postmi+postsigma2-psplit-pmarc-plambdasp-
postalphasp-postdeltasp-postmisp-postsigma2sp
 prob.ace<-exp(vero+priori+trans)
 prob.ace}
#
#################
### compute the acceptance probability of a split move
#
prob.aceitacao.split<-function(residuossp,residuos,sigma2,sigma2sp,
misp,mi,media.mi,sigma2.mi,neta.a,neta.b,alphamg,alphasp1,alphasp2,
media.alpha,sigma2.alpha,deltamg,deltasp1,deltasp2,media.delta,
sigma2.delta,loc.marc,num.QTLs,num.QTLssp,psplit,pmerge,pmarc,
plambdasp,plambdamg,postalpha,postdelta,postmi,postsigma2,postalphasp1,
postalphasp2,postdeltasp1,postdeltasp2,postmisp,postsigma2sp,pri.pos.sp,
pri.pos){
 vero<-sum(dnorm(residuossp,0,sqrt(sigma2sp),log=TRUE))-
sum(dnorm(residuos,0,sqrt(sigma2),log=TRUE))
 priori<-dnorm(misp,media.mi,sqrt(sigma2.mi),log=TRUE)-
dnorm(mi,media.mi,sqrt(sigma2.mi),log=TRUE)+
         dgamma((1/sigma2sp),neta.a,neta.b,log=TRUE)-
dgamma((1/sigma2),neta.a,neta.b,log=TRUE)+
         dnorm(alphasp1,media.alpha,sqrt(sigma2.alpha),log=TRUE)+
dnorm(deltasp1,media.delta,sqrt(sigma2.delta),log=TRUE)+
         dnorm(alphasp2,media.alpha,sqrt(sigma2.alpha),log=TRUE)+
dnorm(deltasp2,media.delta,sqrt(sigma2.delta),log=TRUE)-
         (dnorm(alphamg,media.alpha,sqrt(sigma2.alpha),log=TRUE)+
 dnorm(deltamg,media.delta,sqrt(sigma2.delta),log=TRUE))+
         pri.pos.sp-pri.pos
 trans<-pmerge+plambdamg+postalpha+postdelta+postmi+postsigma2-psplit-pmarc-
plambdasp-postalphasp1-postalphasp2-postdeltasp1-postdeltasp2-postmisp-
postsigma2sp
 prob.ace<-exp(vero+priori+trans)
 prob.ace}
#
##################
# run DDRJ procedure and initialize the chain with k=0 QTLs
##################
#
# hyperparameters of a priori distributions
#
neta.a<-0.1
neta.b<-0.1
sigma2.mi<-100
media.mi<-0
sigma2.alpha<-100
media.alpha<-0
sigma2.delta<-100
media.delta<-0
#
####### Model initialization
#
num.marc<-ncol(dados)-1
residuos<-dados[,1]
```

```
sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
#
mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,0)[[1]]
#
pos.qtls<-numeric() # QTLs positons in initial model.
num.QTLs<-length(pos.qtls)
mat.delinea<-matrix(1,nrow(dados),1)
vet.coef<-matrix(mi,1,1)
predito<-mat.delinea%*%vet.coef
residuos<-dados[,1]-predito
sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
#
indSpMgtotal<-numeric() # 1 - birth candidate; 2 - death candidate
probacetotal<-numeric()
probacemerge<-numeric()
num.QTL.total<-numeric()
indrejtotal<-numeric()
#
#####################
#  run DDRJ
#####################
#
set.seed(493)
amostrasfin<-5000
burnin<-10000
saltos<-10 # jumps
AmostrasTotal<-burnin+amostrasfin*saltos
#
library(compiler)
enableJIT(3)
#
for (int in (1:AmostrasTotal)){
  cat('\n', int)
  cand.sp.mg<-dec.sp.mg(num.QTLs,num.marc)
  indSpMgtotal[int]<-cand.sp.mg[[1]]
  #
  ##############
  ###### QTL birth
  ##############
  #
  if (indSpMgtotal[int]==1) {candidato<-gera.inclusao.QTL(dados,residuos,
mat.delinea,vet.coef,pos.qtls,loc.marc,sigma2.vig,0,0,media.mi,sigma2.mi,
media.alpha,sigma2.alpha,media.delta,sigma2.delta,neta.a,neta.b)
   num.QTL.total[int]<-num.QTLs+1
   residuossp<-dados[,1]-(candidato[[2]]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2sp<-candidato[[4]]
   mi<-vet.coef[1,1]
   misp<-candidato[[3]][1,1]
   alphasp<-candidato[[3]][(nrow(candidato[[3]])-1),1]
   deltasp<-candidato[[3]][(nrow(candidato[[3]])),1]
   num.QTLssp<-num.QTLs+1
   psplit<-cand.sp.mg[[2]][1]
   pmerge<-dec.sp.mg(num.QTLssp,num.marc)[[2]][2]
   pmarc<-candidato[[5]]
   plambdasp<-candidato[[6]]
  #
   efeito<-numeric()
   if (num.QTLs==0) efeito<-0
   if (num.QTLs>0) for (i in 1:num.QTLs) efeito[i]<-1/(abs(vet.coef[2*i,1])+
 abs(vet.coef[(2*i)+1,1]))
   efeitosp<-1/(abs(alphasp)+abs(deltasp))
   plambdamg<-log(efeitosp/(efeitosp+sum(efeito)))
  #
   res.mi<-residuos+mi
   quo<-(length(residuos)/sigma2sp)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2sp)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
  #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
  #
   postalphasp<-candidato[[8]]
   postdeltasp<-candidato[[9]]
   postmisp<-candidato[[10]]
   postsigma2sp<-candidato[[11]]
   pos.qtls.sp<-sort(candidato[[12]])
   pri.pos.sp<-dens.priori.loc.qtls(pos.qtls.sp,loc.marc)
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
  #
   probace<-prob.aceitacao(residuossp,residuos,sigma2,sigma2sp,misp,mi,
media.mi,sigma2.mi,neta.a,neta.b,alphasp,media.alpha,sigma2.alpha,
deltasp,media.delta,sigma2.delta,loc.marc,num.QTLs,num.QTLssp,psplit,
pmerge,pmarc,plambdasp,plambdamg,postmi,postsigma2,postalphasp,
postdeltasp,postmisp,postsigma2sp,pri.pos.sp,pri.pos)}
  #
  ##############
  ###### QTL death
  ##############
  #
  if (indSpMgtotal[int]==2) {candidato<-gera.exclusao.QTL(pos.qtls,vet.coef,
mat.delinea,sigma2.vig,media.mi,sigma2.mi,neta.a,neta.b)
   num.QTL.total[int]<-num.QTLs-1
   residuosmg<-dados[,1]-(candidato[[2]]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2mg<-candidato[[4]]
   mi<-vet.coef[1,1]
   mimg<-candidato[[3]][1,1]
   num.QTLsmg<-num.QTLs-1
   pmerge<-cand.sp.mg[[2]][2]
   psplit<-dec.sp.mg(num.QTLsmg,num.marc)[[2]][1]
   postmimg<-candidato[[6]]
   postsigma2mg<-candidato[[7]]
   pos.qtls.mg<-sort(candidato[[12]])
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
   pri.pos.mg<-dens.priori.loc.qtls(pos.qtls.mg,loc.marc)
   plambdamg<-candidato[[5]]
   alpha<-vet.coef[(2*candidato[[1]]),1]
   delta<-vet.coef[(2*candidato[[1]]+1),1]
  #
   krusk<-prob.selec.marc(dados,residuosmg,pos.qtls.mg,loc.marc)[[3]]
   prob<-krusk/sum(krusk)
   QTLmg<-pos.qtls[candidato[[1]]]
   Marc1<-sum(loc.marc<=QTLmg)
   Marc2<-num.marc-sum(loc.marc>=QTLmg)+1
   loc.Marc1<-dens.pos.qtl(pos.qtls.mg,loc.marc,Marc1,krusk)
   loc.Marc2<-dens.pos.qtl(pos.qtls.mg,loc.marc,Marc2,krusk)
   plambdasp<-log(prob[Marc1]*loc.Marc1+prob[Marc2]*loc.Marc2)
   pmarc<-0
  #
   quo<-(sum(mat.delinea[,2*candidato[[1]]]^2)/sigma2mg)+(1/sigma2.alpha)
   media<-((sum(mat.delinea[,2*candidato[[1]]]*residuosmg)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha<-dnorm(alpha,media,sqrt(variancia),log = TRUE)
  #
   quo<-(sum(mat.delinea[,2*candidato[[1]]+1]^2)/sigma2mg)+(1/sigma2.delta)
   media<-((sum(mat.delinea[,2*candidato[[1]]+1]*residuosmg)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta<-dnorm(delta,media,sqrt(variancia),log = TRUE)
  #
   res.mi<-residuos+vet.coef[1,1]
   quo<-(length(residuos)/sigma2mg)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2mg)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
  #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
  #
   probace<-1/prob.aceitacao(residuos,residuosmg,sigma2mg,sigma2,mi,mimg,
media.mi,sigma2.mi,neta.a,neta.b,alpha,media.alpha,sigma2.alpha,delta,
media.delta,sigma2.delta,loc.marc,num.QTLsmg,num.QTLs,psplit,pmerge,
pmarc,plambdasp,plambdamg,postmimg,postsigma2mg,postalpha,postdelta,
postmi,postsigma2,pri.pos,pri.pos.mg)}
  #
  #####################
  #  update the model parameters
  #####################
  #
  # update the number of QTLs (accept or reject the birth or death candidate)
  #
  probacetotal[int]<-probace
  aux2<-runif(1)
  if (aux2<probace){
   indrejtotal[int]<-0
   pos.qtls<-candidato[[12]]
   num.QTLs<-length(pos.qtls)
   mat.delinea<-candidato[[2]]
   vet.coef<-candidato[[3]]
   sigma2.vig<-candidato[[4]]
   if (num.QTLs>0){
    posicao<-order(pos.qtls)
    pos.qtls<-pos.qtls[posicao]
    posicao2<-1
    for (i in 1:length(posicao)) posicao2<-c(posicao2,posicao[i]*2,posicao[i]*2+1)
    mat.delinea<-mat.delinea[,posicao2]
    vet.coef<-matrix(vet.coef[posicao2,],ncol(mat.delinea),1)}
   residuos<-dados[,1]-(mat.delinea%*%vet.coef)}
  #
  if (aux2>=probace) indrejtotal[int]<-1
  #
  #### evaluate a merge move of two consecutive QTLs
  #
  if (num.QTLs>1){
   candidato<-gera.juncao.QTL(dados,num.QTLs,mat.delinea,vet.coef,pos.qtls,
media.alpha,sigma2.alpha,sigma2.vig,media.delta,sigma2.delta,media.mi,
sigma2.mi,neta.a,neta.b)
   residuosmg<-dados[,1]-(candidato[[2]]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2mg<-candidato[[4]]
   mi<-vet.coef[1,1]
   mimg<-candidato[[3]][1,1]
   num.QTLsmg<-num.QTLs-1
   pmerge<-0
   psplit<-0
   postmimg<-candidato[[6]]
   postsigma2mg<-candidato[[7]]
   postalphamg<-candidato[[8]]
   postdeltamg<-candidato[[9]]
   pos.qtls.mg<-sort(candidato[[12]])
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
   pri.pos.mg<-dens.priori.loc.qtls(pos.qtls.mg,loc.marc)
   plambdamg<-candidato[[5]]
   alphamg<-candidato[[3]][(2*candidato[[11]][1]),1]
   deltamg<-candidato[[3]][(2*candidato[[11]][1]+1),1]
   alphasp1<-vet.coef[(2*candidato[[1]]),1]
   deltasp1<-vet.coef[(2*candidato[[1]]+1),1]
   alphasp2<-vet.coef[(2*candidato[[10]]),1]
   deltasp2<-vet.coef[(2*candidato[[10]]+1),1]
   #
   krusk<-prob.selec.marc(dados,residuosmg,pos.qtls.mg,loc.marc)[[3]]
   prob<-krusk/sum(krusk)
   QTLmg<-pos.qtls[candidato[[1]]]
   Marc1<-sum(loc.marc<=QTLmg)
   Marc2<-num.marc-sum(loc.marc>=QTLmg)+1
   loc.Marc1<-dens.pos.qtl(pos.qtls.mg,loc.marc,Marc1,krusk)
   loc.Marc2<-dens.pos.qtl(pos.qtls.mg,loc.marc,Marc2,krusk)
   plambdasp<-log(prob[Marc1]*loc.Marc1+prob[Marc2]*loc.Marc2)
   pmarc<-0
   #
   quo<-(sum(mat.delinea[,2*candidato[[1]]]^2)/sigma2mg)+(1/sigma2.alpha)
   media<-((sum(mat.delinea[,2*candidato[[1]]]*residuosmg)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha1<-dnorm(alphasp1,media,sqrt(variancia),log = TRUE)
   #
   quo<-(sum(mat.delinea[,2*candidato[[1]]+1]^2)/sigma2mg)+(1/sigma2.delta)
   media<-((sum(mat.delinea[,2*candidato[[1]]+1]*residuosmg)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta1<-dnorm(deltasp1,media,sqrt(variancia),log = TRUE)
   #
   vet_coef_parc<-rbind(candidato[[3]],vet.coef[candidato[[1]]*2,1],
vet.coef[candidato[[1]]*2+1,1])
   mat_delinea_parc<-cbind(candidato[[2]],mat.delinea[,candidato[[1]]*2],
mat.delinea[,candidato[[1]]*2+1])
   vet_coef_parc<-matrix(vet_coef_parc[-(2*candidato[[11]][1]),],ncol=1)
   mat_delinea_parc<-mat_delinea_parc[,-(2*candidato[[11]][1])]
   residuosparc<-dados[,1]-mat_delinea_parc%*%vet_coef_parc
   quo<-(sum(mat.delinea[,2*candidato[[10]]]^2)/sigma2mg)+(1/sigma2.alpha)
   media<-((sum(mat.delinea[,2*candidato[[10]]]*residuosparc)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha2<-dnorm(alphasp2,media,sqrt(variancia),log = TRUE)
   #
   vet_coef_parc<-matrix(vet.coef[-(2*candidato[[10]]+1)],ncol=1)
   mat_delinea_parc<-mat.delinea[,-(2*candidato[[10]]+1)]
   residuosparc<-dados[,1]-mat_delinea_parc%*%vet_coef_parc
   quo<-(sum(mat.delinea[,2*candidato[[10]]+1]^2)/sigma2mg)+(1/sigma2.delta)
   media<-((sum(mat.delinea[,2*candidato[[10]]+1]*residuosparc)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta2<-dnorm(deltasp2,media,sqrt(variancia),log = TRUE)
   #
   res.mi<-residuos+vet.coef[1,1]
   quo<-(length(residuos)/sigma2mg)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2mg)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
   #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
   #
   probace<-1/prob.aceitacao.split(residuos,residuosmg,sigma2mg,sigma2,mi,mimg,
media.mi,sigma2.mi,neta.a,neta.b,alphamg,alphasp1,alphasp2,media.alpha,
sigma2.alpha,deltamg,deltasp1,deltasp2,media.delta,sigma2.delta,loc.marc,
num.QTLsmg,num.QTLs,psplit,pmerge,pmarc,plambdasp,plambdamg,postalphamg,
postdeltamg,postmimg,postsigma2mg,postalpha1,postalpha2,postdelta1,postdelta2,
postmi,postsigma2,pri.pos,pri.pos.mg)
   #
   probacemerge[int]<-probace
   aux2<-runif(1)
   if (aux2<probace){
    pos.qtls<-candidato[[12]]
    num.QTLs<-length(pos.qtls)
    mat.delinea<-candidato[[2]]
    vet.coef<-candidato[[3]]
    sigma2.vig<-candidato[[4]]
    residuos<-dados[,1]-(mat.delinea%*%vet.coef)}}
  #
  #### update QTLs position - Metropolis Hastings
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    M.esq<-sum(loc.marc<=pos.qtls[i])
    M.dir<-num.marc-(sum(loc.marc>=pos.qtls[i]))+1
    loc.cand<-runif(1,min=loc.marc[M.esq],max=loc.marc[M.dir])
  #
    vet.delinea<-mat.delinea
    r1c<-Haldane(abs(loc.cand-loc.marc[M.esq]))
    r2c<-Haldane(abs(loc.cand-loc.marc[M.dir]))
    r12<-Haldane(abs(loc.marc[M.dir]-loc.marc[M.esq]))
    r1<-Haldane(abs(pos.qtls[i]-loc.marc[M.esq]))
    r2<-Haldane(abs(pos.qtls[i]-loc.marc[M.dir]))
    probQTL<-numeric()
    probQTLat<-numeric()
    logdens<-numeric()
    logdensat<-numeric()
    densacumat<-0
    probacumat<-0
    probacum<-0
    densacum<-0
    gen<-c(-1,0,1)
    for (j in 1:nrow(dados)){
     for (l in 1:3){
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1c)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2c))/
calc.prob.gen(dados[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
```

```
      probQTLat[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/
calc.prob.gen(dados[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      dom<-1-abs(gen[l])
      vet.delinea[j,2*i]<-gen[l]
      vet.delinea[j,(2*i)+1]<-dom
      logdens[l]<-dnorm(dados[j,1],vet.delinea[j,]%*%
vet.coef,sqrt(sigma2.vig),log=TRUE)
      logdensat[l]<-dnorm(dados[j,1],vet.delinea[j,]%*%
vet.coef,sqrt(sigma2.vig),log=TRUE)}
     prob<-exp(probQTL+logdens-max(probQTL+logdens))/sum(exp(probQTL+
logdens-max(probQTL+logdens)))
     probat<-exp(probQTLat+logdensat-max(probQTLat+logdensat))/
sum(exp(probQTLat+logdensat-max(probQTLat+logdensat)))
     ger.gen<-rDiscreta(prob)
     vet.delinea[j,2*i]<-gen[ger.gen]
     vet.delinea[j,(2*i)+1]<-1-abs(vet.delinea[j,2*i])
     probacum<-probacum+log(prob[ger.gen])
     densacum<-densacum+logdens[ger.gen]
     gen.at<-sum(gen<=mat.delinea[j,2*i])
     probacumat<-probacumat+log(probat[gen.at])
     densacumat<-densacumat+logdensat[gen.at]}
  #
    numer<-denom<-0
    for (j in 1:nrow(dados)){
     numer<-numer+log((calc.prob.gen(dados[j,(M.esq+1)],vet.delinea[j,2*i],r1c)*
calc.prob.gen(vet.delinea[j,2*i],dados[j,(M.dir+1)],r2c))/calc.prob.gen
(dados[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
     denom<-denom+log((calc.prob.gen(dados[j,(M.esq+1)],mat.delinea[j,2*i],r1)*
calc.prob.gen(mat.delinea[j,2*i],dados[j,(M.dir+1)],r2))/calc.prob.gen
(dados[j,(M.esq+1)],dados[j,(M.dir+1)],r12))}
  #
    paceit<-exp(densacum+numer+probacumat-densacumat-denom-probacum)
    aux3<-runif(1)
    if (aux3<paceit){
     pos.qtls[i]<-loc.cand
     mat.delinea[,2*i]<-vet.delinea[,2*i]
     mat.delinea[,(2*i)+1]<-vet.delinea[,(2*i)+1]}}}
  #
  #### update QTLs genotype - Gibbs sampling
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    M.esq<-sum(loc.marc<=pos.qtls[i])
    M.dir<-num.marc-(sum(loc.marc>=pos.qtls[i]))+1
    r12<-Haldane(abs(loc.marc[M.dir]-loc.marc[M.esq]))
    r1<-Haldane(abs(pos.qtls[i]-loc.marc[M.esq]))
    r2<-Haldane(abs(pos.qtls[i]-loc.marc[M.dir]))
    probQTL<-numeric()
    logdens<-numeric()
    gen<-c(-1,0,1)
    for (j in 1:nrow(dados)){
     for (l in 1:3){
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/
calc.prob.gen(dados[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      dom<-1-abs(gen[l])
      vet.delinea<-mat.delinea[j,]
      vet.delinea[2*i]<-gen[l]
      vet.delinea[(2*i)+1]<-dom
      logdens[l]<-dnorm(dados[j,1],vet.delinea%*%
vet.coef,sqrt(sigma2.vig),log=TRUE)}
     prob<-exp(probQTL+logdens-max(probQTL+logdens))/
sum(exp(probQTL+logdens-max(probQTL+logdens)))
     mat.delinea[j,2*i]<-gen[rDiscreta(prob)]
     mat.delinea[j,(2*i)+1]<-1-abs(mat.delinea[j,2*i])}}}
  residuos<-dados[,1]-(mat.delinea%*%vet.coef)
  #
  #### update mu - Gibbs sampling
  #
  vet.coef[1,1]<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef[1,1])[[1]]
  residuos<-dados[,1]-(mat.delinea%*%vet.coef)
  #
  #### update additive and dominance effect - Gibbs sampling
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    vet.coef[(2*i),1]<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,
residuos,vet.coef[(2*i),1],mat.delinea[,(2*i)])[[1]]
    residuos<-dados[,1]-(mat.delinea%*%vet.coef)
    vet.coef[(2*i)+1,1]<-poster.delta(media.delta,sigma2.delta,sigma2.vig,
residuos,vet.coef[(2*i)+1,1],mat.delinea[,(2*i)+1])[[1]]
    residuos<-dados[,1]-(mat.delinea%*%vet.coef)}}
  #
  #### update error variance - sigma2
  #
  sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
  #
  ############################## save the sample after burn-in and jumps
  #
  if (int>burnin & int%%saltos==0){
   cat('',num.QTLs,file="/home/milan/¡rea de Trabalho/Documentos/Daiane/
numero_QTLs_mg_var1_alfa2_chain1.txt",append=T)
   cat('',pos.qtls,file="/home/milan/¡rea de Trabalho/Documentos/Daiane/
posicao_QTLs_mg_var1_alfa2_chain1.txt",append=T)
   cat('',vet.coef,file="/home/milan/¡rea de Trabalho/Documentos/Daiane/
vetor_coeficientes_mg_var1_alfa2_chain1.txt",append=T)
   cat('',sigma2.vig,file="/home/milan/¡rea de Trabalho/Documentos/Daiane/
sigma2_mg_var1_alfa2_chain1.txt",append=T)}
}
   cat('',indrejtotal,file="/home/milan/¡rea de Trabalho/Documentos/Daiane/
indrej_mg_var1_alfa2_chain1.txt",append=T)
   cat('',round(probacetotal,2),file="/home/milan/¡rea de Trabalho/Documentos/
Daiane/prob_rej_sp_var1_alfa2_chain1.txt",append=T)
   cat('',round(probacemerge,2),file="/home/milan/¡rea de Trabalho/Documentos/
Daiane/prob_rej_mg_var1_alfa2_chain1.txt",append=T)
#
```

Chapter

4

QTL mapping model checking

In this chapter, we apply some model validation statistics described in Lesaffre & Lawson (2012) to check the fit of a QTL mapping model to data.

4.1 Introduction

In Chapter 3 we propose a Bayesian method to select and estimate the best QTL mapping model for a data set among a set of candidate models. However, it does not guarantee a good fit of the selected model to the data. Evaluation of the goodness of fit of a model requires a lot of exploration such as 1 - checking that inference from the chosen model is reasonable, 2 - verifying that the model can reproduce the data, 3 - sensitivity analyses by varying certain aspects of the model.

Bayesian models can be evaluated in several ways. Most simply, the fit of model to data can be assessed using *a posteriori* predictive statistics (Gelman *et al*, 1996, 2014; Rubin *et al.* , 1984) and exploring the *a posteriori* distribution of the residuals. Lesaffre & Lawson (2012) shows many statistics to evaluate the fit of general Bayesian models and detect outliers and influential observations.

Although QTL mapping model is a linear regression model for which there is an extensive literature about model checking, the fit of a QTL mapping model is not extensively discussed, specially for Bayesian QTL mapping.

In this chapter, we briefly describe some of statistics used in model checking and propose using them to check the goodness of fit of a Bayesian QTL mapping model in Section 4.2. In Section 4.3, we apply the suggested checking analysis to the bone mineral density model estimated in Section 3.4.2.

4.2 Bayesian model checking

Let $\mathbf{y} = (y_1, y_2, ..., y_n)$ be a quantitative trait of n individuals from an F2 population. Assume this phenotype has been affected by K QTLs located at positions $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_K)$, $\lambda_k < \lambda_{k+1}$, and consider the following linear model for describing the phenotype as a linear function of QTLs' genotype

$$y_i = \mu + \sum_{k=1}^{K} \alpha_k Q_{ik} + \sum_{k=1}^{K} \delta_k (1 - |Q_{ik}|) + \varepsilon_i, \tag{4.1}$$

where μ is the average of expected values of genotypes AA and aa, α_k is the additive effect of the k-th QTL, δ_k is the dominance effect of k-th QTL, Q_{ik} represents the genotype of k-th QTL of the i-th individual coded as $-1, 0$ or 1 for aa, Aa or AA, respectively, $k = 1, ..., K$ and $i = 1, 2, ..., n$, $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$ is the random error, and ε_i and $\varepsilon_{i'}$ are supposed to be independent for $i \neq i'$.

One way of evaluating the fit of the estimated model at each observation in a Bayesian context is exploring the *a posteriori* distribution of the residuals. The ordinary residual of i-th individual, $i = 1, ..., n$, is defined as $\epsilon_i = y_i - \widehat{y}_i$, where $\widehat{y}_i = \widehat{\mu} - \sum_{k=1}^{K} \widehat{\alpha}_k q_{ik} - \sum_{k=1}^{K} \widehat{\delta}_k (1 - |q_{ik}|)$ is the predicted phenotype and $\widehat{\mu}$, $\widehat{\alpha}_k$ and $\widehat{\delta}_k$ are the Bayesian point estimate of μ, α_k and δ_k, $k = 1, ..., K$. An individual with a large residual is an outlier and indicates that the model falls short in predicting its response properly.

The studentized residual is given by $t_i = \frac{\epsilon_i}{\widehat{\sigma}\sqrt{1-h_{ii}}}$, where $\widehat{\sigma}$ is the Bayesian point estimate of σ, $h_{ii} = \mathbf{x}_i^T \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{x}_i$ is the classical measure of leverage and $\mathbf{X} = \begin{pmatrix} \mathbf{1} & \mathbf{q}_1 & 1 - |\mathbf{q}_1| & \cdots & \mathbf{q}_K & 1 - |\mathbf{q}_K| \end{pmatrix}$ is the design matrix of the QTL mapping model. We expect that roughly 5% of the studentized residuals are beyond $|2|$ and observations whose studentized residuals are higher than $|2|$ are considered as outlying individuals.

Suppose a converged Markov chain $\boldsymbol{\gamma}^1, \cdots, \boldsymbol{\gamma}^L$, where $\boldsymbol{\gamma} = (\mu, \sigma, \boldsymbol{\alpha}, \boldsymbol{\delta})$ is available. Then, a sample of the *a posteriori* distribution of ϵ_i and t_i for the i-th observation, $i = 1, ..., n$, is readily obtained. The l-th ordinary residual of i-th individual is calculated as $\epsilon_i^l = y_i - \widehat{y}_i^l$, where $\widehat{y}_i^l = \mu^l - \sum_{k=1}^{K} \alpha_k^l q_{ik} - \sum_{k=1}^{K} \delta_k^l (1 - |q_{ik}|)$ and the l-th studentized residual of i-th individual is given by $t_i^l = \frac{\epsilon_i^l}{\sigma^l\sqrt{1-h_{ii}}}$, $l = 1, ..., L$. We can identify atypical observations and check normality, independence and homocedasticity of random error through a normal probabilistic plot of studentized residuals mean, plot of studentized residuals *a posteriori* distribution versus index and plot of studentized residuals *a posteriori* distribution versus predicted values.

Another measure used to identify outliers is the *a posteriori* predictive ordinate (PPO) defined as

$$PPO_i = f_{Y_i|\mathbf{y}}(y_i), \tag{4.2}$$

for $i = 1, ..., n$. A low value of PPO_i indicates that the i-th observation is in the tail area of the distribution $f_{Y_i|\mathbf{y}}(\cdot)$ and if it is too low compared with PPO value of other observations, y_i is classified as an atypical observation. PPO_i is estimated as

$$\widehat{PPO}_i = \widehat{f}_{Y_i|\mathbf{y}}(y_i) = \frac{1}{L}\sum_{l=1}^{L} f_{Y_i|\boldsymbol{\gamma}^l}(y_i), \tag{4.3}$$

where $\{\boldsymbol{\gamma}^1, \cdots, \boldsymbol{\gamma}^L\}$ is the MCMC sequence of the joint *a posteriori* distribution of $\boldsymbol{\gamma}$.

In the computation of PPO the observed data are used twice. First in determining the *a posteriori* distribution of $\boldsymbol{\gamma}$ and later in evaluating the density function $f_{Y_i|\mathbf{y}}(\cdot)$ at y_i. To avoid the double use of data, Geisser (1980) suggest the CPO statistic given by

$$CPO_i = f_{Y_i|\mathbf{y}_{(i)}}(y_i), \tag{4.4}$$

for $i = 1, ..., n$, where $\mathbf{y}_{(i)}$ represents the data without observation y_i.

The following simple derivation of CPO, assuming conditional independence of y_i given $\boldsymbol{\gamma}$,

$$\begin{aligned}\frac{1}{f_{Y_i|\mathbf{y}_{(i)}}(y_i)} &= \frac{f_{\mathbf{Y}_{(i)}}(\mathbf{y}_{(i)})}{f_{\mathbf{Y}}(\mathbf{y})} = \int \frac{f_{\mathbf{Y}_{(i)}|\boldsymbol{\gamma}}(\mathbf{y}_{(i)})\pi(\boldsymbol{\gamma})}{f_{\mathbf{Y}}(\mathbf{y})} d\boldsymbol{\gamma} = \int \frac{f_{\mathbf{Y}|\boldsymbol{\gamma}}(\mathbf{y})\pi(\boldsymbol{\gamma})}{f_{Y_i|\boldsymbol{\gamma}}(y_i) f_{\mathbf{Y}}(\mathbf{y})} d\boldsymbol{\gamma} \\ &= \int \frac{1}{f_{Y_i|\boldsymbol{\gamma}}(y_i)} \pi(\boldsymbol{\gamma}|\mathbf{y}) d\boldsymbol{\gamma} = E_{\boldsymbol{\gamma}|\mathbf{y}}\left(\frac{1}{f_{Y_i|\boldsymbol{\gamma}}(y_i)}\right)\end{aligned} \tag{4.5}$$

allows us to estimate CPO_i using the MCMC output as an harmonic mean

$$\widehat{CPO}_i = \left(\frac{1}{L}\sum_{l=1}^{L}\frac{1}{f_{Y_i|\boldsymbol{\gamma}^l}(y_i)}\right)^{-1}. \tag{4.6}$$

Researchers usually prefer to analyze $ICPO = 1/CPO$ instead of CPO since $ICPO$ of atypical observations is more discrepant of $ICPO$ of other observations than CPO. Consequently, atypical individuals are more evident if we compare $ICPO$ values instead of CPO values.

Lesaffre & Lawson (2012) also propose methods for analyzing the model sensitivity to specific observations and identifying global influence individuals that have a great impact on $\boldsymbol{\gamma}$ estimate. One of them is the global influence measure $I_i(h_1)$, $i = 1, ..., n$, that is a divergence measure between $\pi(\boldsymbol{\gamma}|\mathbf{y})$ and $\pi(\boldsymbol{\gamma}|\mathbf{y}_{(i)})$ defined from

$$\frac{\pi(\boldsymbol{\gamma}|\mathbf{y}_{(i)})}{\pi(\boldsymbol{\gamma}|\mathbf{y})} = \frac{f_{\mathbf{Y}_{(i)}|\boldsymbol{\gamma}}(\mathbf{y}_{(i)}) f_{\mathbf{Y}}(\mathbf{y})}{f_{\mathbf{Y}|\boldsymbol{\gamma}}(\mathbf{y}) f_{\mathbf{Y}_{(\mathbf{i})}}(\mathbf{y}_{(i)})} = \frac{CPO_i}{f_{Y_i|\boldsymbol{\gamma}}(y_i)} = h_{1i}(\boldsymbol{\gamma}) \tag{4.7}$$

and estimated as $\widehat{I}_i(h_1) = \frac{1}{L}\sum_{l=1}^{L} \log h_{1i}(\boldsymbol{\gamma}^l)$. Values of $\widehat{I}_i(h_1)$ much higher than zero indicate global influence observations.

Other globally influence measure is the observed variance of $\left\{\frac{h_{1i}(\gamma^l)}{\sum_{l=1}^{L} h_{1i}(\gamma^l)}, l = 1, ..., L\right\}$, $i = 1, ..., n$, called the sequence of normalized importance weight for deleting the i-th observation at l-th MCMC iteration. A high weights variance indicates the i-th individual is very influent in just a small number of iterations and does not show a stable influence behavior.

4.3 Checking the bone mineral density QTL mapping model

In Section 3.4.2 we select and estimate a QTL mapping model for the bone mineral data set (Wergedal *et al*, 2006) using the proposed data-driven reversible jump (DDRJ) scheme. Analyzing each chromosome separately since they are independent, we identify $K = 6$ significant QTLs located in chromosomes 1, 7, 9, 11, 12 and 18. Since their effects estimates shown in Table 3.6 are not jointly estimated, we fix the estimated location and genotype of the $K = 6$ QTLs and run a MCMC chain to estimate $\boldsymbol{\gamma}$ jointly. We run the Gibbs steps to update $\boldsymbol{\gamma}$ $L = 55000$ iterations, discard the first 5000 iterations and take one for every 10 iterations. The convergence is verified using trace plots.

Table 4.1 shows estimates (*a posteriori* mean) and 95% credibility intervals for additive and dominance effects of QTLs. Comparing the estimates of full model shown in Table 4.1 with estimates of separated by chromosomes model in Table 3.6, we observe that they are very similar.

Table 4.1: Point estimates and 95% credibility intervals for parameters.

Chromossome	α	δ
1	0.005 (0.001,0.009)	0.010 (0.003,0.016)
7	0.007 (0.002,0.012)	0.017 (0.011,0.023)
9	0.005 (0.006,0.015)	-0.005 (-0.012,0.001)
11	-0.011 (-0.016,-0.007)	0.001 (-0.005,0.008)
12	0.006 (0.002,0.011)	0.003 (-0.004,0.010)
18	-0.009 (-0.014,-0.004)	0.004 (-0.002,0.011)

Using the converged MCMC sequence $\{\boldsymbol{\gamma}^1, \cdots, \boldsymbol{\gamma}^L\}$, we calculate a sequence of ordinary and studentized residuals for each individual. Figure 4.1 shows diagnostic statistics of goodness of the model fit. Clearly we observe three outlying observations (423, 454 and 496) in normal probabilistic plot of *a posteriori* mean of studentized residuals (Figure 4.1 (A)), in *a posteriori* distribution of studentized residuals versus iterations (Figure 4.1 (B)) and predicted value plots (Figure 4.1 (C)). Individuals 423, 454 and 496 are also identified (Figure 4.1 (D), (E) and (F)) as high influential cases since their ICPO, global influence and weights variance values are too discrepant.

We excluded the observations 423, 454 and 496 from the data set and run the DDRJ scheme for each chromosome to verify the influence of these outlying observations in identifying and

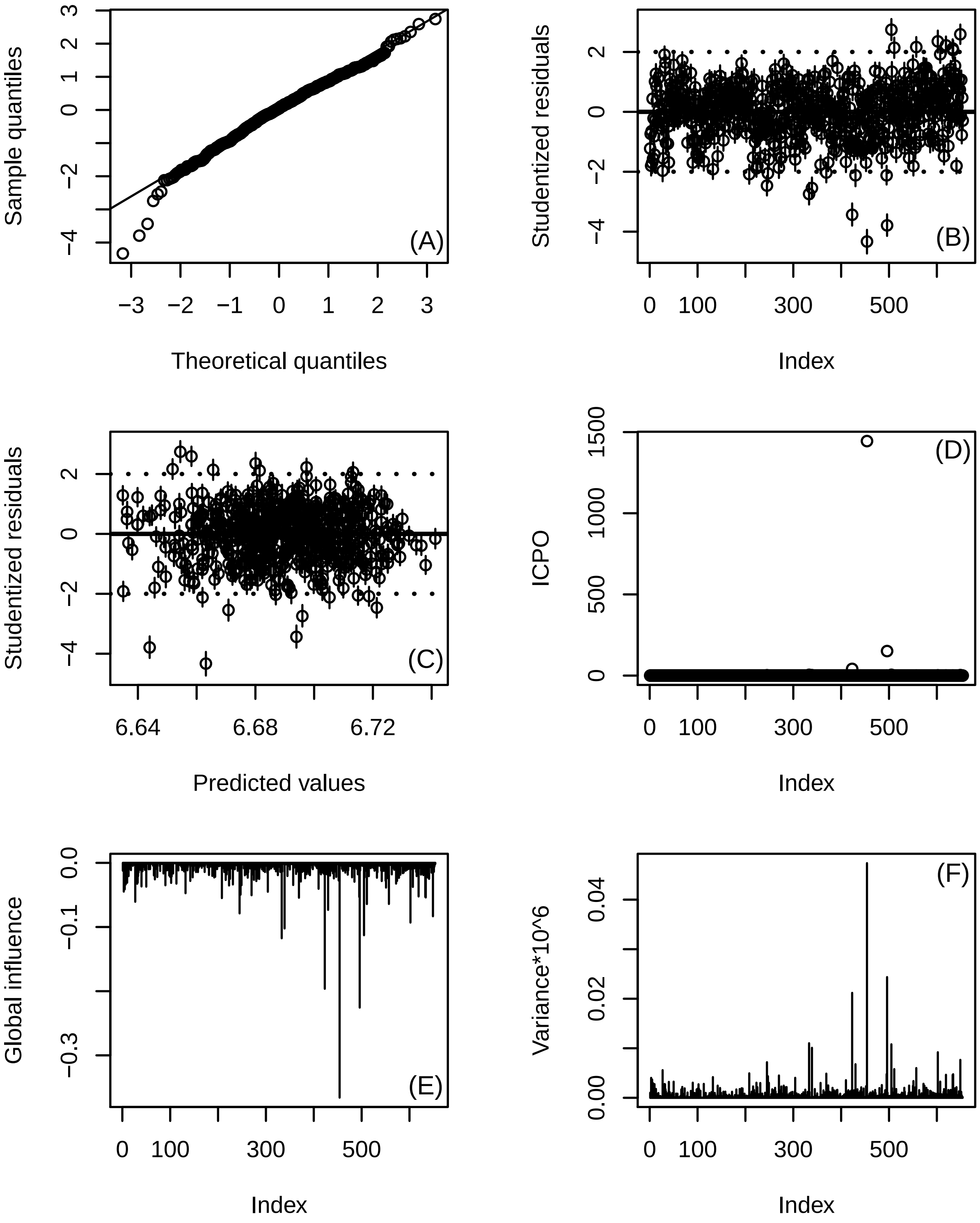


Figure 4.1: Diagnostic measures of goodness of fit: (A) normal probabilistic plot of *a posteriori* mean of studentized residuals; (B) the *a posteriori* distribution of studentized residuals versus iterations plot; (C) the *a posteriori* distribution of studentized residuals versus predicted values plot; (C) index plot of ICPO; (D) index plot of global influence; (E) index plot of normalized importance weight variance.

locating the QTLs. We run DDRJ chains $L = 55000$ iterations, discard the first 5000 iterations and take one for every 10 iterations. The chains are initialized with $K = 0$ which represents a model without QTLs. Convergence is verified using trace plots.

Table 4.2 shows *a posteriori* probabilities for K calculated as the relative frequency of each value of K in the sequence of each chromosome. Comparing these results with results shown in Table 3.5, we note that the *a posteriori* probabilities for K in chromosomes 1, 17 and 18 have considerable changes. The evidence of a QTL in chromosome 1 and 17 disappeared indicating that these QTLs are not important to explain bone mineral density variability when outliers are excluded from data. The evidence of a QTL in chromosome 19 is higher without atypical observations. A QTL's presence is still strong evident in chromosome 7 and moderate in chromosomes 9, 10, 11, 12 and 18. Then, we conclude that atypical individuals can influence QTLs' identification.

Table 4.2: The *a posteriori* probabilities for K in each chromosome without atypical observations.

K	Cromossomo									
	1	2	3	4	5	6	7	8	9	10
0	0.970	0.980	0.980	0.950	0.998	0.900	0.380	0.980	0.830	0.810
1	0.030	0.020	0.020	0.050	0.002	0.090	0.610	0.020	0.170	0.170
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
K	Cromossomo									
	11	12	13	14	15	16	17	18	19	
0	0.590	0.830	0.990	0.990	0.980	0.990	0.960	0.900	0.500	
1	0.390	0.160	0.010	0.010	0.020	0.010	0.040	0.100	0.500	
2	0.010	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	

Tabela 4.3 shows the estimated location (*a posteriori* mean) of QTLs (cM) and their credibility intervals 95%. Comparing results of Tables 4.3 and 3.6, we observe that the point estimate of QTLs' location in chromosome 7, 9 ,10, 11, 12, 18 and 19 is similar and not too sensitive to outliers. The credibility intervals are shorter when atypical observations are not considered.

Table 4.3: Estimates and 95% credibility interval for QTLs' location without atypical observations.

Chromosome	λ (cM)
7	65.7 (60.6,69.6)
9	67.2 (51.4,60.8)
10	58.6 (46.9,64.8)
11	32.6 (18.5,41.8)
12	28.2 (9.6,55.3)
18	43.6 (22.0,56.0)
19	44.5 (35.0,51.6)

Here we do not show the QTLs's effects estimates and 95% credibility interval obtained when we fit the model for each chromosome, but QTLs in chromosomes 10 and 19 are nonsignificant and we exclude both QTLs of the estimation of the full model. We fix the estimated location and genotype of the $K = 5$ QTLs and run a MCMC chain to estimate $\boldsymbol{\gamma}$ jointly. We run the Gibbs steps to update $\boldsymbol{\gamma}$ $L = 55000$ iterations, discard the first 5000 iterations and take one for every 10 iterations. The convergence is verified using trace plots.

Table 4.4 shows QTLs' effects estimates and their 95% credibility interval. Comparing results of Tables 4.4 and 3.6, we note that, although the point estimate of QTL's effects in chromosomes 7, 9, 11, 12 and 18 are similar, the 95% credibility intervals are shorter when the outliers are not in data set. The dominance effect of the QTL in chromosome 12 is significant when we do not consider the outliers.

Table 4.4: Estimates and 95% credibility interval for parameters of model with 5 QTLs and without outliers.

Chromosome	α	δ
7	0.008 (0.004,0.013)	0.014 (0.008,0.020)
9	0.012 (0.007,0.017)	-0.004 (-0.011,0.002)
11	-0.009 (-0.014,-0.005)	0.001 (-0.006,0.007)
12	0.006 (0.001,0.011)	0.011 (0.005,0.017)
18	-0.007 (-0.012,-0.003)	-0.003 (-0.009,0.003)

Figure 4.2 shows diagnostic statistics of goodness of the model fit without the outliers. We observe the fitted model predicts well the response for all individuals since we do not have high values of studentized residuals and they are randomly distributed around zero. No observation is an influential case because no value of ICPO, global influence statistic and weights variance is discrepant.

4.4 Discussion

We briefly describe some diagnostic measure of goodness of fit for Bayesian models and apply them to evaluate a QTL mapping model fit. In Bayesian context, the model checking is usually carried out using the *a posteriori* distribution of residuals and *a posteriori* predictive statistics.

In the analyzed data set, the bone mineral density, we observe that atypical observations can influence the identification of QTLs evidencing false QTLs or hiding true QTLs.

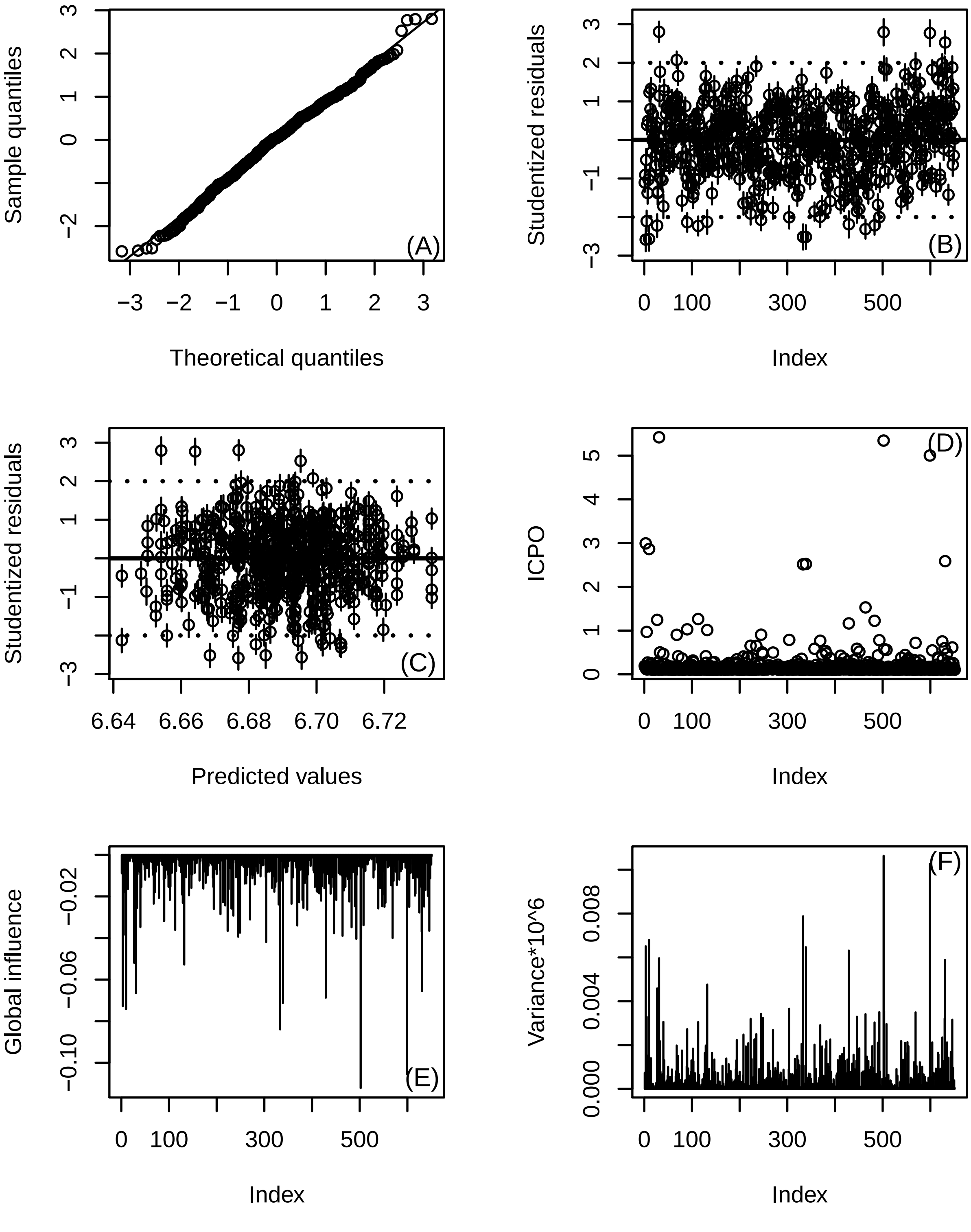


Figure 4.2: Diagnostic measures of goodness of fit: (A) normal probabilistic plot of *a posteriori* mean of studentized residuals; (B) the *a posteriori* distribution of studentized residuals versus iterations plot; (C) the *a posteriori* distribution of studentized residuals versus predicted values plot; (C) index plot of ICPO; (D) index plot of global influence; (E) index plot of normalized importance weight variance.

Chapter

5

A model for QTL mapping of pedigree data [1]

In this chapter, we propose a model for QTL mapping of pedigree data to explain a phenotype as a function of QTLs' fixed effects. The model considers the detailed dependence structure of familiar dependence. It combines the Mendelian probability of inheritance of parents' genotype and the correlation between the *loci* of markers and QTLs. This is a differential to models which use only Mendelian segregation or only the correlation between markers and QTLs to estimate transmission probabilities.

We use the Bayesian approach to estimate the number of QTLs, their location and the additive and dominance effects. We compare the performances of the proposed method and the variance component model using simulated data sets. Under tested conditions the proposed method shows an excellent performance for estimating the number of QTLs and accuracy to estimate QTL positions and their effects.

5.1 Introduction

Variance component (VC) models have been used to identify quantitative trait *loci* (QTL) and estimate their contribution to phenotypic variance for pedigree data. Considering QTLs' effects as random effects is usual in QTL mapping of pedigree data and data's covariance structure is defined as a function of the assumed random effects' covariance structure, see for instance Almasy & Blangero (1998). The mixed VC model have been extended to accommodate general pedigrees of arbitrary size and complexity (Almasy & Blangero, 1998; Comuzzie *et al.* , 1997). However, there are situations where these models are not efficient to identify QTLs' effects, confounding genetic and environmental sources of variation (Lee *et al*, 2010; Suzanne,

[1] This chapter is based on the manuscript "A model for QTL mapping of pedigree data" submitted for publication (Zuanetti & Milan, Submitted).

2008). Accommodating effects like epistasis and interaction between environmental and QTLs' effects is not straightforward in variance component model (Habier *et al*, 2010).

Elston & Stewart (1971), Cannings *et al.* (1978) and Lander & Green (1987) consider the dependence between individual's genotypes and suggest computational algorithms to reduce the complexity involved in the likelihood and genotype probability's computation, known as peeling algorithms. Their basic assumption is that the phenotype of each individual only depends on its own associated genotypes and different *loci* are unlinked. The offspring's genotype in a specific *locus* depends only on parents's genotype through Mendelian segregation. Lander & Green (1987) consider that consecutive *loci* are linked and use hidden Markov model to calculate the probability of a specific gamete (paternal or maternal) based in the previous gamete. Elston & Stewart (1971) consider a nuclear family at a time and Lander & Green (1987) considers one *locus* at a time. See Fishelson & Geiger (2002), Lauritzen & Sheehan (2003) and Li *et al.* (2009) for a revision of these and other approaches and applications.

Heath (1997) proposes a reversible jump scheme to estimate the number of QTLs and their fixed effects using the single-*locus* peeling method (Cannings *et al*, 1978; Elston & Stewart , 1971). It assumes that individual's genetic properties depend on parents genes considering only Mendelian inheritance to explain the genotype of offsprings in a specific *locus*.

We propose a model which describes the dependence structure among individuals and variables through conditional independence structures. The proposed model considers the segregation of a gene in a *locus* depends on Mendelian segregation and also is correlated with genotypes of flanking markers (at left and at right). The assumption of linked genotypes of nearby *loci* is desirable because the chance of a genetic recombination through meiosis process is low and the genotype of flanking *loci* is informative and improves the model which considers sole the Mendel probability of inheritance for parents' genotypes. We also extend the data-driven reversible jump MCMC (DDRJ) proposed in Zuanetti & Milan (2016) for mapping QTLs of independent individuals to estimate the number, positions, additive and dominance effects of QTLs in pedigree data. DDRJ also provides a better computational performance when compared with previous approaches.

The chapter is organized as follows. Section 5.2 proposes the model and conditional independence structures for quantitative traits in pedigree data and its likelihood function. It also proposes how to consider Mendelian segregation and genotype of linked flanking *loci* to determine the genotype of a specific *locus*; in Section 5.3 we address the Bayesian approach for the model including the DDRJ to estimate the number of QTLs; in Section 5.4 we analyze the performance of the method comparing it with the usual mixed variance component model. Finally, we discuss the results in Section 5.5 and show appendices in Section 5.6.

5.2 Model for quantitative traits of pedigree data

Consider $\mathbf{y} = (y_1, y_2, ..., y_n)$ the quantitative trait of n individuals with a known familiar structure (pedigree). Assume that the phenotype is affected by K QTLs located at positions $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_K)$, $\lambda_k < \lambda_{k+1}$ for $k = 1, ..., K-1$, and that m different genotyped markers are available.

Phenotype y_i for the i-th individual is modeled by the linear model

$$y_i = \mu + \sum_{k=1}^{K} \alpha_k Q_{ik} + \sum_{k=1}^{K} \delta_k (1 - |Q_{ik}|) + \varepsilon_i, \tag{5.1}$$

where μ is the average of expected values of genotypes AA and aa; α_k and δ_k are the additive and dominance effects for the k-th QTL; Q_{ik} corresponds to the genotype of k-th QTL of the i-th individual which is coded as $-1, 0$ or 1 for aa, Aa or AA, respectively; and ε_i is a random error, $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$.

The phenotype may also be affected by environmental covariates, interactions between QTLs or between covariates and QTLs. The model in (5.1) does not include these effects but extensions, as modeling environmental covariates as fixed effects, for instance, are straightforward.

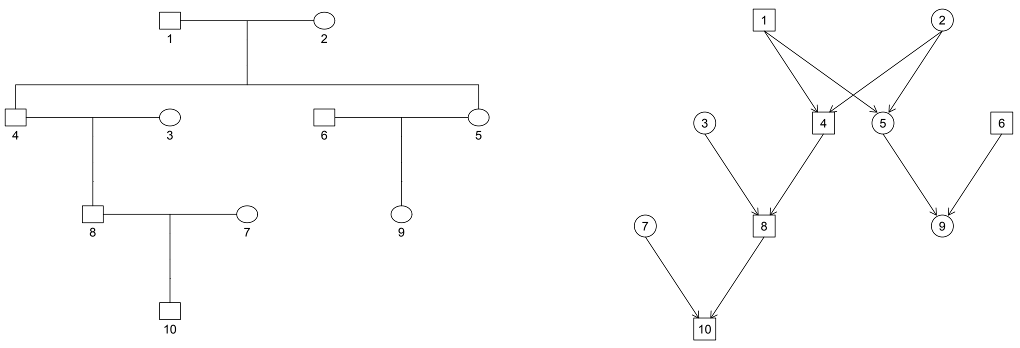
The data consists of the observations $\mathbf{y} = (y_1, y_2, ..., y_n)$; the markers' genotypes $\mathbf{M} = \{m_{ij}\}$, m_{ij} coded as $-1, 0$ or 1 for aa, Aa or AA respectively; $\mathbf{D} = (D_1, D_2, ..., D_m)$ are the distances (in centiMorgan - cM) between each marker and the first marker, $D_1 = 0$; and the pedigree structure. We assume that there is at most one QTL between each pair of consecutive markers.

We consider pedigree as a group of individuals with full specification of parentage. In a pedigree, we define as nonfounders those individuals whose parents are in the group and as founders those individuals whose parents are not in the group. We assume that founders are unrelated and both parents of any nonfounder are available in the pedigree.

A pedigree may be represented by a directed acyclic graph (DAG) with well-defined dependence structure. Nodes denote individuals and arrows connect individuals to their offsprings showing the dependence structure. Figure 5.1 shows a pedigree and its respective DAG with 10 individuals, females are represented by circles and males by squares. Individuals 1, 2, 3, 6 and 7 are founders and individuals 4, 5, 8, 9 and 10 are nonfounders. We consider a nonfounder and its nondescending individuals independent given its parents.

Figure 5.2 shows the complete graphical model for individuals 1, 2 and 4 represented in Figure 5.1. Circles represent random variables and squares represent deterministic values. Each node represents a genotype or a phenotype of an individual and the arrows show the relationship among these nodes.

The model in Figure 5.2 presents the following conditional independence relations:



(a) Pedigree

(b) DAG

Figure 5.1: Pedigree represented by a DAG.

1. Given its QTLs' genotypes, the phenotype of the i-th individual is independent of any other individual's phenotype, $y_i \perp y_{i'}|\mathbf{Q}_i = (q_{i1}, ..., q_{iK})$, for $i \neq i'$ and $i, i' \in \{1, ..., n\}$.
2. The genotype of the k-th QTL of the i-th individual is independent of any QTLs' genotype of i-th individual and k-th QTL's genotype of nondescending individuals, given its flanking markers' genotypes and k-th QTL's genotype of the i-th individual's parents, *i.e.*, $Q_{ik} \perp \{Q_{ik'}, Q_{ND,k}\} | Q_{\mathbf{p}_i k}, M_{ir_k}, M_{il_k}$, for $k \neq k'$ and $k, k' \in \{1, ..., K\}$, where ND is the set of nondescendents of the i-th individual; $\mathbf{p}_i$ are the parents of the i-th individual; M_{ir_k} and M_{il_k} are the genotypes of the markers at right and left of the k-th QTL, respectively, for the i-th individual.

We assume that given the QTLs' genotypes, the polygenic effects are absent and y_i's are independent, as in Heath (1997) and Balding *et al.* (2008). For polygenic effect we means QTLs with minor phenotypic effects that generally remain below the detection threshold (Bink *et al*, 2014).

Considering F the set of founders and F^c the set of nonfounders, the joint distribution of $\mathbf{y}$ and $\mathbf{Q}$ is

$$\begin{aligned} f_{\mathbf{Y},\mathbf{Q}|\mathbf{M},\mathbf{D}}(\mathbf{y},\mathbf{q}) &= f_{\mathbf{Y}|\mathbf{q}}(\mathbf{y}) Pr\left(\mathbf{Q} = \mathbf{q}|\mathbf{M},\mathbf{D}\right) \\ &= \prod_{i=1}^{n} f_{Y_i|\mathbf{q}_i}(y_i) \prod_{i \in F} Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i,\mathbf{D}\right) \prod_{i \in F^c} Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i,\mathbf{D},\mathbf{Q}_{\mathbf{p}_i},\mathbf{M}_{\mathbf{p}_i}\right) \\ &= \prod_{i \in F} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i,\mathbf{D}\right) \prod_{i \in F^c} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i,\mathbf{D},\mathbf{Q}_{\mathbf{p}_i},\mathbf{M}_{\mathbf{p}_i}\right). \end{aligned} \tag{5.2}$$

Using conditional independence we have
$Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i,\mathbf{D}\right) = \prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}\right)$;

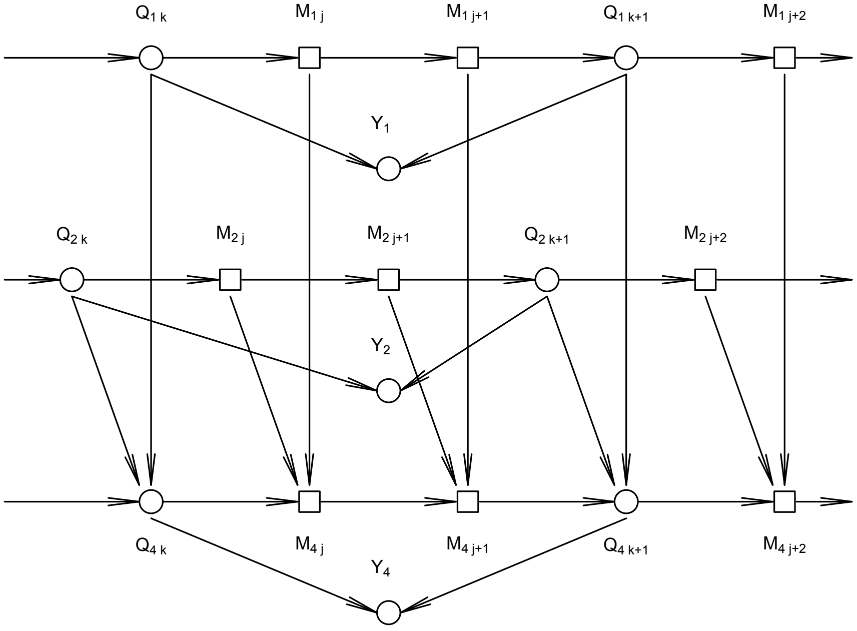


Figure 5.2: DAG for a family of a pair of parents and their offspring.

$Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i, \mathbf{D}, \mathbf{Q}_{\mathbf{p}_i}, \mathbf{M}_{\mathbf{p}_i}\right) = \prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k}\right)$;
$\sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}\right) = 1$ and
$\sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k}\right) = 1$.

The marginal distribution of $\mathbf{y}$ is

$$\begin{aligned} f_{\mathbf{Y}|\mathbf{M},\mathbf{D}}(\mathbf{y}) &= \sum_{\mathbf{q}} f_{\mathbf{Y},\mathbf{Q}|\mathbf{M},\mathbf{D}}(\mathbf{y}, \mathbf{q}) \\ &= \prod_{i\in F} \sum_{q_{i1}} \cdots \sum_{q_{iK}} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(Q_{i1} = q_{i1}, ..., Q_{iK} = q_{iK}|\mathbf{M}_i, \mathbf{D}\right) \\ &\times \prod_{i\in F^c} \sum_{q_{i1}} \cdots \sum_{q_{iK}} f_{Y_i|\mathbf{q}_i}(y_i) Pr\left(Q_{i1} = q_{i1}, ..., Q_{iK} = q_{iK}|\mathbf{M}_i, \mathbf{D}, \mathbf{Q}_{\mathbf{p}_i}, \mathbf{M}_{\mathbf{p}_i}\right), \end{aligned} \tag{5.3}$$

where $\sum_{q_{i1}} \cdots \sum_{q_{iK}} Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i, \mathbf{D}\right) = \prod_{k=1}^{K} \sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}\right) = 1$ and $\sum_{q_{i1}} \cdots \sum_{q_{iK}} Pr\left(\mathbf{Q}_i = \mathbf{q}_i|\mathbf{M}_i, \mathbf{D}, \mathbf{Q}_{\mathbf{p}_i}, \mathbf{M}_{\mathbf{p}_i}\right) = \prod_{k=1}^{K} \sum_{q_{ik}} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k}\right) = 1$.

Equation (5.3) characterizes each y_i as a mixture of 3^K distributions.

From equation (5.2), the likelihood function for $\boldsymbol{\theta} = (K, \mu, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \boldsymbol{\delta}, \sigma^2)$ is

$$\begin{aligned} L(\boldsymbol{\theta}|\mathbf{y}, \mathbf{q}) &= f_{\mathbf{Y},\mathbf{Q}|\mathbf{M},\mathbf{D}}(\mathbf{y}, \mathbf{q}) \\ &= \prod_{i\in F} \left(f_{Y_i|\mathbf{q}_i}(y_i) \prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}\right) \right) \\ &\times \prod_{i\in F^c} \left(f_{Y_i|\mathbf{q}_i}(y_i) \prod_{k=1}^{K} Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k}\right) \right), \end{aligned} \tag{5.4}$$

where $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_K)$, $\boldsymbol{\delta} = (\delta_1, ..., \delta_K)$, $f_{Y_i|\mathbf{q}_i}(y_i)$ is the normal density function with $\mu + \sum_{k=1}^{K} \alpha_k q_{ik} + \sum_{k=1}^{K} \delta_k (1 - |q_{ik}|)$ mean and variance σ^2, $Pr\left(Q_{ik} = q_{ik} | M_{ir_k}, M_{il_k}, D_{l_k}, D_{r_k}\right)$ is the conditional probability of founders' QTL given flanking markers genotypes as defined by Stephens & Fisch (1998) and $Pr\left(Q_{ik} = q_{ik} | M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k}\right)$ is the conditional probability of nonfounders' QTL given flanking markers genotypes and parents' QTL and markers genotype, known as transmission probabilities.

5.2.1 Transmission probabilities

Similar to meiosis or segregation indicators presented by Lander & Green (1987) and Lauritzen & Sheehan (2003), we use auxiliary variables and maternal and paternal allele's probabilities to define the transmission probabilities of nonfounder individuals.

Let $S_{p_{it}}$ and $S_{m_{it}}$ be the meiosis or segregation indicators of i-th individual at *locus* t defined as

$$S_{p_{it}} = \begin{cases} 1, \text{ if paternal allele of } t\text{-th } \textit{locus} \text{ is inherited from } i\text{'s paternal grandfather} \\ 0, \text{ if paternal allele of } t\text{-th } \textit{locus} \text{ is inherited from } i\text{'s paternal grandmother} \end{cases}$$

and

$$S_{m_{it}} = \begin{cases} 1, \text{ if maternal allele of } t\text{-th } \textit{locus} \text{ is inherited from } i\text{'s maternal grandfather} \\ 0, \text{ if maternal allele of } t\text{-th } \textit{locus} \text{ is inherited from } i\text{'s maternal grandmother.} \end{cases}$$

The sequence of paternal meiosis indicators of m markers and K QTLs $\mathbf{S}_p = (S_{p_{M_1}}, ..., S_{p_k}, ..., S_{p_{M_m}})$ and the sequence of maternal meiosis indicators $\mathbf{S}_m = (S_{m_{M_1}}, ..., S_{m_k}, ..., S_{m_{M_m}})$, both ordered by *loci* occurrence in the chromosome are nonhomogenous Markov chains where s_{p_t} and s_{m_t} are realizations of S_{p_t} and S_{m_t} and $s_{p_t}, s_{m_t} \in \{0, 1\}$ for $t = M_1, ..., k, ..., M_m$. The transition probability matrix from *locus* $t-1$ to t is given by $P_{S_t} = \begin{pmatrix} 1 - r_t & r_t \\ r_t & 1 - r_t \end{pmatrix}$ for both chains, where r_t is the recombination fraction between *loci* $t-1$ and t calculated through Haldane function and the elements of P_{S_t} represent $Pr\left(S_{p_t} = s_{p_t} | S_{p_{t-1}}, D_{l_t}\right)$ and $Pr\left(S_{m_t} = s_{m_t} | S_{m_{t-1}}, D_{l_t}\right)$.

Now consider $L_{p_{ik}}$ and $L_{m_{ik}}$ the allele of the k-th QTL which i-th individual inherited from its father (paternal) and mother (maternal) respectively, where the observed values $l_{p_{ik}}, l_{m_{ik}} \in \{1, 0\} \equiv \{A, a\}$. Therefore, $L_{p_{ik}} = \begin{cases} l_{p_{p_i k}}, \text{ if } s_{p_{ik}} = 1 \\ l_{m_{p_i k}}, \text{ if } s_{p_{ik}} = 0 \end{cases}$ and $L_{m_{ik}} = \begin{cases} l_{p_{m_i k}}, \text{ if } s_{m_{ik}} = 1 \\ l_{m_{m_i k}}, \text{ if } s_{m_{ik}} = 0 \end{cases}$, where p_i is the i-th individual's father and m_i is the i-th individual's mother, respectively, and

$$Pr\left(L_{p_{ik}} = l_{p_{ik}} | L_{p_{iM_{l_k}}}, D_{l_k}, L_{p_{p_i k}}, L_{m_{p_i k}}\right) = Pr\left(S_{p_{ik}} = s^*_{p_{ik}} | S_{p_{iM_{l_k}}}, D_{l_k}\right),$$

where $s^*_{p_{ik}}$ is the specific value of $s_{p_{ik}}$ which provides the inheritance of allele $l_{p_{ik}}$ and is shown in Table 5.1 considering heterozygous father, and

$$Pr\left(L_{m_{ik}} = l_{m_{ik}} | L_{m_{iM_{l_k}}}, D_{l_k}, L_{p_{m_i k}}, L_{m_{m_i k}}\right) = Pr\left(S_{m_{ik}} = s^*_{m_{ik}} | S_{m_{iM_{l_k}}}, D_{l_k}\right),$$

where $s^*_{m_{ik}}$ is the specific value of $s_{m_{ik}}$ which provides the inheritance of allele $l_{m_{ik}}$ and is specified in a similar way by Table 5.1 considering heterozygous mother. Cases of homozygous parents are discussed in Section 5.2.1.

Table 5.1: Values of $s^*_{p_{ik}}$ for each combination of $l_{p_{ik}}$, $l_{p_{p_ik}}$ and $l_{m_{p_ik}}$ considering heterozygous father.

		$l_{p_{p_ik}} = 0$ and $l_{m_{p_ik}} = 1$	$l_{p_{p_ik}} = 1$ and $l_{m_{p_ik}} = 0$
$l_{p_{ik}}$	0	1	0
	1	0	1

Note that if $\mathbf{S}_{p_i}$, $\mathbf{S}_{mi}$ and $\mathbf{L}_{p_{p_i}}$, $\mathbf{L}_{mp_i}$, $\mathbf{L}_{p_{m_i}}$ and $\mathbf{L}_{mm_i}$ are known the genotype of the i-th individual is completely defined including its QTLs' genotypes.

Considering the genetic inheritance of parents independent, $L_{p_{ik}} \perp L_{m_{ik}}$, the transmission probabilities are defined as

$$Pr\left(Q_{ik}{=}{-1}|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},\mathbf{Q}_{\mathbf{p}_ik},\mathbf{M}_{\mathbf{p}_ir_k}\right)=$$

$$\begin{aligned}
&=Pr\left(L_{p_{ik}}{=}0|L_{p_iM_{l_k}},L_{p_iM_{r_k}},D_{l_k},D_{r_k},L_{p_{p_ik}},L_{m_{p_ik}},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)\\
&\times Pr\left(L_{m_{ik}}{=}0|L_{m_{iM_{l_k}}},L_{m_{iM_{r_k}}},D_{l_k},D_{r_k},L_{p_{m_ik}},L_{m_{m_ik}},L_{p_{m_iM_{r_k}}},L_{m_{m_iM_{r_k}}}\right),
\end{aligned} \tag{5.5}$$

$$Pr\left(Q_{ik}{=}1|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},\mathbf{Q}_{\mathbf{p}_ik},\mathbf{M}_{\mathbf{p}_ir_k}\right)=$$

$$\begin{aligned}
&=Pr\left(L_{p_{ik}}{=}1|L_{p_iM_{l_k}},L_{p_iM_{r_k}},D_{l_k},D_{r_k},L_{p_{p_ik}},L_{m_{p_ik}},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)\\
&\times Pr\left(L_{m_{ik}}{=}1|L_{m_{iM_{l_k}}},L_{m_{iM_{r_k}}},D_{l_k},D_{r_k},L_{p_{m_ik}},L_{m_{m_ik}},L_{p_{m_iM_{r_k}}},L_{m_{m_iM_{r_k}}}\right)
\end{aligned} \tag{5.6}$$

and $Pr\left(Q_{ik}{=}0|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},\mathbf{Q}_{\mathbf{p}_ik},\mathbf{M}_{\mathbf{p}_ir_k}\right)=$

$$\begin{aligned}
&=Pr\left(L_{p_{ik}}{=}0|L_{p_iM_{l_k}},L_{p_iM_{r_k}},D_{l_k},D_{r_k},L_{p_{p_ik}},L_{m_{p_ik}},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)\\
&\times Pr\left(L_{m_{ik}}{=}1|L_{m_{iM_{l_k}}},L_{m_{iM_{r_k}}},D_{l_k},D_{r_k},L_{p_{m_ik}},L_{m_{m_ik}},L_{p_{m_iM_{r_k}}},L_{m_{m_iM_{r_k}}}\right)\\
&+Pr\left(L_{p_{ik}}{=}1|L_{p_iM_{l_k}},L_{p_iM_{r_k}},D_{l_k},D_{r_k},L_{p_{p_ik}},L_{m_{p_ik}},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)\\
&\times Pr\left(L_{m_{ik}}{=}0|L_{m_{iM_{l_k}}},L_{m_{iM_{r_k}}},D_{l_k},D_{r_k},L_{p_{m_ik}},L_{m_{m_ik}},L_{p_{m_iM_{r_k}}},L_{m_{m_iM_{r_k}}}\right),
\end{aligned} \tag{5.7}$$

where $Pr\left(L_{p_{ik}}{=}l_{p_{ik}}|L_{p_iM_{l_k}},L_{p_iM_{r_k}},D_{l_k},D_{r_k},L_{p_{p_ik}},L_{m_{p_ik}},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)=$

$$\begin{aligned}
&=\frac{Pr\left(L_{p_{ik}}{=}l_{p_{ik}}|L_{p_iM_{l_k}},D_{l_k},L_{p_{p_ik}},L_{m_{p_ik}}\right)Pr\left(L_{p_iM_{r_k}}{=}l_{p_iM_{r_k}}|L_{p_{ik}},D_{r_k},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)}{\sum_{l_{p_{ik}}=0}^{1}Pr\left(L_{p_{ik}}{=}l_{p_{ik}}|L_{p_iM_{l_k}},D_{l_k},L_{p_{p_ik}},L_{m_{p_ik}}\right)Pr\left(L_{p_iM_{r_k}}{=}l_{p_iM_{r_k}}|L_{p_{ik}},D_{r_k},L_{p_{p_iM_{r_k}}},L_{m_{p_iM_{r_k}}}\right)}\\
&=\frac{Pr\left(S_{p_{ik}}{=}s^*_{p_{ik}}|S_{p_iM_{l_k}},D_{l_k}\right)Pr\left(S_{p_iM_{r_k}}{=}s^*_{p_iM_{r_k}}|S_{p_{ik}},D_{r_k}\right)}{\sum_{s_{p_{ik}}=0}^{1}Pr\left(S_{p_{ik}}{=}s_{p_{ik}}|S_{p_iM_{l_k}},D_{l_k}\right)Pr\left(S_{p_iM_{r_k}}{=}s^*_{p_iM_{r_k}}|S_{p_{ik}},D_{r_k}\right)}
\end{aligned} \tag{5.8}$$

and $Pr\left(L_{m_{ik}}=l_{m_{ik}}|L_{m_{iM_{l_k}}},L_{m_{iM_{r_k}}},D_{l_k},D_{r_k},L_{p_{m_ik}},L_{m_{m_ik}},L_{p_{m_iM_{r_k}}},L_{m_{m_iM_{r_k}}}\right)$ is defined similarly to equation (5.8).

Special cases

When one or both parents are homozygous in a specific *locus* t, *i.e.*, $L_{p_{p_it}} = L_{m_{p_it}} = l$ and/or $L_{p_{m_it}} = L_{m_{m_it}} = l$, for $l \in \{1, 0\}$, the parents' genotype are not informative in relation to the origin of each allele of their offspring and

$$Pr\left(L_{p_{it}} = l|L_{p_{iM_{l_t}}}, D_{l_t}, L_{p_{p_it}} = l, L_{m_{p_it}} = l\right) = 1$$

and/or

$$Pr\left(L_{m_{it}} = l|L_{m_{iM_{l_t}}}, D_{l_t}, L_{p_{m_it}} = l, L_{m_{m_it}} = l\right) = 1$$

by Mendel's probability of genetic inheritance.

If $r_t \approx 1/2 \approx r_{t+1}$, indicating that the *loci* $t-1$, t and $t+1$ are so far apart that they segregate independently, the transmission probabilities are also Mendelian and depend only on parents' genotype. In this case, $Pr\left(Q_{ik} = q_{ik}|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q_p}_ik, \mathbf{M_p}_ir_k\right) = Pr\left(Q_{ik} = q_{ik}|\mathbf{Q_p}_ik\right)$ given by Table 5.2 whose elements represent $Pr\left(Q_{ik} = x|Q_{p_ik} = u, Q_{m_ik} = v\right)$, for $u, v, x = -1, 0, 1$, $i = 1, ..., n$ and $k = 1, .., K$.

Table 5.2: Transmission probabilities when $r_t \approx 1/2 \approx r_{t+1}$.

Q_{p_ik} (u)		-1			0			1		
Q_{m_ik} (v)		-1	0	1	-1	0	1	-1	0	1
Q_{ik} (x)	-1	1	1/2	0	1/2	1/4	0	0	0	0
	0	0	1/2	1	1/2	1/2	1/2	1	1/2	0
	1	0	0	0	0	1/4	1/2	0	1/2	1

To illustrate how to calculate these probabilities consider the case where $Q_{p_ik} = Q_{m_ik} = 0$ and $Q_{ik} \in \{-1, 0, 1\}$. From equations (5.5), (5.6), (5.7) and (5.8) the transmission probabilities are

$$Pr\left(Q_{ik}=-1|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},0,0,\mathbf{M_{p}}_ir_k\right)=$$

$$=\frac{Pr\left(S_{p_{ik}}=s^*_{p_{ik}}|S_{p_{iM_{l_k}}},D_{l_k}\right)Pr\left(S_{p_{iM_{r_k}}}=s^*_{p_{iM_{r_k}}}|S_{p_{ik}},D_{r_k}\right)}{\Sigma^1_{s_{p_{ik}}=0}Pr\left(S_{p_{ik}}=s_{p_{ik}}|S_{p_{iM_{l_k}}},D_{l_k}\right)Pr\left(S_{p_{iM_{r_k}}}=s^*_{p_{iM_{r_k}}}|S_{p_{ik}},D_{r_k}\right)}$$
$$\times\frac{Pr\left(S_{m_{ik}}=s^*_{m_{ik}}|S_{m_{iM_{l_k}}},D_{l_k}\right)Pr\left(S_{m_{iM_{r_k}}}=s^*_{m_{iM_{r_k}}}|S_{m_{ik}},D_{r_k}\right)}{\Sigma^1_{s_{m_{ik}}=0}Pr\left(S_{m_{ik}}=s_{m_{ik}}|S_{m_{iM_{l_k}}},D_{l_k}\right)Pr\left(S_{m_{iM_{r_k}}}=s^*_{m_{iM_{r_k}}}|S_{m_{ik}},D_{r_k}\right)}$$
$$\approx\frac{1/2*1/2}{1/2*1/2+1/2*1/2}\,\frac{1/2*1/2}{1/2*1/2+1/2*1/2}=\frac{1}{4}, \tag{5.9}$$

since $Pr\left(S_{p_{ik}} = s^*_{p_{ik}}|S_{p_{iM_{l_k}}}, D_{l_k}\right), Pr\left(S_{p_{iM_{r_k}}} = s^*_{p_{iM_{r_k}}}|s_{p_{ik}}, D_{r_k}\right), Pr\left(S_{m_{ik}} = s^*_{m_{ik}}|S_{m_{iM_{l_k}}}, D_{l_k}\right)$ and $Pr\left(S_{m_{iM_{r_k}}} = s^*_{m_{iM_{r_k}}}|s_{m_{ik}}, D_{r_k}\right) \approx 1/2$ for any value of $s^*_{m_{ik}}$ and $s^*_{m_{iM_{r_k}}}$,

$$
\begin{aligned}
&Pr\Big(Q_{ik}{=}0|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},0,0,\mathbf{M}_{\mathbf{p}_ir_k}\Big){=}\\
&\qquad = \sum_{\{(1,0),(0,1)\}} \frac{Pr\Big(S_{p_{ik}}{=}s^*_{p_{ik}}|S_{p_{iM_{l_k}}},D_{l_k}\Big)Pr\Big(S_{p_{iM_{r_k}}}{=}s^*_{p_{iM_{r_k}}}|S_{p_{ik}},D_{r_k}\Big)}{\sum_{s_{p_{ik}}=0}^{1}Pr\Big(S_{p_{ik}}{=}s_{p_{ik}}|S_{p_{iM_{l_k}}},D_{l_k}\Big)Pr\Big(S_{p_{iM_{r_k}}}{=}s^*_{p_{iM_{r_k}}}|S_{p_{ik}},D_{r_k}\Big)}\\
&\qquad \times \frac{Pr\Big(S_{m_{ik}}{=}s^*_{m_{ik}}|S_{m_{iM_{l_k}}},D_{l_k}\Big)Pr\Big(S_{m_{iM_{r_k}}}{=}s^*_{m_{iM_{r_k}}}|S_{m_{ik}},D_{r_k}\Big)}{\sum_{s_{m_{ik}}=0}^{1}Pr\Big(S_{m_{ik}}{=}s_{m_{ik}}|S_{m_{iM_{l_k}}},D_{l_k}\Big)Pr\Big(S_{m_{iM_{r_k}}}{=}s^*_{m_{iM_{r_k}}}|S_{m_{ik}},D_{r_k}\Big)}\\
&\qquad \approx 2\frac{1/2{*}1/2}{1/2{*}1/2{+}1/2{*}1/2}\,\frac{1/2{*}1/2}{1/2{*}1/2{+}1/2{*}1/2}{=}\frac{1}{2}
\end{aligned}
\tag{5.10}
$$

and $Pr\left(Q_{ik}=1|M_{il_k},M_{ir_k},D_{l_k},D_{r_k},0,0,\mathbf{M}_{\mathbf{p}_ir_k}\right)\approx\frac{1}{4}$. Remaining transmission probabilities are obtained in a similar way.

5.2.2 Missing founder's parents

Founders' parents or grandparents genotypes are usually unknown. We treat them as missing values and estimate them using Bayes' formulas, Mendel's rules and their offspring genotype.

Consider the mother of the i-th individual is unavailable and O_{m_i} represents the set of individuals of her offspring. The genotype of her first marker has probabilities

$$
\begin{aligned}
Pr\Big(M_{m_i1}{=}m_{m_i1}|\mathbf{L}_{m_{O_{m_i}M_1}}\Big) &= \frac{Pr\Big(\mathbf{L}_{m_{O_{m_i}M_1}}{=}\mathbf{l}_{m_{O_{m_i}M_1}}|m_{m_i1}\Big)Pr\Big(M_{m_i1}{=}m_{m_i1}\Big)}{\sum_{m_{m_i1}}Pr\Big(\mathbf{L}_{m_{O_{m_i}M_1}}{=}\mathbf{l}_{m_{O_{m_i}M_1}}|m_{m_i1}\Big)Pr\Big(M_{m_i1}{=}m_{m_i1}\Big)}\\
&=\frac{\prod_{o\in O_{m_i}}Pr\Big(L_{m_{oM_1}}{=}l_{m_{oM_1}}|m_{m_i1}\Big)Pr\Big(M_{m_i1}{=}m_{m_i1}\Big)}{\sum_{m_{m_i1}}\prod_{o\in O_{m_i}}Pr\Big(L_{m_{oM_1}}{=}l_{m_{oM_1}}|m_{m_i1}\Big)Pr\Big(M_{m_i1}{=}m_{m_i1}\Big)},
\end{aligned}
\tag{5.11}
$$

where $m_{m_i1}\in\{-1,0,1\}\equiv\{aa,Aa,AA\}$, $Pr\left(M_{m_i1}=m_{m_i1}\right)$ is the population frequency of the genotype m_{m_i1} that can be assumed as $1/4, 1/2, 1/4$ for $-1, 0$ and 1, respectively, if we have no information about it and $Pr\left(L_{m_{oM_1}}=l_{m_{oM_1}}|m_{m_i1}\right)$ is the Mendelian probability of o-th child has inherited the maternal allele $l_{m_{oM_1}}$ given maternal genotype m_{m_i1}.

Consider also that beyond missing mother's genotypes the i-th individual's maternal grandparents are also missing. The first marker's genotype of grandparents, $M_{p_{m_i}1}$ and $M_{m_{m_i}1}$, has distribution given by (5.11) considering only i-th individual's mother as their offspring (founders are supposed to be unrelated and she has no full-sib neither half-sib in the data set).

For second marker, the paternal and maternal alleles $L_{p_{m_i2}}$ and $L_{m_{m_i2}}$ which comprise M_{m_i2} can be estimated separately. First, for $o\in O_{m_i}$, we determine the value of $S_{m_{o1}}$ based on M_{m_i1}, $M_{p_{m_i}1}$ and $M_{m_{m_i}1}$, and generate $S_{m_{o2}}$ through $Pr\left(S_{m_{o2}}=s_{m_{o2}}|S_{m_{o1}},D_2\right)$, considering $S_{m_{o2}}$ and $S_{m_{o'2}}$ independent for $o\neq o'$ and $o,o'\in O_{m_i}$. Then, we define

$$
L_{p_{m_i2}}=\begin{cases} l_{m_{o2}}, \text{ if } s_{m_{o2}}=1 \text{ for some } o\in O_{m_i},\\ \text{sampled from } Pr\left(L_{p_{m_i2}}=l_{p_{m_i2}}|L_{p_{m_i1}},D_2\right), \text{ if } s_{m_{o2}}=0\ \forall\ o\in O_{m_i},\end{cases}
$$

where $Pr\left(L_{p_{m_i2}}=l_{p_{m_i2}}|L_{p_{m_i1}},D_2\right)$ is estimated using the recombination fraction between paternal allele of markers 2 and 1, and

$$L_{m_{m_i}2} = \begin{cases} l_{m_{o2}}, \text{ if } s_{m_{o2}} = 0 \text{ for some } o \in O_{m_i}, \\ \text{sampled from } Pr\left(L_{m_{m_i}2} = l_{m_{m_i}2} | L_{m_{m_i}1}, D_2\right), \text{ if } s_{m_{o2}} = 1 \ \forall\ o \in O_{m_i}. \end{cases}$$

The second marker's genotype of maternal grandfather $M_{p_{m_i}2} = (L_{p_{p_{m_i}}2}, L_{m_{p_{m_i}}2})$, for instance, is generated from

$$\begin{aligned} Pr\Big(M_{p_{m_i}2}{=}m_{p_{m_i}2}|l_{p_{m_i}2},M_{p_{m_i}1},D_2\Big) &= \frac{Pr\Big(M_{p_{m_i}2}{=}m_{p_{m_i}2},L_{p_{m_i}2}{=}l_{p_{m_i}2}|M_{p_{m_i}1},D_2\Big)}{\sum_{m_{p_{m_i}2}} Pr\Big(M_{p_{m_i}2}{=}m_{p_{m_i}2},L_{p_{m_i}2}{=}l_{p_{m_i}2}|M_{p_{m_i}1},D_2\Big)} \\ &= \frac{Pr\Big(M_{p_{m_i}2}{=}(l_{\cdot_{p_{m_i}}2},l_{p_{m_i}2})|M_{p_{m_i}1},D_2\Big)}{\sum_{l_{\cdot_{p_{m_i}}2}} Pr\Big(M_{p_{m_i}2}{=}(l_{\cdot_{p_{m_i}}2},l_{p_{m_i}2})|M_{p_{m_i}1},D_2\Big)}, \end{aligned} \qquad (5.12)$$

where $\cdot$ represents either the paternal or maternal allele, $Pr\left(M_{p_{m_i}2} = (l_{\cdot_{p_{m_i}}2}, l_{p_{m_i}2})|M_{p_{m_i}1}, D_2\right)$ is the conditional probability of $M_{p_{m_i}2}$ genotype considering that one of its two alleles is $l_{p_{m_i}2}$ (since he transferred this allele for his offspring) and given his previous marker genotype. This conditional probability is defined in the transition matrix

$$P_{M_2} = \begin{array}{c} \\ -1(aa) \\ 0(Aa) \\ 1(AA) \end{array} \begin{array}{c} \begin{array}{ccc} -1(aa) & 0(Aa) & 1(AA) \end{array} \\ \begin{pmatrix} (1-r_2)^2 & 2r_2(1-r_2) & {r_2}^2 \\ r_2(1-r_2) & (1-r_2)^2 + {r_2}^2 & r_2(1-r_2) \\ {r_2}^2 & 2r_2(1-r_2) & (1-r_2)^2 \end{pmatrix} \end{array},$$

and r_2 is the recombination fraction between 2-nd and 1-st markers. The second marker's genotype of maternal grandmother can be estimated similarly.

Genotypes of markers $j = 3, ..., m$ are drawn analogously to the procedure described for $j = 2$. Note that the genotype of a specific marker is generated for all the missing parents and grandparents before the method generates the genotype of the next marker.

Therefore, we simulate the missing genotypes of founder's parents and grandparents in order to obtain the complete likelihood.

5.3 Bayesian approach

We use a Bayesian approach to select and estimate the best model. The *a priori* distributions are

1. $K \sim$ Uniform$(0, 1, ..., m-1)$;

2. $\alpha_k | K \sim$ Normal$(\nu_\alpha, \sigma^2_\alpha)$, $k = 1, ..., K$, where ν_α and $\sigma^2_\alpha > 0$ are known hyper-parameters;

3. $\delta_k | K \sim$ Normal$(\nu_\delta, \sigma^2_\delta)$, $k = 1, ..., K$, where ν_δ and $\sigma^2_\delta > 0$ are known hyper-parameters;

4. $\mu \sim$ Normal(ν_μ, σ^2_μ), where ν_μ and $\sigma^2_\mu > 0$ are known hyper-parameters;

5. $\sigma^2 \sim$ Inverse-gamma(η_a, η_b), where $\eta_a > 0$ and $\eta_b > 0$ are known hyper-parameters and;

6. $\pi(\boldsymbol{\lambda}|K) = \pi(\lambda_1, ..., \lambda_K|K) = \pi(\lambda_1|K)\pi(\lambda_2|\lambda_1, K) \dots \pi(\lambda_K|\lambda_{K-1}, K)$. If there is no *a priori* information about the QTL's location, each location can be assumed uniformly distributed over the possible *loci*.

Combining the likelihood function (5.4) with the *a priori* distributions, we obtain the conditional *a posteriori* distributions,

$$\mu|\left(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\mu}\right) \sim \text{Normal}\left(\frac{\frac{\sum_{i=1}^{n}\left(y_i - \sum_{k=1}^{K}\alpha_k q_{ik} - \sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)}{\sigma^2} + \frac{\nu_\mu}{\sigma_\mu^2}}{\frac{n}{\sigma^2} + \frac{1}{\sigma_\mu^2}}, \frac{1}{\frac{n}{\sigma^2} + \frac{1}{\sigma_\mu^2}}\right), \tag{5.13}$$

$$\sigma^2|\left(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\sigma^2}\right) \sim \text{Inv-gamma}\left(\frac{n}{2} + \eta_a, \frac{\sum_{i=1}^{n}\left(y_i - \sum_{k=1}^{K}\alpha_k q_{ik} - \sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)^2}{2} + \eta_b\right), \tag{5.14}$$

$$\alpha_{k^*}|\left(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\alpha_{k^*}}\right) \sim \text{Normal}\left(\frac{\frac{\sum_{i=1}^{n} q_{ik^*}\left(y_i - \mu - \sum_{k\neq k^*}\alpha_k q_{ik} - \sum_{k=1}^{K}\delta_k(1-|q_{ik}|)\right)}{\sigma^2} + \frac{\nu_\alpha}{\sigma_\alpha^2}}{\frac{\sum_{i=1}^{n} q_{ik^*}^2}{\sigma^2} + \frac{1}{\sigma_\alpha^2}}, \frac{1}{\frac{\sum_{i=1}^{n} q_{ik^*}^2}{\sigma^2} + \frac{1}{\sigma_\alpha^2}}\right), \tag{5.15}$$

$$\delta_{k^*}|\left(\mathbf{y}, \mathbf{q}, \boldsymbol{\theta}_{-\delta_{k^*}}\right) \sim \text{Normal}\left(\frac{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)\left(y_i - \mu - \sum_{k=1}^{K}\alpha_k q_{ik} - \sum_{k\neq k^*}\delta_k(1-|q_{ik}|)\right)}{\sigma^2} + \frac{\nu_\delta}{\sigma_\delta^2}}{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2} + \frac{1}{\sigma_\delta^2}}, \frac{1}{\frac{\sum_{i=1}^{n}(1-|q_{ik^*}|)^2}{\sigma^2} + \frac{1}{\sigma_\delta^2}}\right), \tag{5.16}$$

for $k^* = 1, ..., K$, and

$$Q_{ik}|(\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D}) \sim \text{Multinomial}(1, (p_{ik(-1)}, p_{ik0}, p_{ik1})), \tag{5.17}$$

where $p_{ikj} = \frac{f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k})}{\sum_{j'=-1,0,1} f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j'|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k})}$, for $j \in \{-1, 0, 1\}$ and $i \in F$, or

$$Q_{ik}|(\mathbf{y}, \mathbf{q}_{-q_{ik}}, \mathbf{M}, \mathbf{D}) \sim \text{Multinomial}(1, (p_{ik(-1)}, p_{ik0}, p_{ik1})), \tag{5.18}$$

where $p_{ikj} = \frac{f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j|M_{il_k}, M_{ir_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k})}{\sum_{j'=-1,0,1} f_{Y_i|\mathbf{q}_i}(y_i)Pr(Q_{ik}=j'|M_{ir_k}, M_{il_k}, D_{l_k}, D_{r_k}, \mathbf{Q}_{\mathbf{p}_i k}, \mathbf{M}_{\mathbf{p}_i r_k})}$, for $j \in \{-1, 0, 1\}$ and $i \in F^c$.

The location of k-th QTL, λ_k, is updated jointly with $\mathbf{q}_k$ by Metropolis-Hastings steps in which $\lambda_k^{'}$ is simulated from a Uniform(D_{l_k}, D_{r_k}) distribution and the block $(\lambda_k^{'}, \mathbf{q}_k^{'})$ is accepted with probability $\Psi((\lambda_k^{'}, \mathbf{q}_k^{'})|(\lambda_k, \mathbf{q}_k)) = min(1, A_\lambda)$, where

$$A_\lambda = \frac{exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^{'2}\right\}}{exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^{2}\right\}} \frac{\prod_{i\in F} Pr\left(Q_{ik} = q_{ik}^{'}|\mathbf{M},\mathbf{D}\right)\prod_{i\in F^c} Pr\left(Q_{ik} = q_{ik}^{'}|\mathbf{M},\mathbf{D},\mathbf{Q}_{\mathbf{p}_ik}^{'},\mathbf{M}_{\mathbf{p}_ir_k}\right)}{\prod_{i\in F} Pr\left(Q_{ik} = q_{ik}|\mathbf{M},\mathbf{D}\right)\prod_{i\in F^c} Pr\left(Q_{ik} = q_{ik}|\mathbf{M},\mathbf{D},\mathbf{Q}_{\mathbf{p}_ik},\mathbf{M}_{\mathbf{p}_ir_k}\right)} \\ \times \frac{\prod_{i\in F} Pr\left(Q_{ik} = q_{ik}|\mathbf{y},\mathbf{q}_{-q_{ik}},\mathbf{M},\mathbf{D}\right)\prod_{i\in F^c} Pr\left(Q_{ik} = q_{ik}|\mathbf{y},\mathbf{q}_{-q_{ik}},\mathbf{M},\mathbf{D},\mathbf{Q}_{\mathbf{p}_ik}^{'},\mathbf{M}_{\mathbf{p}_ir_k}\right)}{\prod_{i\in F} Pr\left(Q_{ik} = q_{ik}^{'}|\mathbf{y},\mathbf{q}_{-q_{ik}},\mathbf{M},\mathbf{D}\right)\prod_{i\in F^c} Pr\left(Q_{ik} = q_{ik}^{'}|\mathbf{y},\mathbf{q}_{-q_{ik}},\mathbf{M},\mathbf{D},\mathbf{Q}_{\mathbf{p}_ik}^{'},\mathbf{M}_{\mathbf{p}_ir_k}\right)}, \tag{5.19}$$

$\epsilon_i = y_i - \mu - \sum_{k=1}^{K}\alpha_k q_{ik} - \sum_{k=1}^{K}\delta_k(1 - |q_{ik}|)$ is the residual of the i-th individual, $i = 1, ..., n$, and $\epsilon_i^{'}$ is calculated using $\mathbf{q}_{-\mathbf{q}_k}$ and $\mathbf{q}_k^{'}$.

The number of QTLs K is updated through a birth or a death move. If $K = 1$, a birth is chosen. If $K = K_{max}$, a death is chosen. If $1 < K < K_{max}$, a new candidate value is chosen between $K-1$ and $K+1$ with probability 0.5. When a birth movement is chosen, the association between markers genotype and residuals of the current model (measured by the Kruskal-Wallis test statistic) is used to define a *locus* for the new QTL. The new QTL's genotype, additive and dominance effects are drawn from their conditional *a posteriori* distributions defined by (5.17) or (5.18), (5.15) and (5.16), and μ and σ^2 are updated by (5.13) and (5.14), respectively. When a death movement is chosen, we select a QTL to be excluded from the current model. Its genotype and effects are excluded from the model and μ and σ^2 are updated using (5.13) and (5.14), respectively. For more details about updating K see Zuanetti & Milan (2016).

Let $x = (\mathbf{q}, \boldsymbol{\theta})$ be the current state with K QTLs and $x^{'} = (\mathbf{q}^{'}, \boldsymbol{\theta}')$ be the proposed movement, where signal $'$ represents either a birth (b) or a death (d) of a QTL. The proposed move from x to $x^{'}$ is accepted according to the Metropolis-Hastings probability $\Psi(x^{'}|x) = min(1, A^{'})$ where

$$A^{'} = \frac{L(\boldsymbol{\theta}'|\mathbf{y},\mathbf{q}^{'})}{L(\boldsymbol{\theta}|\mathbf{y},\mathbf{q})}\frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})}\frac{q(x|x^{'})}{q(x^{'}|x)}, \tag{5.20}$$

and $q(\cdot|\cdot)$ is the transition function.

5.3.1 Algorithm DDRJ for pedigree data

The data-driven reversible jump is specified as follows:

1. Initialize a configuration for $\boldsymbol{\theta}$ and $\mathbf{q}$.

2. For l-th iteration, $l = 1, ..., L$, do:

   (a) Simulate markers's genotype for missing founders' parents and grandparents as described in Section 5.2.2.

   (b) Simulate values of $\mathbf{s}_{p_i}$ and $\mathbf{s}_{m_i}$ for all nonfounder individuals.

(c) Choose between a death or birth movement.

(d) Generate candidate-values $x^{'}$ depending on chosen movement as described in Section 5.3.

(e) Accept the proposal with probability $\Psi(x^{'}|x) = min(1, A^{'})$, where $A^{'}$ is given by (5.20) and $'$ is either b or d.

   i. If a birth movement is accepted, do $K^{(l)} = K^{(l-1)} + 1$ and consider x^{b}.

   ii. If a death movement is accepted, do $K^{(l)} = K^{(l-1)} - 1$ and consider x^{d}.

   iii. If no movement is accepted, do $K^{(l)} = K^{(l-1)}$ and replicate x.

(f) Update λ_k, for $k = 1, ..., K^{(l)}$, using (5.19).

(g) Update q_{ik}, for $i = 1, ..., n$ and $k = 1, ..., K^{(l)}$, from their conditional *a posteriori* distribution defined by (5.17) or (5.18).

(h) Update α_k and δ_k, for $k = 1, ..., K^{(l)}$, from their conditional *a posteriori* distributions defined by (5.15) and (5.16), respectively.

(i) Update μ from its conditional *a posteriori* distribution defined by (5.13).

(j) Update σ^2 from its conditional *a posteriori* distribution defined by (5.14).

This algorithm is implemented in R language and the codes are available in the appendices of this chapter. R is a free software environment for statistical computing and graphics and more details are found in its homepage `https://www.r-project.org`'.

5.4 Applications

We apply the proposed method to simulated data sets and compare its performance with the variance components analysis performance. The test data sets were generated from program SimPed (Leal *et al*, 2005) available in `http://bioinformatics.org/simped/doc.html`. We simulate the genotype of 200 marker *loci* equally distributed in a 2 Morgans (M) chromosome (1 marker every 1 cM). Our data consists of $n = 500$ individuals from 50 families structured as in Figure 5.1. We interchange groups of diallelic markers in linkage disequilibrium and blocks of diallelic markers in linkage equilibrium.

We choose $K = 5$ markers to randomly locate the QTLs and simulate phenotypes as the sum of the overall mean, additive and dominance QTLs' effects and a random residual drawn from the Normal$(0, \sigma^2)$ distribution, for $\sigma^2 = 1, 4, 9, 16, 25$, as in Lund *et al.* (2009). The additive and dominance effects were drawn from the Normal$(0, 1)$ and Normal$(0, 0.25)$ distributions, respectively. The true values used to simulate the phenotypes are: $\mu = 5$, $\boldsymbol{\lambda} = (0.12, 0.52, 0.62, 0.92, 1.10)$, $\boldsymbol{\alpha} = (0.759, -0.529, -0.784, 0.679, 1.896)$ and $\boldsymbol{\delta} = (0.005, -0.238, -0.523, -0.278, 0.085)$. The 5-th and 2-nd QTLs have the highest and lowest additive effect, respectively. The dominance effects of the 1-st and 5-th QTLs are near zero and they mimic QTLs which have only additive effects.

Heritability reflects all the genetic contributions to population's phenotypic variance and is defined as $H^2 = \frac{\text{total variance}-\sigma^2}{\text{total variance}}$. For our simulated data sets the heritability are approximately 0.75, 0.40, 0.20, 0.10, 0.05 for $\sigma^2 = 1, 4, 9, 16, 25$, respectively. Lower heritability usually means harder modeling.

In order to get a vague *a priori* distribution we set hyperparameters as $\nu_\alpha = \nu_\delta = \nu_\mu = 0$, $\sigma^2_\alpha = \sigma^2_\delta = \sigma^2_\mu = 100$ and $\eta_a = \eta_b = 0.1$. We run DDRJ for $L = 11000$ iterations and discard the 1000 first iterations. We take one draw for every 10 iterations to obtain a final sequence of 1000 values. The chains are initialized with $K = 0$. The analysis of convergence indicates that the number of iterations is enough for reliable results.

Table 5.3 shows the *a posteriori* probabilities for K. It is calculated as the relative frequency of each value of K in the MCMC sequence.

Table 5.3: The *a posteriori* probability for K.

K	σ				
	1	2	3	4	5
0	0.0	0.0	0.0	0.0	5.0
1	0.0	0.2	17.1	44.2	57.3
2	0.0	13.5	69.6	44.4	28.9
3	0.0	41.0	12.0	10.6	7.3
4	0.0	34.4	1.1	0.7	1.4
5	99.7	9.9	0.2	0.1	0.1
≥ 6	0.3	1.0	0.0	0.0	0.0

The proposed methodology shows good performance in estimating $K = 5$ when $\sigma = 1$. When $\sigma = 2$, although the estimate of number of QTLs is $K = 3$, we notice in Figure 5.3 that the first estimated QTL is clearly located in three different regions: around 0.10, 0.50 and 0.60 M where true QTLs 1, 2 and 3 are located. Figure 5.3 indicates that the three distinct regions should be studied in details to verify QTL's presence. As the error variability rises and the heritability goes down the method identifies only QTLs with higher effect in phenotype.

Table 5.4 shows the Bayesian estimates (*a posteriori* average) for the model's parameters. It also provides the 95% credibility interval for each parameter. When $\sigma = 2$, we consider the three distinct regions shown in Figure 5.3 as different QTLs in order to show separated estimates for each of them. The dominance effect of 1-th and 5-th QTLs are nonsignificant since their true value is very close to zero. When $\sigma \leq 2$ the QTLs' effects and remaining parameters are well estimated. The 4-th and 5-th QTLs are identified when $\sigma = 3$ and only the 5-th QTLs is estimated when $\sigma = 4$. Even when only some QTLs are identified, their effects, the error variability and the overall mean are well estimated.

We apply the variance components modeling to the simulated data sets in order to compare

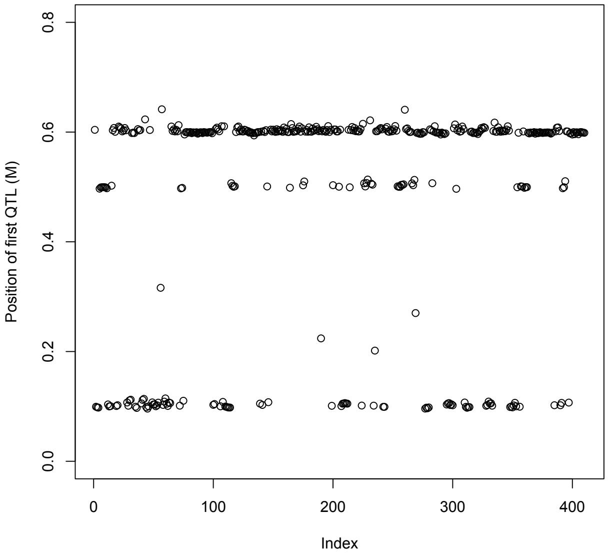


Figure 5.3: Position (M) of the first estimated QTL when $\sigma = 2$.

with the proposed method. In a variance components model the effects of QTLs are defined as random effects. The variance structure of $\mathbf{y}$ is function of σ_α^2 (total additive genetic variance), σ_δ^2 (total dominance genetic variance) and σ^2 (variance of error term or environmental variance). Heritability in these models is defined as $H^2 = \frac{\sigma_\alpha^2+\sigma_\delta^2}{\sigma_\alpha^2+\sigma_\delta^2+\sigma^2}$.

We use software Mendel, available at `http://www.genetics.ucla.edu/software/mendel`, to do the variance component analysis, see Bauman *et al.* (2005); Lange & Sobel (2006); Lange *et al.* (1976, 1983, 2013) for details. We analyze the data with additive and dominance random effects in the model using variance components option of analysis.

Table 5.5 shows the estimates of variance components models. We note for cases considered here that the dominance effect is nonsignificant in all cases and the error variability is always underestimated. Consequently, additive variance and heritability are overestimated and we wrongly justify part of environmental variability as genetic contribution to phenotypes.

Table 5.5: Estimates and standard errors of variance components models.

Parameter	σ				
	1	2	3	4	5
μ	4.88 (0.13)	4.82 (0.15)	4.80 (0.18)	4.78 (0.22)	4.78 (0.25)
σ_α^2	3.34 (0.46)	3.50 (0.75)	3.96 (1.18)	4.76 (1.77)	5.87 (2.51)
σ_δ^2	0.02 (1.28)	0.00	0.00	0.00	0.00
σ^2	0.69 (1.34)	3.26 (0.56)	7.39 (1.05)	13.03 (1.70)	20.20 (2.51)
H^2	0.83 (0.19)	0.52 (0.06)	0.35 (0.07)	0.27 (0.07)	0.22 (0.08)

Table 5.4: The *a posteriori* estimates for models.

Parameter	True value	σ				
		1	2	3	4	5
λ_1	0.12	0.10 (0.09,0.11)	0.10 (0.09,0.11)			
λ_2	0.52	0.50 (0.49,0.51)	0.50 (0.49,0.52)			
λ_3	0.62	0.60 (0.59,0.61)	0.61 (0.60,0.63)			
λ_4	0.92	0.90 (0.89,0.91)	0.90 (0.89,0.91)	0.90 (0.89,0.91)	0.90 (0.50,0.98)	
λ_5	1.10	1.08 (1.07,1.09)	1.08 (1.07,1.09)	1.08 (1.07,1.09)	1.08 (1.07,1.09)	1.07 (1.06,1.08)
μ	5	4.95 (4.70,5.18)	5.00 (4.45,5.48)	5.10 (4.54,5.63)	5.18 (4.00,5.97)	4.61 (3.90,5.95)
α_1	0.76	0.67 (0.53,0.82)	0.67 (0.42,0.92)			
α_2	-0.53	-0.59 (-0.73,-0.45)	-0.57 (-0.87,-0.36)			
α_3	-0.78	-0.71 (-0.85,-0.59)	-0.62 (-0.94,-0.29)			
α_4	0.68	0.77 (0.63,0.90)	0.84 (0.57,1.14)	0.90 (0.43,1.34)	0.95 (0.03,1.58)	
α_5	1.90	1.75 (1.62,1.90)	1.60 (1.32,1.87)	1.52 (1.15,1.86)	1.33 (0.73,1.83)	1.35 (0.11,1.95)
δ_1	0.01	-0.04 (-0.24,0.13)	-0.08 (-0.37,0.29)			
δ_2	-0.24	-0.01 (-0.16,0.20)	-0.36 (-0.69,0.05)			
δ_3	-0.52	-0.39 (-0.59,-0.21)	-0.25 (-0.70,0.24)			
δ_4	-0.28	-0.55 (-0.75,-0.34)	-0.69 (-1.11,-0.27)	-0.92 (-1.58,-0.27)	-0.98 (-1.77,0.35)	
δ_5	0.08	0.12 (-0.07,0.30)	0.09 (-0.29,0.51)	0.05 (-0.47,0.55)	0.03 (-0.55,0.67)	0.06 (-2.00,1.09)
σ		0.95 (0.89,1.02)	1.97 (1.85,2.09)	2.89 (2.71,3.09)	3.81 (3.57,4.10)	4.82 (4.53,5.14)

5.5 Discussion

We propose a QTL mapping model which describes the pedigree relationship between individuals and their genotypes through conditional independence structures without random effects in the model. Random effects may increase model's variability making more difficult the identification of QTLs with small effect.

The method shows good performance under the tested conditions. The number of QTLs, their position and effects on the phenotype are well estimated outperforming the variance components method for QTL mapping of pedigree data. The variance components modeling confounds genetic and environmental sources of variation. The proposed method also provides credibility interval about the uncertainty of estimates while conventional methods usually provide point estimates or asymptotic confidence intervals for large samples.

The major differential of the proposed method is that it considers Mendelian **and** multiple linked *loci* segregation to estimate the genotype's probability of a specific *locus*. Also, it estimates missing parents genotype and it runs on reasonable computation times (partially thanks to the DDRJ procedure).

5.6 Appendices

5.6.1 R codes to carry out DDRJ

In this section, we show DDRJ R codes.

```
#
####################################
# Functions
####################################
```

```
#
################
# calculation of Crámeer’ coefficient
#
cv.test = function(x,y) {
  CV = sqrt(chisq.test(x, y, correct=FALSE)$statistic /
    (length(x) * (min(length(unique(x)),length(unique(y))) - 1)))
  print.noquote("CramÈr V / Phi:")
  return(as.numeric(CV))}
#
### identify if an allele is paternal or maternal
#
identalel_pai_mae<-function(genpai,genmae,genfilho){
 # Alelo = 2 representa alelo dominante e Alelo = 1 representa alelo recessivo
 if (genfilho==1) {
  afp<-2
  afm<-2} else {
  if (genfilho==-1){
   afp<-1
   afm<-1} else {
   if (genpai==1) {
    afp<-2
    afm<-1} else {
    if (genpai==-1) {
     afp<-1
     afm<-2} else {
     if (genmae==1) {
      afp<-1
      afm<-2} else {
      if (genmae==-1) {
       afp<-2
       afm<-1} else {
       aux<-rbinom(1,1,0.5)
       afp<-(2^aux)*(1^(1-aux))
       afm<-(2^(1-aux))*(1^aux)}}}}}}
 list(afp,afm)}
#
#################
```

```
### sample a birth QTL candidate and compute its transition function
#
gera.inclusao.QTL<-function(dados,residuos,mat.delinea,vet.coef,pos.qtls,
loc.marc,sigma2.vig,alpha.vig,delta.vig,media.mi,sigma2.mi,media.alpha,
sigma2.alpha,media.delta,sigma2.delta,neta.a,neta.b,mat.ped.pais,founders,
nonfounders,alelo.paterno,alelo.materno,Sp,Sm,mat.alelo.qtls,comfenot){
 #
 marcador<-prob.selec.marc(dados[comfenot,],residuos,pos.qtls,loc.marc)
 qtl<-pos.qtl(pos.qtls,loc.marc,marcador[[1]],marcador[[3]])
 #
 dQTL<-qtl[[1]]
 Marc1<-sum(loc.marc<=dQTL)
 Marc2<-length(loc.marc)-(sum(loc.marc>=dQTL)-1)
 dM1<-loc.marc[Marc1]
 dM2<-loc.marc[Marc2]
 r12<-Haldane(abs(dM2-dM1))
 r1<-Haldane(abs(dQTL-dM1))
 r2<-Haldane(abs(dQTL-dM2))
 matrec1<-matrix(c(1-r1,r1,r1,1-r1),2,2,byrow=TRUE)
 matrec2<-matrix(c(1-r2,r2,r2,1-r2),2,2,byrow=TRUE)
 #
 ## sample the genotype of founders' QTL
 #
 matriz.prob.gen.QTL<-numeric()
 probQTL<-numeric(3)
 for (j in 1:length(founders)){
  gen<-c(-1,0,1)
  for (i in 1:3) probQTL[i]<-(calc.prob.gen(dados[j,(Marc1+1)],gen[i],r1)*
calc.prob.gen(gen[i],dados[j,(Marc2+1)],r2))/calc.prob.gen(
dados[j,(Marc1+1)],dados[j,(Marc2+1)],r12)
  matriz.prob.gen.QTL<-rbind(matriz.prob.gen.QTL,probQTL)}
 #
 dados.QTL<-matrix(0,length(founders),1)
 for (i in 1:length(founders)) dados.QTL[i,1]<-gen[rDiscreta(
matriz.prob.gen.QTL[i,])]
 #
 ale.pai<-matrix(99,nrow=length(founders),1)
 ale.mae<-matrix(99,nrow=length(founders),1)
 probale<-numeric(2)
 #
 for (k in 1:length(founders)){
  if (dados.QTL[k,1]==-1){
   ale.mae[k,1]<-1
   ale.pai[k,1]<-1}
  if (dados.QTL[k,1]==1){
   ale.mae[k,1]<-2
   ale.pai[k,1]<-2}
  if (dados.QTL[k,1]==0){
   for (ale in 1:2) probale[ale]<-matrec1[alelo.paterno[k,Marc1],ale]*
matrec2[ale,alelo.paterno[k,Marc2]]
   probale<-probale/sum(probale)
   aux2<-rDiscreta(probale)-1
   ale.mae[k,1]<-1^(aux2)*2^(1-aux2)
   ale.pai[k,1]<-2^(aux2)*1^(1-aux2)}}
 dados.alelo<-cbind(ale.pai,ale.mae)
 #
 ## sample the genotype of nonfounders' QTL
 #
  matriz.prob.S.pat<-numeric()
 matriz.prob.S.mat<-numeric()
 probSpat<-numeric(2)
 probSmat<-numeric(2)
 for (j in 1:length(nonfounders)){
  for (i in 1:2){
   probSmat[i]<-matrec1[Sm[j,Marc1],i]*matrec2[i,Sm[j,Marc2]]
   probSpat[i]<-matrec1[Sp[j,Marc1],i]*matrec2[i,Sp[j,Marc2]]}
  matriz.prob.S.mat<-rbind(matriz.prob.S.mat,(probSmat/sum(probSmat)))
  matriz.prob.S.pat<-rbind(matriz.prob.S.pat,(probSpat/sum(probSpat)))}
 Smqtl<-numeric(length(nonfounders))
 Spqtl<-numeric(length(nonfounders))
 for (j in 1:length(nonfounders)){
  Smqtl[j]<-rDiscreta(matriz.prob.S.mat[j,])
  Spqtl[j]<-rDiscreta(matriz.prob.S.pat[j,])}
 for (i in 1:length(nonfounders)) dados.alelo<-rbind(dados.alelo,c(dados.alelo
[mat.ped.pais[i,1],Spqtl[i]],dados.alelo[mat.ped.pais[i,2],Smqtl[i]]))
 #
 dados.QTL<-matrix(0,nrow(dados.alelo),1)
 for (i in 1:nrow(dados.alelo)){
  if (dados.alelo[i,1]==1 & dados.alelo[i,2]==1) dados.QTL[i,1]<--1
  if (dados.alelo[i,1]==2 & dados.alelo[i,2]==2) dados.QTL[i,1]<-1
  if (dados.alelo[i,1]!= dados.alelo[i,2]) dados.QTL[i,1]<-0}
 mat.delinea<-cbind(mat.delinea,dados.QTL)
 #
 alpha<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,residuos,
alpha.vig,dados.QTL[comfenot,1])
 vet.coef<-rbind(vet.coef,alpha[[1]])
 predito<-mat.delinea[comfenot,]%*%vet.coef
 residuos<-dados[comfenot,1]-predito
 #
 dados.QTL<-cbind(dados.QTL,(1-abs(dados.QTL[,1])))
 delta<-poster.delta(media.delta,sigma2.delta,sigma2.vig,residuos,
delta.vig,dados.QTL[comfenot,2])
 mat.delinea<-cbind(mat.delinea,dados.QTL[,2])
 vet.coef<-rbind(vet.coef,delta[[1]])
 predito<-mat.delinea[comfenot,]%*%vet.coef
 residuos<-dados[comfenot,1]-predito
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef[1,1])
 vet.coef[1,1]<-mi[[1]]
 predito<-mat.delinea[comfenot,]%*%vet.coef
 residuos<-dados[comfenot,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 #
 list(qtl[[1]],mat.delinea,vet.coef,sigma2[[1]],marcador[[2]],qtl[[2]],
 dados.alelo,alpha[[2]],delta[[2]],mi[[2]],sigma2[[2]],c(pos.qtls,
 qtl[[1]]),cbind(mat.alelo.qtls,dados.alelo))}
#
#################
### sample a death QTL candidate and compute its transition function
#
gera.exclusao.QTL<-function(pos.qtls,vet.coef,mat.delinea,sigma2.vig,
media.mi,sigma2.mi,neta.a,neta.b,mat.alelo.qtls,comfenot){
 num.QTLs<-length(pos.qtls)
 qtl<-prob.excl.marc(num.QTLs,vet.coef)
 #
 pos.qtls<-pos.qtls[-qtl[[1]]]
 mat.alelo.qtls<-mat.alelo.qtls[,-c((2*qtl[[1]])-1,(2*qtl[[1]]))]
 mat.delinea<-matrix(mat.delinea[,-c((2*qtl[[1]]),(2*qtl[[1]]+1))],
nrow=nrow(dados))
 vet.coef<-matrix(vet.coef[-c((2*qtl[[1]]),(2*qtl[[1]]+1))],ncol=1)
 predito<-mat.delinea[comfenot,]%*%vet.coef
 residuos<-dados[comfenot,1]-predito
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef[1,1])
 vet.coef[1,1]<-mi[[1]]
 predito<-mat.delinea[comfenot,]%*%vet.coef
 residuos<-dados[comfenot,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 #
 list(qtl[[1]],mat.delinea,vet.coef,sigma2[[1]],qtl[[2]],mi[[2]],sigma2[[2]],
sigma2[[2]],sigma2[[2]],sigma2[[2]],sigma2[[2]],pos.qtls,mat.alelo.qtls)}
#
#################
### sample a merge QTL candidate and compute its transition function
#
gera.juncao.QTL<-function(dados,num.QTLs,mat.delinea,vet.coef,
pos.qtls,media.alpha,sigma2.alpha,sigma2.vig,media.delta,sigma2.delta,
media.mi,sigma2.mi,neta.a,neta.b,mat.alelo.qtls,comfenot){
 cramer<-numeric(num.QTLs-1)
 for (i in 1:(num.QTLs-1)) cramer[i]<-abs(cv.test(mat.delinea
[comfenot,2*i],mat.delinea[comfenot,2*(i+1)])) # cramer entre os QTLs vizinhos
 prob_par<-cramer/sum(cramer)
 junta<-rDiscreta(prob_par)
 par_QTL<-c(junta,junta+1)
 efeitos<-c(1/sum(abs(vet.coef[2*junta,1]),abs(vet.coef[2*junta+1,1])),
1/sum(abs(vet.coef[2*(junta+1),1]),abs(vet.coef[2*(junta+1)+1,1])))
 prob<-efeitos/sum(efeitos)
 gera<-rDiscreta(prob) # escolhe o QTL que vai sumir
 qtl<-par_QTL[gera]
 #
 pos.qtls.c<-pos.qtls[-qtl]
 mat.alelo.qtls.c<-mat.alelo.qtls[,-c((2*qtl)-1,(2*qtl))]
 mat.delinea.c<-matrix(mat.delinea[,-c((2*qtl),(2*qtl+1))],nrow=nrow(dados))
 vet.coef.c<-matrix(vet.coef[-c((2*qtl),(2*qtl+1))],ncol=1)
 predito<-mat.delinea.c[comfenot,]%*%vet.coef.c
 residuos<-dados[comfenot,1]-predito
 #
 alpha<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,residuos,
vet.coef.c[(2*par_QTL[[1]]),1],mat.delinea.c[comfenot,(2*par_QTL[[1]])])
 vet.coef.c[(2*par_QTL[[1]]),1]<-alpha[[1]]
 residuos<-dados[comfenot,1]-(mat.delinea.c[comfenot,]%*%vet.coef.c)
 delta<-poster.delta(media.delta,sigma2.delta,sigma2.vig,residuos,
vet.coef.c[(2*par_QTL[[1]])+1,1],mat.delinea.c[comfenot,(2*par_QTL[[1]])+1])
 vet.coef.c[(2*par_QTL[[1]])+1,1]<-delta[[1]]
 residuos<-dados[comfenot,1]-(mat.delinea.c[comfenot,]%*%vet.coef.c)
 #
 mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,vet.coef.c[1,1])
 vet.coef.c[1,1]<-mi[[1]]
 predito<-mat.delinea.c[comfenot,]%*%vet.coef.c
 residuos<-dados[comfenot,1]-predito
 sigma2<-poster.sigma2(neta.a,neta.b,residuos)
 prob_merge<-log(prob[gera])+log(prob_par[junta])
 #
 list(qtl,mat.delinea.c,vet.coef.c,sigma2[[1]],prob_merge,mi[[2]],sigma2[[2]],
alpha[[2]],delta[[2]],par_QTL[which(par_QTL!=qtl)],par_QTL,pos.qtls.c,
mat.alelo.qtls.c)}
#
##################
# run DDRJ procedure and initialize the chain with k=0 QTLs
##################
#
####### hyper parameters of a priori distributions
#
neta.a<-0.1
neta.b<-0.1
sigma2.mi<-100
media.mi<-0
sigma2.alpha<-100
media.alpha<-0
sigma2.delta<-100
```

```
media.delta<-0
#
####### Model initialization
#
set.seed(510)
residuos<-dados[comfenot,1]
sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
#
mi<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,0)[[1]]
#
pos.qtls<-numeric()
num.QTLs<-length(pos.qtls)
mat.alelo.qtls<-numeric()
mat.delinea<-matrix(1,nrow(dados),1)
vet.coef<-matrix(mi,1,1)
predito<-mat.delinea[comfenot,]%*%vet.coef
residuos<-dados[comfenot,1]-predito
sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
#
amostrasfin<-1000
burnin<-1000
saltos<-10
AmostrasTotal<-burnin+amostrasfin*saltos
#
indSpMgtotal<-numeric(AmostrasTotal)
probacetotal<-numeric(AmostrasTotal)
probacemerge<-numeric(AmostrasTotal)
num.QTL.total<-numeric(AmostrasTotal)
indrejtotal<-numeric(AmostrasTotal)
#
#####################
#  run DDRJ - some functions used here are available
#  in R codes of Chapter 3
#####################
#
library(compiler)
enableJIT(3)
#
```

```
for (int in (1:AmostrasTotal)){
  cat('\n', crom, int,num.QTLs)
  cand.sp.mg<-dec.sp.mg(num.QTLs,num.marc)
  indSpMgtotal[int]<-cand.sp.mg[[1]]
  #
  ##############
  ###### QTL birth
  ##############
  #
  if (indSpMgtotal[int]==1) {candidato<-gera.inclusao.QTL(dados,residuos,
mat.delinea,vet.coef,pos.qtls,loc.marc,sigma2.vig,0,0,media.mi,sigma2.mi,
media.alpha,sigma2.alpha,media.delta,sigma2.delta,neta.a,neta.b,
mat.ped.pais,founders,nonfounders,alelo.paterno,alelo.materno,Sp,Sm,
mat.alelo.qtls,comfenot)
   num.QTL.total[int]<-num.QTLs+1
   residuossp<-dados[comfenot,1]-(candidato[[2]][comfenot,]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2sp<-candidato[[4]]
   mi<-vet.coef[1,1]
   misp<-candidato[[3]][1,1]
   alphasp<-candidato[[3]][(nrow(candidato[[3]])-1),1]
   deltasp<-candidato[[3]][(nrow(candidato[[3]])),1]
   num.QTLssp<-num.QTLs+1
   psplit<-cand.sp.mg[[2]][1]
   pmerge<-dec.sp.mg(num.QTLssp,num.marc)[[2]][2]
   pmarc<-candidato[[5]]
   plambdasp<-candidato[[6]]
  #
   efeito<-numeric()
   if (num.QTLs==0) efeito<-0
   if (num.QTLs>0) for (i in 1:num.QTLs) efeito[i]<-1/(abs(vet.coef[2*i,1])+
abs(vet.coef[(2*i)+1,1]))
   efeitosp<-1/(abs(alphasp)+abs(deltasp))
   plambdamg<-log(efeitosp/(efeitosp+sum(efeito)))
  #
   res.mi<-residuos+mi
   quo<-(length(residuos)/sigma2sp)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2sp)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
  #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
  #
   postalphasp<-candidato[[8]]
   postdeltasp<-candidato[[9]]
   postmisp<-candidato[[10]]
   postsigma2sp<-candidato[[11]]
   pos.qtls.sp<-sort(candidato[[12]])
   pri.pos.sp<-dens.priori.loc.qtls(pos.qtls.sp,loc.marc)
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
  #
   probace<-prob.aceitacao(residuossp,residuos,sigma2,sigma2sp,misp,mi,
media.mi,sigma2.mi,neta.a,neta.b,alphasp,media.alpha,sigma2.alpha,deltasp,
media.delta,sigma2.delta,loc.marc,num.QTLs,num.QTLssp,psplit,pmerge,pmarc,
plambdasp,plambdamg,postmi,postsigma2,postalphasp,postdeltasp,postmisp,
postsigma2sp,pri.pos.sp,pri.pos)}
  #
  ##############
  ###### QTL death
  ##############
  #
  if (indSpMgtotal[int]==2) {candidato<-gera.exclusao.QTL(pos.qtls,vet.coef,
mat.delinea,sigma2.vig,media.mi,sigma2.mi,neta.a,neta.b,mat.alelo.qtls,comfenot)
   num.QTL.total[int]<-num.QTLs-1
   residuosmg<-dados[comfenot,1]-(candidato[[2]][comfenot,]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2mg<-candidato[[4]]
   mi<-vet.coef[1,1]
   mimg<-candidato[[3]][1,1]
   num.QTLsmg<-num.QTLs-1
   pmerge<-cand.sp.mg[[2]][2]
   psplit<-dec.sp.mg(num.QTLsmg,num.marc)[[2]][1]
   postmimg<-candidato[[6]]
   postsigma2mg<-candidato[[7]]
   pos.qtls.mg<-sort(candidato[[12]])
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
   pri.pos.mg<-dens.priori.loc.qtls(pos.qtls.mg,loc.marc)
   plambdamg<-candidato[[5]]
   alpha<-vet.coef[(2*candidato[[1]]),1]
   delta<-vet.coef[(2*candidato[[1]]+1),1]
  #
   krusk<-prob.selec.marc(dados[comfenot,],residuosmg,pos.qtls.mg,loc.marc)[[3]]
   prob<-krusk/sum(krusk)
   QTLmg<-pos.qtls[candidato[[1]]]
   Marc1<-sum(loc.marc<=QTLmg)
   Marc2<-num.marc-sum(loc.marc>=QTLmg)+1
   loc.Marc1<-dens.pos.qtl(QTLmg,pos.qtls.mg,loc.marc,Marc1,krusk)
   loc.Marc2<-dens.pos.qtl(QTLmg,pos.qtls.mg,loc.marc,Marc2,krusk)
   plambdasp<-log(prob[Marc1]*loc.Marc1+prob[Marc2]*loc.Marc2)
   pmarc<-0
  #
   quo<-(sum(mat.delinea[comfenot,2*candidato[[1]]]^2)/sigma2mg)+
(1/sigma2.alpha)
   media<-((sum(mat.delinea[comfenot,2*candidato[[1]]]*residuosmg)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha<-dnorm(alpha,media,sqrt(variancia),log = TRUE)
  #
   quo<-(sum(mat.delinea[comfenot,2*candidato[[1]]+1]^2)/sigma2mg)+
(1/sigma2.delta)
   media<-((sum(mat.delinea[comfenot,2*candidato[[1]]+1]*residuosmg)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta<-dnorm(delta,media,sqrt(variancia),log = TRUE)
  #
   res.mi<-residuos+vet.coef[1,1]
   quo<-(length(residuos)/sigma2mg)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2mg)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
  #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
  #
   probace<-1/prob.aceitacao(residuos,residuosmg,sigma2mg,sigma2,mi,mimg,
media.mi,sigma2.mi,neta.a,neta.b,alpha,media.alpha,sigma2.alpha,delta,
media.delta,sigma2.delta,loc.marc,num.QTLsmg,num.QTLs,psplit,pmerge,pmarc,
plambdasp,plambdamg,postmimg,postsigma2mg,postalpha,postdelta,postmi,
postsigma2,pri.pos,pri.pos.mg)}
  #
  #####################
  #  update the model parameters
  #####################
  #
  # update the number of QTLs (accept or reject the birth or death candidate)
  #
  probacetotal[int]<-probace
  aux2<-runif(1)
  if (aux2<probace){
   indrejtotal[int]<-0
   pos.qtls<-candidato[[12]]
   num.QTLs<-length(pos.qtls)
   mat.delinea<-candidato[[2]]
   vet.coef<-candidato[[3]]
   sigma2.vig<-candidato[[4]]
   mat.alelo.qtls<-candidato[[13]]
   if (num.QTLs>0){
    posicao<-order(pos.qtls)
    pos.qtls<-pos.qtls[posicao]
    posicao2<-1
    for (i in 1:length(posicao)) posicao2<-c(posicao2,posicao[i]*2,posicao[i]*2+1)
    mat.alelo.qtls<-mat.alelo.qtls[,posicao2[-1]-1]
    mat.delinea<-mat.delinea[,posicao2]
    vet.coef<-matrix(vet.coef[posicao2,],ncol(mat.delinea),1)}
   residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)}
  #
  if (aux2>=probace) indrejtotal[int]<-1
  #
  #### evaluate a merge move of two consecutive QTLs
  #
  if (num.QTLs>1){
   candidato<-gera.juncao.QTL(dados,num.QTLs,mat.delinea,
vet.coef,pos.qtls,media.alpha,sigma2.alpha,sigma2.vig,media.delta,sigma2.delta,
media.mi,sigma2.mi,neta.a,neta.b,mat.alelo.qtls,comfenot)
   residuosmg<-dados[comfenot,1]-(candidato[[2]][comfenot,]%*%candidato[[3]])
   sigma2<-sigma2.vig
   sigma2mg<-candidato[[4]]
   mi<-vet.coef[1,1]
   mimg<-candidato[[3]][1,1]
   num.QTLsmg<-num.QTLs-1
   pmerge<-0
   psplit<-0
   postmimg<-candidato[[6]]
   postsigma2mg<-candidato[[7]]
   postalphamg<-candidato[[8]]
   postdeltamg<-candidato[[9]]
   pos.qtls.mg<-sort(candidato[[12]])
   pri.pos<-dens.priori.loc.qtls(pos.qtls,loc.marc)
   pri.pos.mg<-dens.priori.loc.qtls(pos.qtls.mg,loc.marc)
   plambdamg<-candidato[[5]]
   alphamg<-candidato[[3]][(2*candidato[[11]][1]),1]
   deltamg<-candidato[[3]][(2*candidato[[11]][1]+1),1]
   alphasp1<-vet.coef[(2*candidato[[1]]),1]
   deltasp1<-vet.coef[(2*candidato[[1]]+1),1]
   alphasp2<-vet.coef[(2*candidato[[10]]),1]
   deltasp2<-vet.coef[(2*candidato[[10]]+1),1]
   #
   krusk<-prob.selec.marc(dados[comfenot,],residuosmg,pos.qtls.mg,loc.marc)[[3]]
   prob<-krusk/sum(krusk)
   QTLmg<-pos.qtls[candidato[[1]]]
   Marc1<-sum(loc.marc<=QTLmg)
   Marc2<-num.marc-sum(loc.marc>=QTLmg)+1
   loc.Marc1<-dens.pos.qtl(QTLmg,pos.qtls.mg,loc.marc,Marc1,krusk)
   loc.Marc2<-dens.pos.qtl(QTLmg,pos.qtls.mg,loc.marc,Marc2,krusk)
   plambdasp<-log(prob[Marc1]*loc.Marc1+prob[Marc2]*loc.Marc2)
   pmarc<-0
   #
   quo<-(sum(mat.delinea[comfenot,2*candidato[[1]]]^2)/sigma2mg)+
 (1/sigma2.alpha)
   media<-((sum(mat.delinea[comfenot,2*candidato[[1]]]*residuosmg)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha1<-dnorm(alphasp1,media,sqrt(variancia),log = TRUE)
   #
   quo<-(sum(mat.delinea[comfenot,2*candidato[[1]]+1]^2)/sigma2mg)+(1/sigma2.delta)
   media<-((sum(mat.delinea[comfenot,2*candidato[[1]]+1]*residuosmg)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta1<-dnorm(deltasp1,media,sqrt(variancia),log = TRUE)
   #
   vet_coef_parc<-rbind(candidato[[3]],vet.coef[candidato[[1]]*2,1],vet.coef
[candidato[[1]]*2+1,1])
   mat_delinea_parc<-cbind(candidato[[2]],mat.delinea[,candidato[[1]]*2],
mat.delinea[,candidato[[1]]*2+1])
   vet_coef_parc<-matrix(vet_coef_parc[-(2*candidato[[11]][1]),],ncol=1)
   mat_delinea_parc<-mat_delinea_parc[,-(2*candidato[[11]][1])]
   residuosparc<-dados[comfenot,1]-mat_delinea_parc[comfenot,]%*%vet_coef_parc
   quo<-(sum(mat.delinea[comfenot,2*candidato[[10]]]^2)/sigma2mg)+
(1/sigma2.alpha)
   media<-((sum(mat.delinea[comfenot,2*candidato[[10]]]*residuosparc)/sigma2mg)+
(media.alpha/sigma2.alpha))/quo
   variancia<-1/quo
   postalpha2<-dnorm(alphasp2,media,sqrt(variancia),log = TRUE)
   #
   vet_coef_parc<-matrix(vet.coef[-(2*candidato[[10]]+1)],ncol=1)
   mat_delinea_parc<-mat.delinea[,-(2*candidato[[10]]+1)]
   residuosparc<-dados[comfenot,1]-mat_delinea_parc[comfenot,]%*%vet_coef_parc
   quo<-(sum(mat.delinea[comfenot,2*candidato[[10]]+1]^2)/sigma2mg)+
(1/sigma2.delta)
   media<-((sum(mat.delinea[comfenot,2*candidato[[10]]+1]*residuosparc)/sigma2mg)+
(media.delta/sigma2.delta))/quo
   variancia<-1/quo
   postdelta2<-dnorm(deltasp2,media,sqrt(variancia),log = TRUE)
   #
   res.mi<-residuos+vet.coef[1,1]
```

```
   quo<-(length(residuos)/sigma2mg)+(1/sigma2.mi)
   media<-((sum(res.mi)/sigma2mg)+(media.mi/sigma2.mi))/quo
   variancia<-1/quo
   postmi<-dnorm(mi,media,sqrt(variancia),log = TRUE)
   #
   aa1<-(length(residuos)/2)+neta.a
   bb1<-(sum(residuos^2)/2)+neta.b
   postsigma2<-dgamma((1/sigma2),aa1,bb1,log = TRUE)
   #
   probace<-1/prob.aceitacao.split(residuos,residuosmg,sigma2mg,sigma2,
mi,mimg,media.mi,sigma2.mi,neta.a,neta.b,alphamg,alphasp1,alphasp2,
media.alpha,sigma2.alpha,deltamg,deltasp1,deltasp2,media.delta,sigma2.delta,
loc.marc,num.QTLsmg,num.QTLs,psplit,pmerge,pmarc,plambdasp,plambdamg,
postalphamg,postdeltamg,postmimg,postsigma2mg,postalpha1,postalpha2,
postdelta1,postdelta2,postmi,postsigma2,pri.pos,pri.pos.mg)
   #
   probacemerge[int]<-probace
   aux2<-runif(1)
   if (aux2<probace){
    pos.qtls<-candidato[[12]]
    num.QTLs<-length(pos.qtls)
    mat.alelo.qtls<-candidato[[13]]
    mat.delinea<-candidato[[2]]
    vet.coef<-candidato[[3]]
    sigma2.vig<-candidato[[4]]
    residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)}}
  #
  #### update QTLs position - Metropolis Hastings
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    M.esq<-sum(loc.marc<=pos.qtls[i])
    M.dir<-num.marc-(sum(loc.marc>=pos.qtls[i]))+1
    loc.cand<-runif(1,min=loc.marc[M.esq],max=loc.marc[M.dir])
    #
    vet.delinea<-mat.delinea
    mat.alelo.qtlsc<-mat.alelo.qtls
    r1c<-Haldane(abs(loc.cand-loc.marc[M.esq]))
    r2c<-Haldane(abs(loc.cand-loc.marc[M.dir]))
    r1<-Haldane(abs(pos.qtls[i]-loc.marc[M.esq]))
    r2<-Haldane(abs(pos.qtls[i]-loc.marc[M.dir]))
    r12<-Haldane(abs(loc.marc[M.dir]-loc.marc[M.esq]))
    matrec1<-matrix(c(1-r1,r1,r1,1-r1),2,2,byrow=TRUE)
    matrec2<-matrix(c(1-r2,r2,r2,1-r2),2,2,byrow=TRUE)
    matrec1c<-matrix(c(1-r1c,r1c,r1c,1-r1c),2,2,byrow=TRUE)
    matrec2c<-matrix(c(1-r2c,r2c,r2c,1-r2c),2,2,byrow=TRUE)
    probQTLc<-numeric(3)
    probQTL<-numeric(3)
    logdens<-numeric(3)
    probacum<-0
    probacumc<-0
    densacum<-0
    densacumc<-0
    numer<-denom<-0
    #
    # update QTLs genotype of founders - Gibbs sampling
    #
    gen<-c(-1,0,1)
    for (j in 1:length(founders)){
      if (j %in% comfenot){
     for (l in 1:3){
      probQTLc[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1c)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2c))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      dom<-1-abs(gen[l])
      vet.delinea[j,2*i]<-gen[l]
      vet.delinea[j,(2*i)+1]<-dom
      logdens[l]<-dnorm(dados[j,1],vet.delinea[j,]%*%vet.coef,sqrt(sigma2.vig),
log=TRUE)}
     probc<-exp(probQTLc+logdens-max(probQTLc+logdens))/sum(exp(probQTLc+
logdens-max(probQTLc+logdens)))
     prob<-exp(probQTL+logdens-max(probQTL+logdens))/sum(exp(probQTL+
logdens-max(probQTL+logdens)))
```

```
     ger.gen<-rDiscreta(probc)
     vet.delinea[j,2*i]<-gen[ger.gen]
     vet.delinea[j,(2*i)+1]<-1-abs(vet.delinea[j,2*i])
     probacumc<-probacumc+log(probc[ger.gen])
     densacumc<-densacumc+logdens[ger.gen]
     numer<-numer+probQTLc[ger.gen]
     gen.at<-which(gen==mat.delinea[j,2*i])
     probacum<-probacum+log(prob[gen.at])
     densacum<-densacum+logdens[gen.at]
     denom<-denom+probQTL[gen.at]} else {
     for (l in 1:3){
      probQTLc[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1c)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2c))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      dom<-1-abs(gen[l])
      vet.delinea[j,2*i]<-gen[l]
      vet.delinea[j,(2*i)+1]<-dom}
     probc<-exp(probQTLc)/sum(exp(probQTLc))
     prob<-exp(probQTL)/sum(exp(probQTL))
     ger.gen<-rDiscreta(probc)
     vet.delinea[j,2*i]<-gen[ger.gen]
     vet.delinea[j,(2*i)+1]<-1-abs(vet.delinea[j,2*i])}}
    #
    ale.pai<-matrix(99,nrow=length(founders),1)
    ale.mae<-matrix(99,nrow=length(founders),1)
    probale<-numeric(2)
    for (k in 1:length(founders)){
     if (vet.delinea[k,2*i]==-1){
      ale.mae[k,1]<-1
      ale.pai[k,1]<-1}
     if (vet.delinea[k,2*i]==1){
      ale.mae[k,1]<-2
      ale.pai[k,1]<-2}
     if (vet.delinea[k,2*i]==0){
      for (ale in 1:2) probale[ale]<-matrec1[alelo.paterno[k,M.esq],ale]*matrec2
```

```
[ale,alelo.paterno[k,M.dir]]
      probale<-probale/sum(probale)
#      aux2<-which(probale==max(probale))-1
      aux2<-rDiscreta(probale)-1
      ale.mae[k,1]<-1^(aux2)*2^(1-aux2)
      ale.pai[k,1]<-2^(aux2)*1^(1-aux2)}
     mat.alelo.qtlsc[k,(2*i-1)]<-ale.pai[k,1]
     mat.alelo.qtlsc[k,(2*i)]<-ale.mae[k,1]}
    #
    matriz.prob.S<-numeric()
    matriz.prob.Sc<-numeric()
    probSpat<-numeric(2)
    probSmat<-numeric(2)
    probSpatc<-numeric(2)
    probSmatc<-numeric(2)
    #
    # update QTLs genotype of nonfounders - Gibbs sampling
    #
    for (j in 1:length(nonfounders)){
     if ((length(founders)+j) %in% comfenot){
     for (l in 1:2){
      probSmat[l]<-matrec1[Sm[j,M.esq],l]*matrec2[l,Sm[j,M.dir]]
      probSpat[l]<-matrec1[Sp[j,M.esq],l]*matrec2[l,Sp[j,M.dir]]
      probSmatc[l]<-matrec1c[Sm[j,M.esq],l]*matrec2c[l,Sm[j,M.dir]]
      probSpatc[l]<-matrec1c[Sp[j,M.esq],l]*matrec2c[l,Sp[j,M.dir]]}
     probSpat<-probSpat/sum(probSpat)
     probSmat<-probSmat/sum(probSmat)
     probSpatc<-probSpatc/sum(probSpatc)
     probSmatc<-probSmatc/sum(probSmatc)
     matriz.prob.S<-c(probSpat*probSmat[1],probSpat*probSmat[2])
     matriz.prob.Sc<-c(probSpatc*probSmatc[1],probSpatc*probSmatc[2])
     #
     mat.possc<-cbind(matrix(c(1,1,2,1,1,2,2,2),4,2,byrow=TRUE),matrix(0,4,2))
     for (l in 1:4){
      mat.possc[l,3]<-mat.alelo.qtls[mat.ped.pais[j,1],2*i-(2-mat.possc[l,1])]
      mat.possc[l,4]<-mat.alelo.qtls[mat.ped.pais[j,2],2*i-(2-mat.possc[l,2])]}
     mat.possc<-cbind(mat.possc,matrix(matriz.prob.S,4,1))
     if ((sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8) &
(sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8)){
      mat.possc[1,5]<-1
      mat.possc<-matrix(mat.possc[-c(2,3,4),],1,5)} else {
      if (sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8){
       mat.possc[1,5]<-mat.possc[1,5]+mat.possc[2,5]
       mat.possc[3,5]<-mat.possc[3,5]+mat.possc[4,5]
       mat.possc<-mat.possc[-c(2,4),]} else {
       if (sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8){
        mat.possc[1,5]<-mat.possc[1,5]+mat.possc[3,5]
        mat.possc[2,5]<-mat.possc[2,5]+mat.possc[4,5]
        mat.possc<-mat.possc[-c(3,4),]}}}
     mat.possc<-cbind(mat.possc,matrix(0,nrow(mat.possc),2))
     for (l in 1:nrow(mat.possc)){
      if (mat.possc[l,3]==1 & mat.possc[l,4]==1) mat.possc[l,6]<--1
      if (mat.possc[l,3]==2 & mat.possc[l,4]==2) mat.possc[l,6]<-1
      if (mat.possc[l,3]!= mat.possc[l,4]) mat.possc[l,6]<-0}
     mat.possc[,7]<-1-abs(mat.possc[,6])
     #
     logdens<-numeric(nrow(mat.possc))
     for (l in 1:nrow(mat.possc)){
      vet.delinea[length(founders)+j,2*i]<-mat.possc[l,6]
      vet.delinea[length(founders)+j,(2*i)+1]<-mat.possc[l,7]
      logdens[l]<-dnorm(dados[length(founders)+j,1],vet.delinea[
length(founders)+j,]%*%vet.coef,sqrt(sigma2.vig),log=TRUE)}
     probc<-exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+
logdens))/sum(exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+logdens)))
     gen<-which(mat.possc[,3]==mat.alelo.qtls[length(founders)+j,(2*i)-1] &
mat.possc[,4]==mat.alelo.qtls[length(founders)+j,2*i])
     probacum<-probacum+log(probc[gen])
     densacum<-densacum+logdens[gen]
     denom<-denom+log(mat.possc[gen,5])
     #
     mat.possc<-cbind(matrix(c(1,1,2,1,1,2,2,2),4,2,byrow=TRUE),matrix(0,4,2))
     for (l in 1:4){
      mat.possc[l,3]<-mat.alelo.qtlsc[mat.ped.pais[j,1],2*i-(2-mat.possc[l,1])]
      mat.possc[l,4]<-mat.alelo.qtlsc[mat.ped.pais[j,2],2*i-(2-mat.possc[l,2])]}
     mat.possc<-cbind(mat.possc,matrix(matriz.prob.Sc,4,1))
     if ((sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8) &
(sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8)){
      mat.possc[1,5]<-1
      mat.possc<-matrix(mat.possc[-c(2,3,4),],1,5)} else {
      if (sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8){
       mat.possc[1,5]<-mat.possc[1,5]+mat.possc[2,5]
       mat.possc[3,5]<-mat.possc[3,5]+mat.possc[4,5]
       mat.possc<-mat.possc[-c(2,4),]} else {
       if (sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8){
        mat.possc[1,5]<-mat.possc[1,5]+mat.possc[3,5]
        mat.possc[2,5]<-mat.possc[2,5]+mat.possc[4,5]
        mat.possc<-mat.possc[-c(3,4),]}}}
     mat.possc<-cbind(mat.possc,matrix(0,nrow(mat.possc),2))
     for (l in 1:nrow(mat.possc)){
      if (mat.possc[l,3]==1 & mat.possc[l,4]==1) mat.possc[l,6]<--1
      if (mat.possc[l,3]==2 & mat.possc[l,4]==2) mat.possc[l,6]<-1
      if (mat.possc[l,3]!= mat.possc[l,4]) mat.possc[l,6]<-0}
     mat.possc[,7]<-1-abs(mat.possc[,6])
     #
     logdens<-numeric(nrow(mat.possc))
     for (l in 1:nrow(mat.possc)){
      vet.delinea[length(founders)+j,2*i]<-mat.possc[l,6]
      vet.delinea[length(founders)+j,(2*i)+1]<-mat.possc[l,7]
      logdens[l]<-dnorm(dados[length(founders)+j,1],vet.delinea
[length(founders)+j,]%*%vet.coef,sqrt(sigma2.vig),log=TRUE)}
     probc<-exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+
logdens))/sum(exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+logdens)))
     ger.gen<-rDiscreta(probc)
     vet.delinea[length(founders)+j,2*i]<-mat.possc[ger.gen,6]
     vet.delinea[length(founders)+j,(2*i)+1]<-mat.possc[ger.gen,7]
     mat.alelo.qtlsc[length(founders)+j,(2*i-1)]<-mat.possc[ger.gen,3]
     mat.alelo.qtlsc[length(founders)+j,(2*i)]<-mat.possc[ger.gen,4]
     probacumc<-probacumc+log(probc[ger.gen])
     densacumc<-densacumc+logdens[ger.gen]
     numer<-numer+log(mat.possc[ger.gen,5])} else {
     for (l in 1:2){
      probSmat[l]<-matrec1[Sm[j,M.esq],l]*matrec2[l,Sm[j,M.dir]]
      probSpat[l]<-matrec1[Sp[j,M.esq],l]*matrec2[l,Sp[j,M.dir]]
      probSmatc[l]<-matrec1c[Sm[j,M.esq],l]*matrec2c[l,Sm[j,M.dir]]
      probSpatc[l]<-matrec1c[Sp[j,M.esq],l]*matrec2c[l,Sp[j,M.dir]]}
     probSpat<-probSpat/sum(probSpat)
     probSmat<-probSmat/sum(probSmat)
     probSpatc<-probSpatc/sum(probSpatc)
     probSmatc<-probSmatc/sum(probSmatc)
     matriz.prob.S<-c(probSpat*probSmat[1],probSpat*probSmat[2])
     matriz.prob.Sc<-c(probSpatc*probSmatc[1],probSpatc*probSmatc[2])
     #
     mat.possc<-cbind(matrix(c(1,1,2,1,1,2,2,2),4,2,byrow=TRUE),matrix(0,4,2))
     for (l in 1:4){
      mat.possc[l,3]<-mat.alelo.qtlsc[mat.ped.pais[j,1],2*i-(2-mat.possc[l,1])]
      mat.possc[l,4]<-mat.alelo.qtlsc[mat.ped.pais[j,2],2*i-(2-mat.possc[l,2])]}
     mat.possc<-cbind(mat.possc,matrix(matriz.prob.Sc,4,1))
     if ((sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8) &
(sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8)){
      mat.possc[1,5]<-1
      mat.possc<-matrix(mat.possc[-c(2,3,4),],1,5)} else {
      if (sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8){
       mat.possc[1,5]<-mat.possc[1,5]+mat.possc[2,5]
       mat.possc[3,5]<-mat.possc[3,5]+mat.possc[4,5]
       mat.possc<-mat.possc[-c(2,4),]} else {
       if (sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8){
        mat.possc[1,5]<-mat.possc[1,5]+mat.possc[3,5]
        mat.possc[2,5]<-mat.possc[2,5]+mat.possc[4,5]
        mat.possc<-mat.possc[-c(3,4),]}}}
     mat.possc<-cbind(mat.possc,matrix(0,nrow(mat.possc),2))
     for (l in 1:nrow(mat.possc)){
      if (mat.possc[l,3]==1 & mat.possc[l,4]==1) mat.possc[l,6]<--1
      if (mat.possc[l,3]==2 & mat.possc[l,4]==2) mat.possc[l,6]<-1
      if (mat.possc[l,3]!= mat.possc[l,4]) mat.possc[l,6]<-0}
     mat.possc[,7]<-1-abs(mat.possc[,6])
     #
     probc<-mat.possc[,5]/sum(mat.possc[,5])
     ger.gen<-rDiscreta(probc)
     vet.delinea[length(founders)+j,2*i]<-mat.possc[ger.gen,6]
     vet.delinea[length(founders)+j,(2*i)+1]<-mat.possc[ger.gen,7]
     mat.alelo.qtlsc[length(founders)+j,(2*i-1)]<-mat.possc[ger.gen,3]
     mat.alelo.qtlsc[length(founders)+j,(2*i)]<-mat.possc[ger.gen,4]}}
```

```
    #
    paceit<-exp(densacumc+numer+probacum-densacum-denom-probacumc)
    aux3<-runif(1)
    if (aux3<paceit){
     pos.qtls[i]<-loc.cand
     mat.delinea[,2*i]<-vet.delinea[,2*i]
     mat.delinea[,(2*i)+1]<-vet.delinea[,(2*i)+1]
     mat.alelo.qtls[,(2*i-1)]<-mat.alelo.qtlsc[,(2*i-1)]
     mat.alelo.qtls[,(2*i)]<-mat.alelo.qtlsc[,(2*i)]}}}
  #
  #### update QTLs genotype - Gibbs sampling
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    M.esq<-sum(loc.marc<=pos.qtls[i])
    M.dir<-num.marc-(sum(loc.marc>=pos.qtls[i]))+1
    r1<-Haldane(abs(pos.qtls[i]-loc.marc[M.esq]))
    r2<-Haldane(abs(pos.qtls[i]-loc.marc[M.dir]))
    r12<-Haldane(abs(loc.marc[M.dir]-loc.marc[M.esq]))
    matrec1<-matrix(c(1-r1,r1,r1,1-r1),2,2,byrow=TRUE)
    matrec2<-matrix(c(1-r2,r2,r2,1-r2),2,2,byrow=TRUE)
    probQTL<-numeric(3)
    logdens<-numeric(3)
    matriz.prob.S<-numeric()
    probSpat<-numeric(2)
    probSmat<-numeric(2)
    gen<-c(-1,0,1)
    #
    for (j in 1:length(founders)){
     if (j %in% comfenot){
     for (l in 1:3){
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))
      dom<-1-abs(gen[l])
      vet.delinea<-mat.delinea[j,]
      vet.delinea[2*i]<-gen[l]
      vet.delinea[(2*i)+1]<-dom
      logdens[l]<-dnorm(dados[j,1],vet.delinea%*%vet.coef,sqrt(sigma2.vig),
log=TRUE)}
     prob<-exp(probQTL+logdens-max(probQTL+logdens))/sum(exp(probQTL+
logdens-max(probQTL+logdens)))
     mat.delinea[j,2*i]<-gen[rDiscreta(prob)]
     mat.delinea[j,(2*i)+1]<-1-abs(mat.delinea[j,2*i])} else {
     for (l in 1:3){
      probQTL[l]<-log((calc.prob.gen(dados[j,(M.esq+1)],gen[l],r1)*
calc.prob.gen(gen[l],dados[j,(M.dir+1)],r2))/calc.prob.gen(dados
[j,(M.esq+1)],dados[j,(M.dir+1)],r12))}
     prob<-exp(probQTL)/sum(exp(probQTL))
     mat.delinea[j,2*i]<-gen[rDiscreta(prob)]
     mat.delinea[j,(2*i)+1]<-1-abs(mat.delinea[j,2*i])}}
    #
    monoto<-length(table(mat.delinea[1:length(founders),2*i]))
    if (monoto==1){
     indiv<-sample(seq(1:length(founders)),1)
     if (mat.delinea[indiv,2*i]==-1) mat.delinea[indiv,2*i]<-0
     if (mat.delinea[indiv,2*i]==0) mat.delinea[indiv,2*i]<-1
     if (mat.delinea[indiv,2*i]==1) mat.delinea[indiv,2*i]<-0
     mat.delinea[indiv,(2*i)+1]<-1-abs(mat.delinea[indiv,2*i])}
    #
    ale.pai<-matrix(99,nrow=length(founders),1)
    ale.mae<-matrix(99,nrow=length(founders),1)
    probale<-numeric(2)
    for (k in 1:length(founders)){
     if (mat.delinea[k,2*i]==-1){
      ale.mae[k,1]<-1
      ale.pai[k,1]<-1}
     if (mat.delinea[k,2*i]==1){
      ale.mae[k,1]<-2
      ale.pai[k,1]<-2}
     if (mat.delinea[k,2*i]==0){
      for (ale in 1:2) probale[ale]<-matrec1[alelo.paterno[k,M.esq],ale]*
matrec2[ale,alelo.paterno[k,M.dir]]
      probale<-probale/sum(probale)
      aux2<-rDiscreta(probale)-1
      ale.mae[k,1]<-1^(aux2)*2^(1-aux2)
      ale.pai[k,1]<-2^(aux2)*1^(1-aux2)}
     mat.alelo.qtls[k,(2*i-1)]<-ale.pai[k,1]
     mat.alelo.qtls[k,(2*i)]<-ale.mae[k,1]}
    #
    for (j in 1:length(nonfounders)){
      if ((length(founders)+j) %in% comfenot){
     for (l in 1:2){
      probSmat[l]<-matrec1[Sm[j,M.esq],l]*matrec2[l,Sm[j,M.dir]]
      probSpat[l]<-matrec1[Sp[j,M.esq],l]*matrec2[l,Sp[j,M.dir]]}
     probSpat<-probSpat/sum(probSpat)
     probSmat<-probSmat/sum(probSmat)
     matriz.prob.S<-c(probSpat*probSmat[1],probSpat*probSmat[2])
     mat.possc<-cbind(matrix(c(1,1,2,1,1,2,2,2),4,2,byrow=TRUE),matrix(0,4,2))
     for (l in 1:4){
      mat.possc[l,3]<-mat.alelo.qtls[mat.ped.pais[j,1],2*i-(2-mat.possc[l,1])]
      mat.possc[l,4]<-mat.alelo.qtls[mat.ped.pais[j,2],2*i-(2-mat.possc[l,2])]}
     mat.possc<-cbind(mat.possc,matrix(matriz.prob.S,4,1))
     if ((sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8) &
(sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8)){
      mat.possc[1,5]<-1
      mat.possc<-matrix(mat.possc[-c(2,3,4),],1,5)} else {
      if (sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8){
       mat.possc[1,5]<-mat.possc[1,5]+mat.possc[2,5]
       mat.possc[3,5]<-mat.possc[3,5]+mat.possc[4,5]
       mat.possc<-mat.possc[-c(2,4),]} else {
       if (sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8){
        mat.possc[1,5]<-mat.possc[1,5]+mat.possc[3,5]
        mat.possc[2,5]<-mat.possc[2,5]+mat.possc[4,5]
        mat.possc<-mat.possc[-c(3,4),]}}}
     mat.possc<-cbind(mat.possc,matrix(0,nrow(mat.possc),2))
     for (l in 1:nrow(mat.possc)){
      if (mat.possc[l,3]==1 & mat.possc[l,4]==1) mat.possc[l,6]<--1
      if (mat.possc[l,3]==2 & mat.possc[l,4]==2) mat.possc[l,6]<-1
      if (mat.possc[l,3]!= mat.possc[l,4]) mat.possc[l,6]<-0}
     mat.possc[,7]<-1-abs(mat.possc[,6])
     #
     logdens<-numeric(nrow(mat.possc))
     vet.delinea<-mat.delinea[length(founders)+j,]
     for (l in 1:nrow(mat.possc)){
      vet.delinea[2*i]<-mat.possc[l,6]
      vet.delinea[(2*i)+1]<-mat.possc[l,7]
      logdens[l]<-dnorm(dados[length(founders)+j,1],vet.delinea%*%
vet.coef,sqrt(sigma2.vig),log=TRUE)}
     prob<-exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+logdens))/
sum(exp(log(mat.possc[,5])+logdens-max(log(mat.possc[,5])+logdens)))
     ger.gen<-rDiscreta(prob)
     mat.delinea[length(founders)+j,2*i]<-mat.possc[ger.gen,6]
     mat.delinea[length(founders)+j,(2*i)+1]<-mat.possc[ger.gen,7]
     mat.alelo.qtls[length(founders)+j,(2*i-1)]<-mat.possc[ger.gen,3]
     mat.alelo.qtls[length(founders)+j,(2*i)]<-mat.possc[ger.gen,4]} else {
     for (l in 1:2){
      probSmat[l]<-matrec1[Sm[j,M.esq],l]*matrec2[l,Sm[j,M.dir]]
      probSpat[l]<-matrec1[Sp[j,M.esq],l]*matrec2[l,Sp[j,M.dir]]}
     probSpat<-probSpat/sum(probSpat)
     probSmat<-probSmat/sum(probSmat)
     matriz.prob.S<-c(probSpat*probSmat[1],probSpat*probSmat[2])
     mat.possc<-cbind(matrix(c(1,1,2,1,1,2,2,2),4,2,byrow=TRUE),matrix(0,4,2))
     for (l in 1:4){
      mat.possc[l,3]<-mat.alelo.qtls[mat.ped.pais[j,1],2*i-(2-mat.possc[l,1])]
      mat.possc[l,4]<-mat.alelo.qtls[mat.ped.pais[j,2],2*i-(2-mat.possc[l,2])]}
     mat.possc<-cbind(mat.possc,matrix(matriz.prob.S,4,1))
     if ((sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8) &
(sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8)){
      mat.possc[1,5]<-1
      mat.possc<-matrix(mat.possc[-c(2,3,4),],1,5)} else {
      if (sum(mat.possc[,3])==4 | sum(mat.possc[,3])==8){
       mat.possc[1,5]<-mat.possc[1,5]+mat.possc[2,5]
       mat.possc[3,5]<-mat.possc[3,5]+mat.possc[4,5]
       mat.possc<-mat.possc[-c(2,4),]} else {
       if (sum(mat.possc[,4])==4 | sum(mat.possc[,4])==8){
        mat.possc[1,5]<-mat.possc[1,5]+mat.possc[3,5]
        mat.possc[2,5]<-mat.possc[2,5]+mat.possc[4,5]
        mat.possc<-mat.possc[-c(3,4),]}}}
     mat.possc<-cbind(mat.possc,matrix(0,nrow(mat.possc),2))
     for (l in 1:nrow(mat.possc)){
      if (mat.possc[l,3]==1 & mat.possc[l,4]==1) mat.possc[l,6]<--1
      if (mat.possc[l,3]==2 & mat.possc[l,4]==2) mat.possc[l,6]<-1
      if (mat.possc[l,3]!= mat.possc[l,4]) mat.possc[l,6]<-0}
     mat.possc[,7]<-1-abs(mat.possc[,6])
     #
     prob<-mat.possc[,5]/sum(mat.possc[,5])
     ger.gen<-rDiscreta(prob)
     mat.delinea[length(founders)+j,2*i]<-mat.possc[ger.gen,6]
     mat.delinea[length(founders)+j,(2*i)+1]<-mat.possc[ger.gen,7]
     mat.alelo.qtls[length(founders)+j,(2*i-1)]<-mat.possc[ger.gen,3]
     mat.alelo.qtls[length(founders)+j,(2*i)]<-mat.possc[ger.gen,4]}}}}
  residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)
  #
  #### update mu - Gibbs sampling
  #
  vet.coef[1,1]<-poster.mi(media.mi,sigma2.mi,sigma2.vig,residuos,
vet.coef[1,1])[[1]]
  residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)
  #
  #### update additive and dominance effect - Gibbs sampling
  #
  if (num.QTLs>0){
   for (i in 1:num.QTLs){
    vet.coef[(2*i),1]<-poster.alpha(media.alpha,sigma2.alpha,sigma2.vig,
residuos,vet.coef[(2*i),1],mat.delinea[comfenot,(2*i)])[[1]]
    residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)
    vet.coef[(2*i)+1,1]<-poster.delta(media.delta,sigma2.delta,sigma2.vig,
residuos,vet.coef[(2*i)+1,1],mat.delinea[comfenot,(2*i)+1])[[1]]
    residuos<-dados[comfenot,1]-(mat.delinea[comfenot,]%*%vet.coef)}}
  #
  #### update error variance - sigma2
  #
  sigma2.vig<-poster.sigma2(neta.a,neta.b,residuos)[[1]]
  #
  #################### export results
  #
  if (int>burnin & int%%saltos==0){
   cat('',num.QTLs,file=paste(caminhosim,"numero_QTLs_b2_dd2_crom",crom,
".txt",sep=""),append=T)
```

```
   cat('',pos.qtls,file=paste(caminhosim,"posicao_QTLs_b2_dd2_crom",crom,
".txt",sep=""),append=T)
   cat('',vet.coef,file=paste(caminhosim,"vetor_coeficientes_b2_dd2_crom",crom,
".txt",sep=""),append=T)
   cat('',sigma2.vig,file=paste(caminhosim,"sigma2_b2_dd2_crom",crom,
".txt",sep=""),append=T)}
}
cat('',indrejtotal,file=paste(caminhosim,"indrej_b2_dd2_crom",crom,
".txt",sep=""),append=T)
cat('',round(probacetotal,2),file=paste(caminhosim,"prob_rej_b2_dd2_crom",crom,
".txt",sep=""),append=T)
cat('',round(probacemerge,2),file=paste(caminhosim,"prob_rej_mg_b2_dd2_crom",crom,
".txt",sep=""),append=T)
```

5.6.2 Codes used to simulate data sets in SimPed

In this section, we show the codes used to simulate data sets in SimPed

```
#
# family structure
#
1       1       0       0       1       1
1       2       0       0       2       1
1       3       0       0       2       1
1       4       1       2       1       1
1       5       1       2       2       1
1       6       0       0       1       1
1       7       0       0       2       1
1       8       4       3       1       1
1       9       6       5       2       1
1       10      8       7       1       1
#
# Input file
#
pedin3.pre pedfile1.pre << name of pedigree file, name of output file
23221 1601  21001   << three random seeds
0  0 << # of columns for affection status/quantitative trait,
# autosomal data to be generated
50 <<number of replicates
200 20 << Total # of marker loci, # of times pattern to be repeated
```

```
3 << "1" recomb fraction, "2" Kosambi map distance & "3" Haldane map distance
10 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
1 7  10  1 << "1" for haplotypes, # of haplotypes, # of marker loci, # of times
# pattern repeated
0.25 0.25 0.125 0.125 0.1 0.1 0.05<< the frequency for each haplotype
1 1 1 1 1 1 1 1 1 1    < << observed alleles for each haplotype
2 2 2 2 2 2 2 2 2 2
1 1 2 2 2 2 2 2 2 2
1 1 1 1 1 1 1 2 2 2
2 2 2 1 1 1 1 1 1 1
1 1 1 2 2 2 2 2 2 2
2 2 1 1 2 2 1 1 2 2
```

Chapter

6

A marginal NDP – clustering distributions [1]

In this chapter, we introduce a marginal version of the nested Dirichlet process to cluster distributions or histograms. We apply the model to cluster genes by patterns of gene-gene interaction. The proposed approach is based on the nested partition that is implied in the original construction of the nested Dirichlet process. It allows simulation exact inference, as opposed to a truncated Dirichlet process approximation. More importantly, the construction highlights the nature of the nested Dirichlet process as a nested partition of experimental units.

We apply the proposed model to inference on clustering genes related to DNA mismatch repair (DMR) by the distribution of gene-gene interactions with other genes. Gene-gene interactions are recorded as coefficients in an auto-logistic model for the co-expression of two genes, adjusting for copy number variation, methylation and protein activation. These coefficients are extracted from an online database, called Zodiac, computed based on The Cancer Genome Atlas (TCGA) data.

We compare results with a variation of k-means clustering that is set up to cluster distributions. The proposed inference shows favorable performance, under simulated conditions and also in the real data set.

6.1 Introduction

We discuss clustering for distributions of gene-gene interactions. The aim is inference on groups of genes that are similar in terms of the distribution of their interactions with other genes. We use a nested Dirichlet process *a priori* (NDP) (Rodriguez *et al*, 2008) for the desired

[1] This chapter is based on the manuscript "A Marginal NDP – Clustering Distributions" submitted for publication (Zuanetti *et al*, Submitted a). The manuscript is a joint work supervised by Prof. Dr. Peter Müller and with colaboration of Dr. Yitan Zhu, Dr. Shengjie Yang and Prof. Dr. Yuan Ji.

grouping of distributions. We show how inference in the NDP can be implemented exactly, rather than the approximate inference based on finite truncations that is used in the original NDP. The proposed Markov chain Monte Carlo scheme clarifies the nature of the NDP as a prior for nested clustering. We apply the model and proposed implementation to cluster DNA mismatch-repair (DMR) genes by similar distributions of gene-gene interactions, and find well defined clusters.

Recent literature proposed several alternative methods for such nested partitions. Wade *et al.* (2011) introduce the enriched Dirichlet process, which can be set up to define exactly the same nested partition as the NDP, although the construction starts quite differently. Other Dirichlet process-based models for nested partition include Rodríguez & Ghosh (2012) and Lee *et al.* (2013) who in contrast to the NDP impose matching nested partitions for lower level units. Details are discussed below, including how the enriched Dirichlet process generates the same nested partition as the NDP.

Genetic interactions play a critical role in cancer development and have been extensively studied. A typical example is Zodiac (Zhu *et al*, 2015), an on-line search engine for genetic interactions. Zodiac reports inference on pairwise gene-gene interactions based on statistical analysis of data for thousands of samples from dozen of cancer types collected by The Cancer Genome Atlas (TCGA). For each pair of genes, statistical inference in Zodiac is based on an auto-logistic model for gene expression (GE), protein activation (where available), methylation and copy number variation for that pair. For the upcoming discussion the GE-GE coefficients in that model are the data. We do not make use of the detail methods of how Zodiac derives them (using a model proposed in Mitra et al. 2013). Instead we consider the problem of grouping genes by similar interaction patterns represented by the distribution of auto-logistic GE-GE coefficients for each gene. That is, we aim to group genes by similar histograms of reported GE-GE coefficients. Such inference is helpful to understand the relative importance of the genes, for example, by identifying a group of genes that has, systematically, the highest correlations with many other genes.

Rodriguez *et al.* (2008) address a formally similar problem. They consider quality of care measurements for hospitals across the US, and cluster states by similar distributions of quality of care outcomes. Their approach uses Markov chain Monte Carlo (MCMC) simulation based on an approximation with a truncated stick-breaking construction to estimate the model. Truncation levels are chosen by empirical analysis. In this manuscript, we develop an exact simulation MCMC algorithm. The proposed implementation is a marginal algorithm. The random distributions are analytically marginalized and inference proceeds on the implied nested partition. The implementation uses an approximation of *a posteriori* inference in Dirichlet process mixture models that was developed in MacEachern *et al.* (1999). We apply the proposed approach of clustering distributions to group DMR genes.

The chapter is organized as follows: In Section 6.2 we briefly define the NDP and propose the

MCMC algorithm to estimate a marginal NDP in Section 6.3. Section 6.4 presents a simulated example to illustrate the performance of NDP in clustering histograms with same features of GE coefficients distributions. We apply the proposed procedure to cluster the coefficients distributions of DMR genes in Section 6.5 and compare the NDP results with results under a k-means method, which is a widely used deterministic method for clustering. Finally, we close with a brief discussion in Section 6.6 and appendices in Section 6.7.

6.2 Nested Dirichlet process

The data are GE-GE coefficients in auto-logistic models for gene-expression, protein activation, methylation and copy number variation for pairs of genes. Let $\boldsymbol{y}_j = (y_{j1}, \ldots, y_{jI_j})$, for $j = 1, \ldots, J$, be the GE-GE coefficients for the gene expression of gene j and I_j other genes. The y_{ji} are the data in the upcoming discussion. We assume gene-specific sampling models F_j. That is, we assume $y_{ji} \sim F_j$. Our interest is to cluster $\{F_j, j = 1, \ldots, J\}$.

We start with a brief description of the NDP model. Consider a collection of distributions $\{G_1, \ldots, G_J\}$. The NDP assumes $G_j \mid Q \sim Q$, $j = 1, \ldots, J$, and $Q \sim \text{DP}(\alpha\text{DP}(\beta G_0))$, where DP denotes a Dirichlet process, G_0 is a non-atomic baseline probability measure, $\alpha, \beta > 0$ are the total mass parameters. That is, $Q = \sum_h w_h \delta_{G_h^{\star\star}}$ is a discrete distribution of distributions and $G_h^{\star\star} \sim \text{DP}(\beta G_0)$, i.i.d. Similarly, $G_h^{\star\star} = \sum_f v_f \delta_{\boldsymbol{\theta}_f^{\star\star}}$ with $\boldsymbol{\theta}_f^{\star\star} \sim G_0$, i.i.d. The weights w_h in Q are generated by stick-breaking (Sethuraman, 1994) with the total mass parameter α of the outer DP, and the weights v_f in $G_h^{\star\star}$ are generated with the total mass parameter β of the nested, inner DP. We use an additional convolution with a continuous kernel $p(\cdot \mid \boldsymbol{\theta})$,

$$F_j(\cdot) = \int_{\boldsymbol{\theta}} p(\cdot \mid \boldsymbol{\theta}) G_j(d\boldsymbol{\theta}), \tag{6.1}$$

to achieve a continuous distribution. Here $p(\cdot \mid \boldsymbol{\theta})$ is a sampling model for y_{ji} and $\boldsymbol{\theta}$ is the finite dimensional parameter associated with the sampling model. The collection $\{F_1, \ldots, F_J\}$ is said to follow a nested Dirichlet process (NDP) mixture first introduced in Rodriguez *et al.* (2008). It is a hierarchical model involving two Dirichlet processes. The baseline probability measure for the first DP is given by the second DP. In summary, and replacing (6.1) by a hierarchical model with latent variables $\boldsymbol{\theta}_{ji} \sim G_j$, we have

$$\begin{gathered} y_{ji} \mid \boldsymbol{\theta}_{ji} \overset{indep}{\sim} p(y_{ji} \mid \boldsymbol{\theta}_{ji}) \\ \boldsymbol{\theta}_{ji} \mid G_j \sim G_j \\ G_j \sim Q \text{ and } Q \overset{iid}{\sim} \text{DP}(\alpha\text{DP}(\beta G_0)) \end{gathered} \tag{6.2}$$

The discrete nature of Q gives rise to ties among the G_j. Let $\{G_1^\star, \ldots, G_K^\star\}$ denote the unique elements among the G_j, and let $S_k = \{j : G_j = G_k^\star\}$ denote clusters that are defined by the configuration of these ties. Let $n_k = |S_k|$ and $s_j = k$ when $j \in S_k$ denote cluster size and cluster membership indicators. Alternatively to $\{S_1, \ldots, S_K\}$, the vector $\boldsymbol{s} = (s_1, \ldots, s_n)$ can be used to equivalently identify the partition. We use the two representations interchangeably. It can be shown that sampling G_j from a DP random measure with total mass parameter α implies

$$p(\boldsymbol{s}) \propto \alpha^K \prod_{k=1}^{K} (n_k - 1)! \tag{6.3}$$

with the understanding of indexing clusters by appearance. That is, $s_1 = 1$ and $s_{i+1} \le \max\{s_1, \ldots, s_i\} + 1$. The random partition (6.3) is known as the Polya urn. We will refer to it as $\boldsymbol{s} \sim \mathrm{PU}(\alpha)$. See, for example, Ghoshal (2010) for a review.

Similarly, the discrete nature of $G^\star_k$ gives rise to ties among $\{\boldsymbol{\theta}_{ji};\ j \in S_k \text{ and } i = 1, \ldots, I_j\}$. Let $\{\boldsymbol{\theta}^\star_{k1}, \ldots, \boldsymbol{\theta}^\star_{kL_k}\}$ denote the L_k unique elements, let $R_{k\ell} = \{(j,i) :\ s_j = k \text{ and } \boldsymbol{\theta}_{ji} = \boldsymbol{\theta}^\star_{k\ell}\}$ denote the clusters and $r_{ji} = \ell$ when $(j,i) \in R_{k\ell}$ denote cluster membership indicators. And let $m_{k\ell} = |R_{k\ell}|$ denote the sizes of the clusters. We refer to S_k as distributional clusters, since they are defined by common distributions, and $R_{k\ell}$ as observational clusters since they are defined by ties in the parameters $\boldsymbol{\theta}_{ji}$. Let $\tilde{S}_k = \{(j,i) :\ G_j = G^\star_k \text{ and } i = 1, \ldots, I_j\}$ denote all pairs (j,i) in cluster S_k. That is, $\tilde{S}_k$ describes S_k at the level of the $\boldsymbol{\theta}_{ji}$. Observational clusters are nested within distributional clusters, that is, $\bigcup_{\ell=1}^{L_k} R_{k\ell} = \tilde{S}_k$. Using the latent cluster membership indicators, and letting $\boldsymbol{s} = (s_1, \ldots, s_J)$ and $\boldsymbol{r}_j = (r_{ji},\ i = 1, \ldots, I_j)$ we can rewrite (6.2) as

$$\begin{aligned} y_{ji} \mid s_j = k, r_{ji} = \ell &\overset{indep}{\sim} p(y_{ji} \mid \boldsymbol{\theta}^\star_{k\ell}) \\ \boldsymbol{\theta}^\star_{k\ell} &\sim G_0 \\ \boldsymbol{s} \sim \mathrm{PU}(\alpha) \text{ and } \boldsymbol{r}_j \mid \boldsymbol{s} &\overset{i.i.d.}{\sim} \mathrm{PU}(\beta), \end{aligned} \tag{6.4}$$

$k = 1, \ldots, K$ and $\ell = 1, \ldots, L_k$. The last two representations marginalize with respect to the infinite dimensional Q and $G^\star_k$.

The joint distribution on $G_1, \ldots, G_K$ is defined as

$$p(\boldsymbol{s}, \boldsymbol{r}, (\boldsymbol{\theta}^\star_{k\ell}), \boldsymbol{y}) = \left\{ \prod_{k=1}^{K} \prod_{\ell=1}^{L_k} \left(\prod_{(j,i) \in R_{k\ell}} p(y_{ji} \mid \boldsymbol{\theta}^\star_{k\ell}) \right) \right\} \left\{ \prod_{k=1}^{K} \prod_{\ell=1}^{L_k} G_0(\boldsymbol{\theta}^\star_{k\ell}) \right\} \left\{ \prod_{j=1}^{J} p(\boldsymbol{r}_j \mid \boldsymbol{s}) \right\} p(\boldsymbol{s}). \tag{6.5}$$

We complete the joint model construction with a normal sampling model, $p\left(y_{ji} \mid \boldsymbol{\theta}^\star_{k\ell} = (\mu^\star_{k\ell}, \sigma^{\star 2}_{k\ell})\right) = \mathrm{N}(\mu^\star_{k\ell}, \sigma^{\star 2}_{k\ell})$ and a conjugate baseline measure $G_0(\boldsymbol{\theta}^\star_{k\ell}) = \mathrm{NIG}(\mu_0, \lambda, a, b)$. Here, $\mathrm{NIG}(\mu, \sigma^2 \mid \mu_0, \lambda, a, b)$ denotes a normal inverse gamma distribution for (μ, σ^2). That is, a gamma *a priori* $p(1/\sigma^2) = \mathrm{Ga}(a, b)$ for $1/\sigma^2$ and a conditional normal *a priori* for μ, $p(\mu \mid \sigma^2) = \mathrm{N}(\mu_0, \sigma^2/\lambda)$. The gamma *a priori* is parametrized such that $E(1/\sigma^2) = a/b$.

The statement of the NDP model in (6.2), or equivalently, in (6.4) highlights the nature of the NDP as a random nested partition. The clusters S_k define an upper level partition of experimental units $j = 1, \ldots, J$, in our case genes. Nested within each cluster S_k, $R_{k\ell}$ is a partition of lower level units, in our case the auto-logistic GE-GE coefficients. Similar nested partition models are defined as the enriched DP in Wade *et al.* (2011), nested clustering in Rodríguez & Ghosh (2012) and as local clustering in Lee *et al.* (2013). In fact, the enriched DP can be used to define exactly the same random nested partition as the NDP (Trippa, 2011). The enriched DP is an *a priori* for a random probability measure on a partitioned vector (x_j, y_j). If x_j are interpreted as gene-specific tags (that are not used further), then the conditional $P_{Y|X}(\cdot \mid x_j)$ can be used as G_j. Like the NDP the enriched DP assumes a DP *a priori* for the conditional $P_{Y|X}(\cdot \mid x_j)$, adding the additional generality of allowing dependence of the base measure of this DP on x_j. Rodríguez & Ghosh (2012) construct nested partitions by ties of cluster-specific parameters. Similarly, Lee *et al.* (2013) construct a random partition of upper level units, defined by all upper level units in the same cluster sharing the same partition of lower level units. The construction is without reference to cluster-specific parameters, which in fact can vary across matching second level clusters. The latter two methods are closely related to the NDP, but do not allow the desired clustering of distributions.

Therefore, later, in the simulation study we will set up another practical and easy alternative for clustering distributions. We define a version of k-means to allow for clustering of random distributions. We achieve this by representing each distribution by a high order Jacobi polynomial (Arbel *et al*, 2015).

6.3 The *a posteriori* simulation for the marginal NDP

6.3.1 Gibbs sampling transition probabilities

We implement Markov chain Monte Carlo simulation for model (6.4). That is, we define *a posteriori* simulation for the NDP based on the marginal model on nested partitions, without resorting to truncated DP random measures simulation. We define MCMC transition probabilities for the *a posteriori* distribution under (6.5), including updates for $\boldsymbol{s}, \boldsymbol{r}$, $(\boldsymbol{\theta}^{\star}_{k\ell})$ and, consequently, K and L_k, for $k = 1, \ldots, K$.

Updating $\boldsymbol{\theta}^{\star}_{k\ell}$: As the NIG base measure $G_0(\boldsymbol{\theta}^{\star}_{k\ell})$ is conjugate to the normal kernel $p(y_{ji} \mid \boldsymbol{\theta}^{\star}_{k\ell})$, the marginal distribution of y_{ji} and complete conditional *a posteriori* distribution of $(\boldsymbol{\theta}^{\star}_{k\ell})$ and r_{ji} are available in closed form. We can therefore define Gibbs sampling transition probabilities to update $\boldsymbol{\theta}^{\star}_{k\ell}$ by draws from the complete conditional *a posteriori* distribution. For a given configuration of $\boldsymbol{s}$ and $\boldsymbol{r}$, the complete conditional *a posteriori* distribution of $\boldsymbol{\theta}^{\star}_{k\ell}$ is

$$p(\boldsymbol{\theta}^{\star}_{k\ell} \mid \ldots) = \text{NIG}(\mu_{k\ell}, \lambda_{k\ell}, a_{k\ell}, b_{k,\ell}) \tag{6.6}$$

with

$$\mu_{k\ell} = \frac{m_{k\ell}\bar{y}_{k\ell} + \lambda\mu_0}{\lambda + m_{k\ell}},\ \lambda_{k\ell} = \lambda + m_{k\ell},\ a_{k\ell} = \frac{m_{k\ell}}{2} + a,\ b_{k\ell} = b + \frac{1}{2}\left(s^2_{k\ell} + \frac{m_{k\ell}\lambda(\bar{y}_{k\ell} - \mu_0)^2}{\lambda + m_{k\ell}}\right),$$

where $\bar{y}_{k\ell} = \sum_{(j,i)\in R_{k\ell}} y_{ji}/m_{k\ell}$ and $s^2_{k\ell} = \sum_{(j,i)\in R_{k\ell}}(y_{ji} - \bar{y}_{k\ell})^2$ are cluster-specific sample means and (scaled) variances.

Updating r_{ji}: The observational cluster indicator r_{ji}, for $j = 1, \ldots, J$ and $i = 1, \ldots, I_j$, is drawn using its complete conditional *a posteriori* distribution. Let $h_0(y_{ji}) = \int \mathrm{N}(y_{ji} \mid \boldsymbol{\theta})\,\mathrm{NIG}(\boldsymbol{\theta} \mid \mu_0, \lambda, a, b)\, d\boldsymbol{\theta}$, use the superscript xx^- to represent the appropriate quantity xx with r_{ji} excluded from the sample, and let $\boldsymbol{r}_{-ji}$ denote $\boldsymbol{r}$ with r_{ji} removed. The complete conditional is

$$p(r_{ji} = \ell \mid \boldsymbol{r}_{-ji}, s_j = k, \boldsymbol{y}, \boldsymbol{\theta}^\star) \propto \begin{cases} m^-_{k\ell}\mathrm{N}(y_{ji} \mid \boldsymbol{\theta}^\star_{k\ell}), & \text{for } \ell = 1, \ldots, L^-_k \\ \beta h_0(y_{ji}), & \text{for } \ell = L^-_k + 1. \end{cases} \tag{6.7}$$

Inspection of the right hand side shows that r_{ji} is conditionally independent of $y_{j'i'}$, $(j', i') \neq (j, i)$. For later reference we note that we could replace $\boldsymbol{y}$ in the conditioning set by y_{ji}. Let $\mathrm{t}(x \mid \nu, m, s)$ denote a t-distribution with ν degrees of freedom and location and scale parameters m and s, evaluated at x. Similarly, let $\mathrm{N}(x \mid \mu, \sigma^2)$ denote a normal p.d.f. evaluated at x and similarly for $\mathrm{NIG}(x \mid \mu, \lambda, a, b)$. We find $h_0(y_{ji}) = \mathrm{t}\left(y_{ji} \mid 2a, \mu_0, \left(\frac{b(1+\lambda)}{a\lambda}\right)^{1/2}\right)$. Note that the second line in (6.7) allows to generate a new observational cluster inside k-th distributional cluster and, consequently, increment L_k by one. In that case we use (6.6) to generate a value for $\boldsymbol{\theta}^\star_{k1}$ for the new $k = L^-_k + 1$. In each step of the algorithm, empty observational clusters in the new configuration of $\boldsymbol{r}$ and their respective parameters are excluded from the model, the remaining observational clusters are relabeled and values of L_k's are recalculated.

6.3.2 Transition probabilities for distributional clusters

Updating the distributional cluster indicator s_j, for $j = 1, \ldots, J$, is more complicated since allocating $\boldsymbol{y}_j$ in a different distributional cluster also involves reallocating observational cluster memberships $\boldsymbol{r}_j$. Therefore, we have to update s_j and $\boldsymbol{r}_j$ jointly.

One possible approach is proposed in Jain & Neal (2007) who, instead of updating cluster indicators sequentially, construct a proposal by first sampling two observations and then update cluster indicators for all observations that share cluster memberships with these two observations. The proposal involves a split if both observations are in the same cluster and a merge if they are in different clusters. We propose a different strategy, using an approximation of the joint *a posteriori* for a random partition proposed in MacEachern *et al.* (1999). They use the approximation as importance sampling density in an importance sampling scheme. We use a similar partial *a posteriori* distribution here to define a suitable proposal distribution for a Metropolis-Hastings transition probability that updates s_j and $\boldsymbol{r}_j$ jointly.

Let $x = (s_j, \boldsymbol{r}_j)$ be the current configuration and let $x^{'} = (s_j^{'}, \boldsymbol{r}_j^{'})$ be a proposal configuration in a Metropolis-Hastings type transition probability. Let $\boldsymbol{r}_{-j} = (r_{j'i},\ j' \neq j,\ i = 1, \ldots, I_{j'})$. The new state $x^{'}$ is accepted with probability $\Psi(x^{'} \mid x) = min(1, A^{'})$, where

$$A^{'} = \frac{p(x^{'} \mid \boldsymbol{y}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})}{p(x \mid \boldsymbol{y}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})} \frac{q(x \mid x^{'})}{q(x^{'} \mid x)}, \tag{6.8}$$

where $p(x \mid \boldsymbol{y}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})$ is the marginal *a posteriori* probability of cluster indicators and $q(\cdot \mid \cdot)$ is the proposal distribution to generate a proposal in the Metropolis-Hastings transition probability. Both are described below.

The marginal *a posteriori.* The marginal *a posteriori* distribution of $x = (s_j, \boldsymbol{r}_j)$ is evaluated as

$$\begin{aligned} p(s_j = k, \boldsymbol{r}_j \mid \boldsymbol{y}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}) &= \frac{p(s_j = k, \boldsymbol{r}_j, \boldsymbol{y}_j \mid \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})}{p(\boldsymbol{y}_j \mid \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})} \\ &\propto p(s_j = k \mid \boldsymbol{s}_{-j}) p(\boldsymbol{r}_j \mid s_j = k, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})\, p(\boldsymbol{y}_j \mid \boldsymbol{r}_j, s_j = k, \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}), \end{aligned} \tag{6.9}$$

where the conditional *a priori* on s_j follows from (6.3) as

$$p(s_j = k \mid \boldsymbol{s}_{-j}) \propto \begin{cases} n_k^-, \text{ if } k \in \{1, \ldots, K^-\} \\ \alpha, \text{ if } k = K^- + 1 \end{cases} \tag{6.10}$$

and the conditional *a priori* for $\boldsymbol{r}_j$ is similarly derived from (6.3) as
$p(\boldsymbol{r}_j \mid s_j = k, \boldsymbol{r}_{-j}) = \prod_{i=1}^{I_j} p(r_{ji} = \ell \mid s_j = k, \boldsymbol{r}_{-j}, r_{j1}, \ldots, r_{ji-1})$ with

$$p(r_{ji} = \ell \mid s_j = k, \boldsymbol{r}_{-j}, r_{j1}, \ldots, r_{ji-1}) \propto \begin{cases} m_{k\ell}^{-(r_{ji}, \ldots, r_{jI_j})}, & \text{if } \ell \in \{1, \ldots, L_k^-\} \\ \beta, & \text{if } \ell = L_k^- + 1. \end{cases}$$

Here $m_{k\ell}^{-(r_{ji}, \ldots, r_{jI_j})}$ is the size of cluster $R_{k\ell}$ without r_{jh}, $h = i, \ldots, I_j$, and note that r_{ji} can share clusters with $r_{j'i'}$ for other genes $j' \neq j$ (which are included in $\boldsymbol{r}_{-j}$). Finally, the likelihood in (6.9) is

$$\begin{aligned} p(\boldsymbol{y}_j \mid \boldsymbol{r}_j, s_j = k, \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}) &= \int_{\boldsymbol{\Theta}} p(\boldsymbol{y}_j \mid \boldsymbol{r}_j, s_j = k, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \boldsymbol{y}_{-j}, \boldsymbol{r}, \boldsymbol{s}) d\boldsymbol{\theta} \\ &= \prod_{\ell=1}^{L_k} \int_{\boldsymbol{\Theta}} \left(\prod_{i: r_{ji} = \ell} p(y_{ji} \mid \boldsymbol{\theta}_{k\ell}^{\star}) \right) p(\boldsymbol{\theta}_{k\ell}^{\star} \mid \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}) d\boldsymbol{\theta} \end{aligned}$$

and $p(\boldsymbol{\theta}_{k\ell}^{\star} \mid \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j})$ is the complete conditional *a posteriori* distribution for $\boldsymbol{\theta}_{k\ell}^{\star}$ as in (6.6), but conditional on only $\boldsymbol{y}_{-j}$, $\boldsymbol{r}_{-j}$ and $\boldsymbol{s}_{-j}$. Under the normal sampling model and the conjugate baseline probability measure,

$$\int_{\boldsymbol{\Theta}} \left(\prod_{i:r_{ji}=\ell} p(y_{ji} \mid \boldsymbol{\theta}^{\star}_{k\ell}) \right) p(\boldsymbol{\theta}^{\star}_{k\ell} \mid \boldsymbol{y}_{-j}, \boldsymbol{r}_{-j}, \boldsymbol{s}_{-j}) d\boldsymbol{\theta} = (2\pi)^{-\frac{(m_{k\ell})_j}{2}} \left(\frac{\lambda^{-}_{\star}}{(m_{k\ell})_j + \lambda^{-}_{\star}} \right)^{1/2}$$

$$\times \frac{\Gamma\left(\frac{(m_{k\ell})_j}{2} + a^{-}_{\star}\right)}{\Gamma\left(a^{-}_{\star}\right)} \left(b^{-}_{\star}\right)^{a^{-}_{\star}} \left(b^{-}_{\star} + \frac{(s_{k\ell})^2_j}{2} + \frac{\lambda^{-}_{*}(m_{k\ell})_j \left((\bar{y}_{k\ell})_j - \mu^{-}_{\star}\right)^2}{2\left((m_{k\ell})_j + \lambda^{-}_{*}\right)} \right)^{-\left(\frac{(m_{k\ell})_j}{2} + a^{-}_{\star}\right)},$$

where $\mu^{-}_{\star}$, $\lambda^{-}_{\star}$, $a^{-}_{\star}$ and $b^{-}_{\star}$ are the parameters of the NIG distribution given by (6.6) considering $\boldsymbol{y}_{-j}$, $\boldsymbol{r}_{-j}$ and $\boldsymbol{s}_{-j}$ and $(m_{k\ell})_j$, $(s^2_{k\ell})_j$ and $(\bar{y}_{k\ell})_j$ are calculated only using $\boldsymbol{y}_j$. Finally, note that the denominator of (6.9) is the same for configuration x and $x^{'}$ and it cancels out in (6.8).

Proposal distribution. The proposal distribution $q(x' \mid x)$ is constructively defined in the following two steps. In words, for s_j we use the conditional *a priori*. For $\boldsymbol{r}_j$ we construct a proposal distribution that is similar to the importance sampling density in MacEachern *et al.* (1999).

1. sample $s^{'}_j$ from its *a priori* distribution given $\boldsymbol{s}_{-j}$ defined by (6.10). Observe that the second line of (6.10) allows to start a new distributional cluster and, consequently, increase the current value of K by one;

2. sample $r^{'}_{ji}$, for $i = 1, \ldots, I_J$, from

$$p(r^{'}_{ji} = \ell \mid \boldsymbol{r}_{-j}, r^{'}_{j1}, \ldots, r^{'}_{ji-1}, s^{'}_j = k, y_{ji}, \boldsymbol{\theta}^{\star})$$

   given by (6.7) and where $m^{-}_{k\ell}$ is substituted for $m^{-(r_{ji},\ldots,r_{jI_j})}_{k\ell}$ since the last $I_j - i + 1$ observations have not been reallocated yet. Note that if $s^{'}_j = K^{-} + 1$, $r^{'}_{j1} = 1$ necessarily and we use the second line in (6.7) to sample $r^{'}_{j2}, \ldots, r^{'}_{jI_j}$. That is we replace the $\mathrm{N}(y_{ji} \mid \boldsymbol{\theta}^{\star}_{k\ell})$ density in the first line by $h_0(y_{ji})$.

In summary, the proposal distribution $q(x^{'} \mid x)$ is:

$$q(x^{'} \mid x) = p(s^{'}_j = k \mid \boldsymbol{s}_{-j}) \prod_{i=1}^{I_j} p(r^{'}_{ji} = \ell \mid \boldsymbol{r}_{-j}, r^{'}_{j1}, \ldots, r^{'}_{ji-1}, s^{'}_j = k, y_{ji}, \boldsymbol{\theta}^{\star}) \tag{6.11}$$

and the transition function $q(x \mid x^{'})$ is defined in a similar way.

After evaluating the acceptance of the proposal configuration $x^{'}$, we drop empty observational and distributional clusters, relabel remaining clusters, recalculate K and values of L_k and update $\boldsymbol{\theta}^{\star}_{k\ell}$, for $k = 1, \ldots, K$ and $\ell = 1, \ldots, L_k$, from their complete conditional *a posteriori* distributions (6.6).

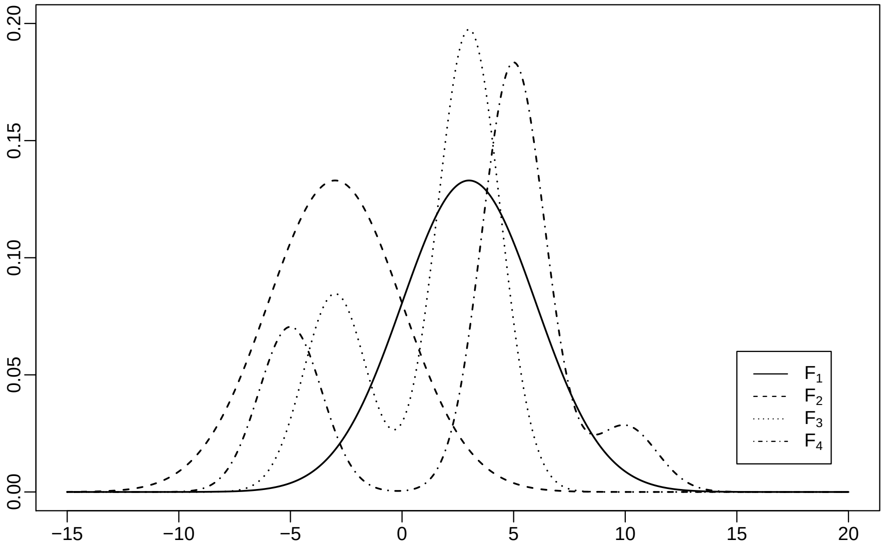


Figure 6.1: Simulation truth.

6.4 **Simulation**

6.4.1 Marginal NDP

We generate $J = 40$ samples of size $I_j = 100$, for $j = 1, \ldots, 40$. Each sample is obtained from one of $K = 4$ mixtures of normal distributions shown in Table 6.1 and plotted in Figure 6.1. These distributions have been chosen to mimic the data in the application to DMR genes (see the next section).

Table 6.1: Parameters of the true distributions used to simulate the data set, where w represents the component weight.

Distribution	Component 1			Component 2			Component 3		
	w	μ	σ^2	w	μ	σ^2	w	μ	σ^2
F_1	1	-3	9						
F_2	1	3	9						
F_3	0.3	-3	2	0.7	3	2			
F_4	0.25	-5	2	0.65	5	2	0.10	10	2

The total mass parameters α and β are both fixed to 1 and the hyperparameters of the NIG baseline measure are $\mu_0 = 0$, $\lambda = 0.01$, $a = 3$ and $b = 5$, implying that, an *a priori*i, $E(\sigma^2) = 2.5$, $Var(\sigma^2) = 6.5$, $E(\mu \mid \sigma^2) = 0$ and $Var(\mu \mid \sigma^2) = 100\sigma^2$. We use the approach described in Section 6.3 to generate a *a posteriori* Monte Carlo sample. We run two MCMC chains with different starting point and 11,000 iterations, discarding the first 4000 iterations and then thinning out to save one in every 10 iterations. The first chain is initialized with $K = 1$ and $L_1 = 1$ distributional and observational cluster, respectively, and the second chain is initialized with $K = 8$ distributional clusters and $L_k = 2$ observational clusters for all k. The

R code for clustering the simulated data set is available in the appendices of this chapter.

Figure 6.2 (A) shows the observations ordered by the true clusters and Figure 6.2 (B) shows *a posteriori* co-clustering probabilities $\bar{p}_{ij} = p(s_i = s_j \mid data)$ for each of the $J(J-1)/2$ pairs of samples. Black and white colors represent $\bar{p}_{ij}$ close to 1 and 0, respectively. The *a posteriori* probabilities are evaluated as ergodic averages over the MCMC output. We observe that under the simulated conditions with well separated true clusters *a posteriori* inference distinguishes clearly between distributions and correctly estimates the number of distributional clusters and identifies the clusters. The *a posteriori* inference correctly discriminates between multimodal and unimodal distributions with matching marginal mean and variance.

Inference under the NDP model includes estimation of the cluster-specific distributions, $\hat{F}_k$, $k = 1, \ldots, 4$. Figure 6.3 shows the estimated distribution for each cluster. The *a posteriori* estimates closely recover the simulation truth.

Here and in other plots we use the following point estimate for the *a posteriori* random partition, and the following post-processing to address *label switching.* We report two observations j_1 and j_2 in the same cluster if $\bar{p}_{j_1 j_2} > 0.5$, that is, if they are located in the same cluster in more than half of the MCMC sample. Alternatively one could use any other point estimate, for example the estimate suggested in Dahl (2006).

To mitigate problems related to *label switching* and allow for a meaningful report of $\hat{F}_k = E(F_k \mid \boldsymbol{y})$, $k = 1, \ldots, K$, we impose an order constraint on the observational and distributional clusters, following a suggestion in Richardson & Green (1997). We order observational clusters (within the k-th distributional cluster) by imposing $\mu^\star_{k1} < \mu^\star_{k2} < \ldots < \mu^\star_{kL_k}$. We order distributional clusters by imposing $\bar{\mu}_1 < \bar{\mu}_2 < \ldots < \bar{\mu}_K$, where $\bar{\mu}_k = \frac{\sum_{\ell=1}^{L_k} m_{k\ell}\mu^\star_{k\ell}}{\sum_{\ell=1}^{L_k} m_{k\ell}}$ is the weighted average of unique values $\mu^\star_{k\ell}$ in k-th distributional cluster, $k = 1, \ldots, K$.

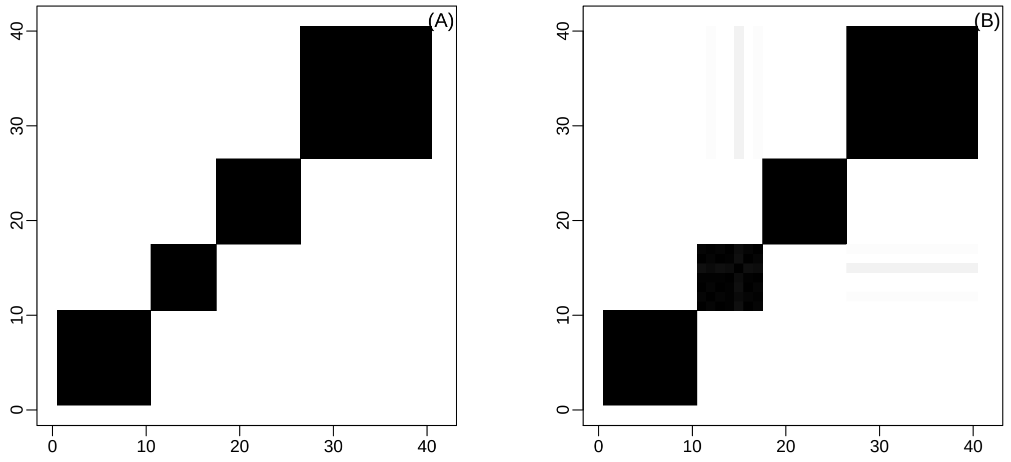
6.4.2 Clustering distributions by k-means

For comparison we also carry out clustering with the popular k-means algorithm (Jain, 2010; Kulis & Jordan, 2012). The k-means algorithm clusters data $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$. The algorithm is deterministic, based on minimizing the k-means criterion. Let $\rho = \{S_1, \ldots, S_K\}$ denote a partition of $\{1, \ldots, n\}$. The algorithm finds the partition which minimizes the objective function

$$\widehat{\rho} = \arg\min_{\rho} \sum_{k=1}^{K} \sum_{j \in S_k} \|\boldsymbol{y}_j - \boldsymbol{\mu}^\star_k\|^2 + \delta K, \tag{6.12}$$

with $\boldsymbol{\mu}^\star_k = \frac{\sum_{j \in S_k} \boldsymbol{y}_j}{|S_k|}$. The δK term rewards parsimony by penalizing additional clusters.

There is a problem in using k-means for the desired clustering of distributions. The k-means criterion is only meaningful when all $\boldsymbol{y}_j$ are of matching length. More importantly, k-means is meant to cluster p-dimensional response vectors. It is not constructed to cluster distributions F_j.



(A) truth (B) $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$.

Figure 6.2: Panel (A) shows the simulation truth for cluster membership by plotting $I(s_i = s_j)$ (black for equality). Panel (B) plots the *a posteriori* probabilities $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$ (black for $\bar{p}_{ij} = 1$, white for 0).

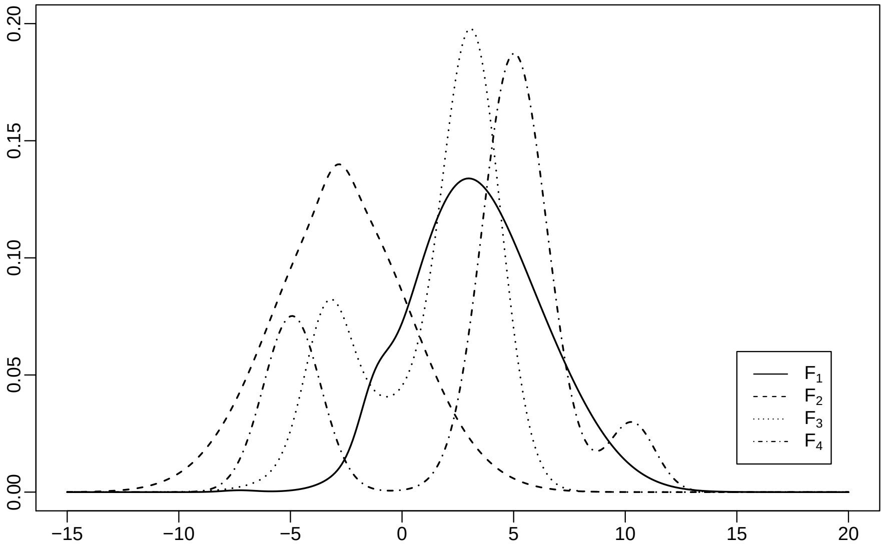


Figure 6.3: Estimated $\widehat{F}_k = E(F_k \mid \boldsymbol{y})$. Compare with Figure 6.1.

We introduce a variation of k-means to adapt the algorithm for the clustering of a set of distributions by adding a pre-processing step before minimizing (6.12). The step is motivated by results in Arbel *et al.* (2015) who show examples where a distribution is very well fit by Jacobi polynomials with the first seven moments. See, for example, Provost (2005) for a discussion of Jacobi polynomials. Let $\widehat{F}_j$ denote the empirical distribution of y_{ji}, $i = 1, \ldots, I_j$, for each gene j. We summarize $\boldsymbol{y}_j$ by the first seven moments $\boldsymbol{M}_j = (M_{j1}, \ldots, M_{j7})$ of $\widehat{F}_j$, where $M_{jr} = \frac{\sum_{j=1}^{I_j} y_{ji}^r}{I_j}$ is the r-th empirical moment. We define transformed data $\tilde{\boldsymbol{y}}_j = \boldsymbol{M}_j$ and then carry out k-means for standardized $\tilde{\boldsymbol{y}}_j$, $j = 1, \ldots, J$. We refer to this modified k-means algorithm as "distributional k-means."

Let SSE denote the error sum of squares and SST the total sum of squares. We calibrate the penalty δ in the k-means criterion by evaluating SSE/SST over a grid of δ values and selecting the largest δ (consequently smallest K) before SSE/SST starts to sharply rise. Algorithm 1 summarizes the distributional k-means algorithm.

Algorithm 1 Distributional k-means algorithm

```
Input: $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_J$ as histograms.
Output: Partition $\{S_1, \ldots, S_K\}$, including the number of clusters $K$

1. For each histogram $\boldsymbol{y}_j$, $j = 1, \ldots, J$, calculate $\tilde{\boldsymbol{y}}_j$.
2. Over a grid on $\delta$.
    (a) Find the partition which minimizes (6.12) for $\tilde{\boldsymbol{y}}$;
    (b) Calculate SSE/SST.
3. Select the largest $\delta$ before SSE/SST starts to sharply rise.
4. Report the partition which minimizes (6.12).
```

We carry out distributional k-means for the simulation data from Section 6.4.1.

Table 6.2 shows the misclassification table for the simulation truth and the estimated partition of $\tilde{\boldsymbol{y}}$ (reporting the misclassification rate for the best match of cluster indices). We observe that the four distributional clusters are well identified by the distributional k-means algorithm. Only one histogram that was truly generated by sampling from F_4 was wrongly classified as coming from F_2 and another histogram was classified as a singleton cluster.

Table 6.2: Misclassification table of true and estimated clusters of $\tilde{\boldsymbol{y}}$.

True clusters	Estimated clusters				
	1	2	3	4	5
1	10	0	0	0	0
2	0	7	0	0	0
3	0	0	9	0	0
4	0	1	0	12	1

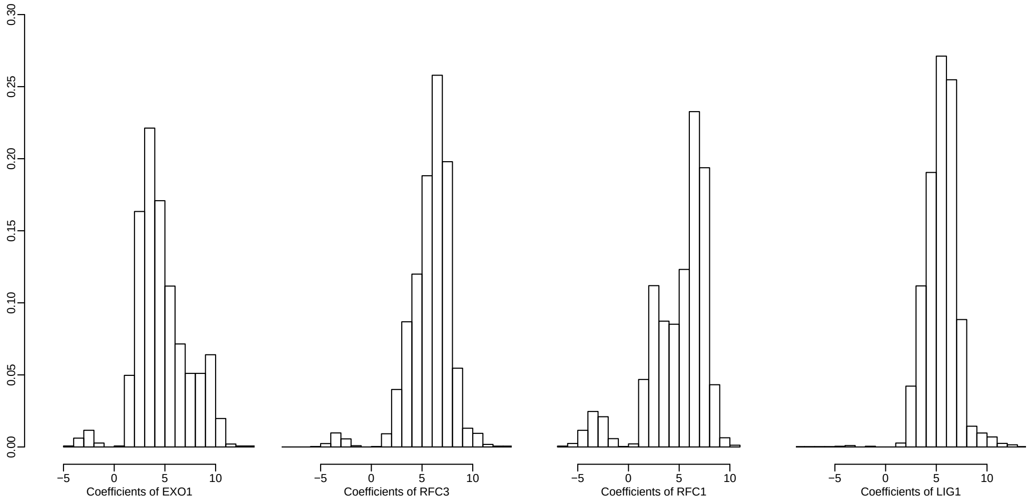


Figure 6.4: Histograms of GE-GE interactions for four genes. The inference goal is to group all J genes into clusters with similar distributions of GE-GE interactions.

6.5 Clustering DMR genes

6.5.1 Data

DNA mismatch repair (DMR) is a system for recognizing and repairing erroneous insertion, deletion, and mis-incorporation of bases that can arise during DNA replication and recombination, as well as repairing some forms of DNA damage. Defects in mismatch repair can result in microsatellite instability which is typical for most human cancers. We selected $J = 23$ genes from the Zodiac database (Zhu *et al*, 2015) which are known to be associated with DMR. Zodiac output reports the auto-logistic coefficients for GE-GE interactions between these J genes and all other genes in the data base. Since the inference goal is to cluster DMR genes according to their significant GE-GE interaction pattern with others genes, we therefore focus only significant coefficients. Significant coefficients are identified by a threshold on false discovery rate (FDR) $<$ 0.1. See, for example, Mitra *et al.* (2013) for more details. After FDR thresholding, this leaves between 222 and 4,393 significant coefficients for each gene, $j = 1, \ldots, J$, with a total of $m = 43, 555$ significant coefficients. Figure 6.4 shows histograms of coefficients for four arbitrarily selected genes.

6.5.2 Results

We run two MCMC chains with different starting points and 11,000 iterations, discard the first 4,000 iterations and thin out to save only every 10-th iteration. The first chain is initialized with $K = 1$ distributional cluster and the second chain is initialized with $K = 23$ distributional clusters, that is, each gene in a singleton cluster. The mass parameters α and β are both fixed to 1 and the hyperparameters of the NIG baseline measure are fixed at $\mu_0 = 0$, $\lambda = 0.01$, $a = 3$

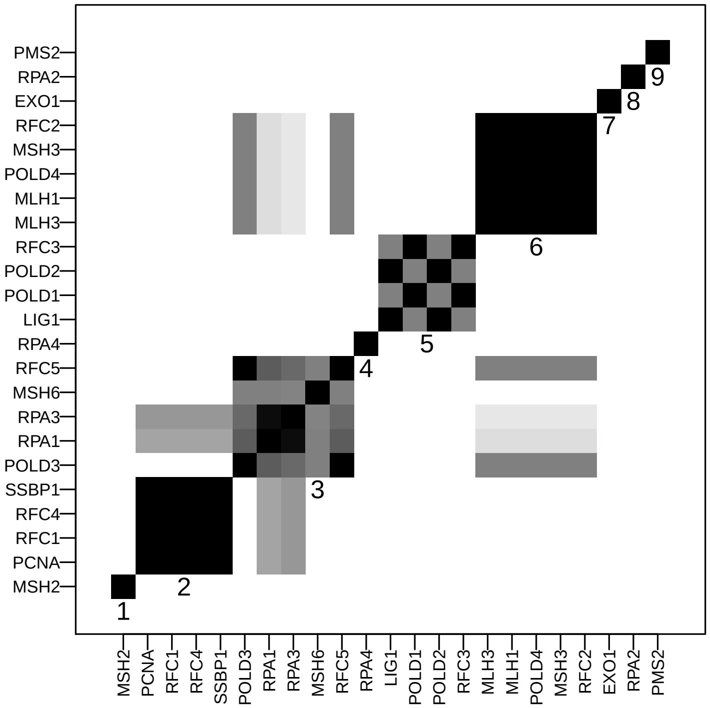


Figure 6.5: The *a posteriori* co-clustering probability $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$ for DMR genes. Numbers under each cluster in the diagonal represent clusters' label.

and $b = 5$.

Figure 6.5 shows *a posteriori* co-clustering probabilities $\bar{p}_{ij} = p(s_i = s_j \mid \boldsymbol{y})$ for each of the 256 pairs of genes. That is, for each pair $j_1 < j_2$, the plot shows the *a posteriori* probability that genes j_1 and j_2 share the same distribution of auto-logistic coefficients. We clearly observe $K = 9$ clusters (the diagonal), five of them are singleton clusters (clusters 1, 4, 7, 8 and 9) and the remaining four clusters contain at least four genes. The distribution of coefficients for the genes POLD3 and RFC5, although it is closer to distributions of cluster 3, has also similarities with distributions of cluster 6 and the distributions of RPA3 and RPA1 genes are not too different from cluster 2.

Figure 6.6 shows the estimated distributions $\widehat{F}_k = E(F_k \mid \boldsymbol{y}) = \widehat{p}(y_{J+1,1} \mid \boldsymbol{y}, s_{J+1} = k)$ that can be approximated by Monte Carlo estimator

$$\frac{1}{D}\sum_{d=1}^{D}\left(\frac{\beta}{\beta + m_{k\ell}^{(d)}}\int_{\boldsymbol{\Theta}} p(y_{J+1,1} \mid \boldsymbol{\theta})G_0(d\boldsymbol{\theta}) + \sum_{\ell=1}^{L_k^{(d)}} \frac{m_{k\ell}^{(d)}}{\beta + m_{k\ell}^{(d)}} p\left(y_{J+1,1} \mid {\boldsymbol{\theta}_{k\ell}^{\star}}^{(d)}\right)\right),$$

where D is the number of iterations in the Markov chain. The thick lines show the estimated cluster-specific distributions $\widehat{F}_k = E(F_k \mid \boldsymbol{y})$, for $k = 1, \ldots, 9$ (4, 8 and 9 are not shown). For comparison, thinner lines show kernel density estimates of the sampling distribution for

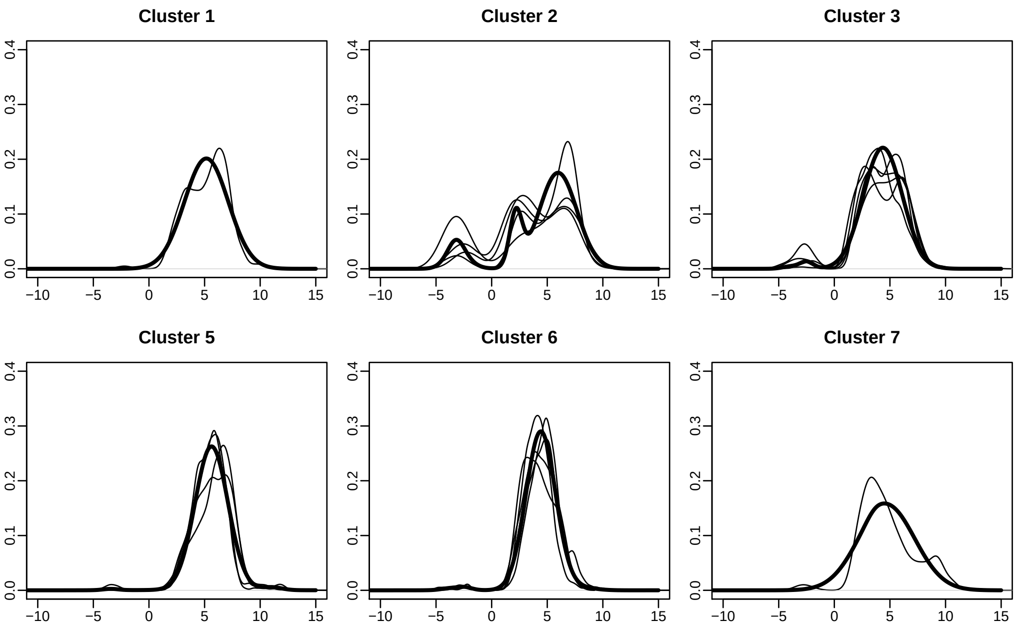


Figure 6.6: Estimated cluster-specific distributions $\widehat{F}_k = E(F_k \mid \boldsymbol{y})$.

each gene. Cluster 2 is composed by the genes PCNA, RFC1, RFC4 and SSBP1 and cluster 3 contains genes POLD3, RPA1, RPA3, MSH6 and RFC5, based on thresholding co-clustering probabilities using a cutoff value of 0.50. The distribution associated with these both clusters are clearly multimodal and they include genes with negative as well as positive GE-GE interaction with other genes. Cluster 5 is composed of genes LIG1, POLD1, POLD2 and RFC3 and cluster 6 includes MLH3, MLH1, POLD4, MSH3 and RFC2. Both clusters are characterized by positive unimodal distributions. That is, genes in these clusters have only positive GE-GE interactions with other genes. Clusters 1 and 7 are singleton and the range of the coefficients distribution is wider than in other clusters. Finally, also clusters 4, 8 and 9 are singletons, composed of the genes RPA4, RPA2 and PMS2, respectively. The estimated distributions $\widehat{F}_k$, $k = 4, 8, 9$ (not shown) put most probability for lower GE-GE interaction coefficients, much lower than the other clusters, and are therefore of less interest.

For comparison, we cluster the same $J = 23$ DMR genes using the earlier defined distributional k-means algorithm. We add the frequency f_{j0} of nonsignificant auto-logistic coefficients in the pre-processed data $\tilde{\boldsymbol{y}}_j = (\boldsymbol{M}_j, \log f_{j0})$ and implement k-means with the standardized empirical moments and the logarithm of the frequency of nonsignificant auto-logistic coefficients. Inspecting SSE/SST over a grid on δ (and the implied K) we find $K = 5$. The first cluster contains 11 genes: MLH3, MLH1, PCNA, POLD3, POLD4, RPA1, RPA3, MSH6, MSH3, RFC2 and RFC5. This is a combination of NDP clusters 3 and 6. Cluster 2 is composed by 6 genes: MSH2, LIG1, POLD1, POLD2, RFC1 and RFC3 which is basically NDP cluster $k = 5$ with the highest GE-GE coefficients.

In summary, similar clusters of coefficient distributions are identified by NDP and k-means.

However, the NDP model identifies minor differences in the features of the distributions and splits them in additional clusters. More importantly, inference under the proposed marginal NDP model includes estimated distributions $\widehat{F}_k$, including a full probabilistic description $p(F_k \mid \boldsymbol{y})$, while a deterministic algorithm like k-mean only delivers a point estimate of the partition, without any notion of uncertainties and without estimated distributions (beyond an ad-hoc kernel density estimate).

6.6 Discussion

We introduced the marginal nested Dirichlet process to cluster histograms and identify groups of genes that have similar patterns of genetic interaction with other genes and propose a MCMC algorithm to estimate it. As a nonparametric model, the NDP is a flexible model and the groups are clustered by their entire distribution, rather than by particular features of the distribution. In addition to clustering the samples, NDP also includes inference on the cluster-specific distributions F_k, including a full description of related uncertainties in the form of $p(F_k \mid \boldsymbol{y})$.

In the small simulation study proposed inference showed good performance in distinguishing multimodal from unimodal distributions even with similar means and variances. The NDP clusters of genes associated with DMR have clearly distinct features. Important for the application, we can easily identify the group of genes that have the highest GE-GE interactions with other genes.

Among the limitations is the computation cost to run the exact MCMC algorithm for clustering hundreds or thousands of genes. Relatedly, the nested DP in the lower level of the NDP model which creates the observational clusters inside the distributional clusters is not exploited for the desired distributional clustering that is of interest in the motivating application here. It could be useful in other applications.

6.7 Appendices

6.7.1 R codes to carry out marginal NDP in simulated data set

In this section, we show R codes for clustering the simulated data set.

```
#
#########################
# simulate the data set from different normal distributions
#########################
#
rDiscreta<-function(p){
 u<-runif(1)
 P<-cumsum(p)
 val<-sum(P<u)+1
 return(val)}
#
K<-4 # number of different distributions (clusters)
J<-40 # number of elements in the sample
Ij<-rep(100,J) # number of replication for each element
#
meanvec1<-c(-3,3)
sigmavec1<-c(2,2)
p1<-c(0.30,0.70)
meanvec2<-c(3)
sigmavec2<-c(9)
p2<-1
meanvec3<-c(-3)
sigmavec3<-c(9)
p3<-1
meanvec4<-c(-5,5,10)
sigmavec4<-c(2,2,2)
p4<-c(0.25,0.65,0.10)
#
id<-numeric(sum(Ij)) # specifies from each distribution each element comes from
iddentr<-numeric(sum(Ij))
Y<-numeric(sum(Ij)) # observations
#
set.seed(1034)
cont<-1
for (j in 1:40){
 dist<-rDiscreta(rep(1/K,K))
 for (i in 1:Ij[j]){
  id[cont]<-dist
  if (id[cont]==1){
   iddentr[cont]<-rDiscreta(p1)
   Y[cont]<-rnorm(1,meanvec1[iddentr[cont]],sqrt(sigmavec1[iddentr[cont]]))}
  if (id[cont]==2){
   iddentr[cont]<-rDiscreta(p2)
   Y[cont]<-rnorm(1,meanvec2[iddentr[cont]],sqrt(sigmavec2[iddentr[cont]]))}
  if (id[cont]==3){
   iddentr[cont]<-rDiscreta(p3)
   Y[cont]<-rnorm(1,meanvec3[iddentr[cont]],sqrt(sigmavec3[iddentr[cont]]))}
  if (id[cont]==4){
   iddentr[cont]<-rDiscreta(p4)
   Y[cont]<-rnorm(1,meanvec4[iddentr[cont]],sqrt(sigmavec4[iddentr[cont]]))}
  cont<-cont+1}}
#
tauj_verd<-id # specifies from each distribution each element comes from in level
of elements and their replication
Sj_verd<-numeric(J) # specifies from each distribution each element comes from
in level of elements
for (i in 1:J){
 Sj_verd[i]<-as.numeric(as.character(data.frame(table(tauj_verd[
(cumsum(Ij)[i]-Ij[i]+1):cumsum(Ij)[i]]))[,1]))}
#
id<-rep(1,Ij[1])
for (j in 2:J) id<-c(id,rep(j,Ij[j]))
dataset<-matrix(c(id,Y),ncol=2) # final data set
#
#########################
# useful functions
#########################
#
############
# sample from a Normal Inverse gamma distribution (mu, lambda, alpha, beta)
############
rinvgamma<-function(media.mi,lambda,alpha,beta){
 n<-length(media.mi)
 sigma2<-1/(rgamma(n,alpha,beta))
 mu<-rnorm(n,media.mi,sqrt(sigma2/lambda))
 sample<-cbind(mu,sigma2)
 return(sample)}
#
############
# sample from a Multinomial distribituion (p)
############
rDiscreta<-function(p){
 u<-runif(1)
 P<-cumsum(p)
 val<-sum(P<u)+1
 return(val)}
#
############
# calculate the density function of a t distribution (yij, ni, mu, sigma)
############
dstudentt<-function(yij,ni,mu,desvio){
 dens<-(gamma((ni+1)/2)/gamma(ni/2))*((1+(1/ni)*((yij-mu)/desvio)**2)**
(-(ni+1)/2))*(1/(sqrt(pi*ni)*desvio))
 return(dens)}
#
############
# calculate the marginalized likelihood of Yj given (Xij ,tauj) and specific l
############
dmarglikeli<-function(Yjl,lambda,agam,bgam,mu){
 mlkj<-length(Yjl)
 media<-mean(Yjl)
 varia<-var(Yjl)*(mlkj-1)
 if (is.na(varia)) varia<-0
 logdens<-(lgamma(mlkj/2+agam)-lgamma(agam))+((-mlkj/2)*log(2*pi))+
(0.5*(log(lambda)-log(mlkj+lambda)))+(agam*log(bgam))+((-(mlkj/2+agam))*
log(bgam+(varia/2)+((lambda*mlkj*((media-mu)**2))/(2*(mlkj+lambda)))))
 return(logdens)}
#
############
# sample from conditional a posteriori distribution of theta for all k and l
############
posteriori.theta<-function(dataset, Zij, tauj, lambda, agam, bgam,mu){
 mlk<-aggregate(rep(1,nrow(dataset)), by = list(Zij, tauj), FUN = "sum")[,3]
 Ymeanlk<-aggregate(dataset[,2], by = list(Zij, tauj), FUN = "mean")[,3]
 Yvarlk<-aggregate(dataset[,2], by = list(Zij, tauj), FUN = "var")[,3]*(mlk-1)
 mupost<-((mlk*Ymeanlk)+(lambda*mu))/(lambda+mlk)
 lambdapost<-lambda+mlk
 agampost<-agam+(mlk/2)
 Yvarlk[is.na(Yvarlk)] <- 0
 bgampost<-bgam+(Yvarlk+((mlk*lambda*(Ymeanlk-mu)**2)/(lambda+mlk)))/2
 theta.samp<-rinvgamma(mupost,lambdapost,agampost,bgampost)
 return(theta.samp)}
```

```
#
############
# sample from conditional a posteriori distribution of Zij
############
posteriori.Zij<-function(dataset, Zij, tauj, K, theta.samp, ni, mu, desvio, Lk,
beta){
 for (k in 1:K){
  observ<-which(tauj==k)
  Yclusterk<-dataset[observ,2]
  Zclusterk<-Zij[observ]
  cont<-1
  for (i in observ){
   probZij<-numeric(Lk[k])
   contmlk<-Zclusterk[-cont]
   l<-seq(1:Lk[k])
   mlk<-rep(0,Lk[k])
   need<-data.frame(table(contmlk[which(contmlk<=Lk[k])]))
   categr<-as.numeric(as.character(need[,1]))
   mlk[categr]<-as.numeric(need[,2])
   probZij<-mlk*dnorm(Yclusterk[cont],theta.samp[(cumsum(Lk)[k]-Lk[k]+l),1],
sqrt(theta.samp[(cumsum(Lk)[k]-Lk[k]+l),2]))
   probZij<-c(probZij,beta*dstudentt(Yclusterk[cont],ni,mu,desvio))
   probZij<-probZij/sum(probZij)
   Zij[i]<-rDiscreta(probZij)
   Zclusterk<-Zij[observ]
   cont<-cont+1}
  while (length(table(Zclusterk))<max(Zclusterk)){ # exclude empty clusters
   categr<-as.numeric(as.character(data.frame(table(Zclusterk))[,1]))
   categd<-seq(1:length(table(Zclusterk)))
   dif<-which(categr!=categd)
   for (i in 1:length(Zclusterk)) if (Zclusterk[i]>dif[1])
Zclusterk[i]<-Zclusterk[i]-1}
  Zij[observ]<-Zclusterk}
 return(Zij)}
#
############
# sample from conditional a posteriori distribution of Zij for just one j
############
```

```
posteriori.Zij2<-function(dataset, Zij, tauj, K, theta.samp, ni, mu, desvio,
Lk, beta, j, Sj){
 k<-Sj[j]
 observ<-which(tauj==k)
 Yclusterk<-dataset[observ,2]
 Zclusterk<-Zij[observ]
 cont<-1
 for (i in observ){
  probZij<-numeric(Lk[k])
  contmlk<-Zclusterk[-cont]
  l<-seq(1:Lk[k])
  mlk<-rep(0,Lk[k])
  need<-data.frame(table(contmlk[which(contmlk<=Lk[k])]))
  categr<-as.numeric(as.character(need[,1]))
  mlk[categr]<-as.numeric(need[,2])
  probZij<-mlk*dnorm(Yclusterk[cont],theta.samp[(cumsum(Lk)[k]-Lk[k]+l),1],
sqrt(theta.samp[(cumsum(Lk)[k]-Lk[k]+l),2]))
  probZij<-c(probZij,beta*dstudentt(Yclusterk[cont],ni,mu,desvio))
  probZij<-probZij/sum(probZij)
  Zij[i]<-rDiscreta(probZij)
  Zclusterk<-Zij[observ]
  cont<-cont+1}
 while (length(table(Zclusterk))<max(Zclusterk)){ # exclude empty clusters
  categr<-as.numeric(as.character(data.frame(table(Zclusterk))[,1]))
  categd<-seq(1:length(table(Zclusterk)))
  dif<-which(categr!=categd)
  for (i in 1:length(Zclusterk)) if (Zclusterk[i]>dif[1])
 Zclusterk[i]<-Zclusterk[i]-1}
 Zij[observ]<-Zclusterk
 return(Zij)}
#
############
# build a proposal of new cluster for each Sj and realocate its Zij
############
sample.tau.xi<-function(dataset, Ij, Sj, Zij, tauj, K, Lk, theta.samp, j, alpha,
beta, ni, mu, desvio){
 nk<-numeric(K)
 for (k in 1:K) nk[k]<-sum(Sj[-j]==k)
 observ<-which(dataset[,1]==j)
 Yclusterj<-dataset[observ,2]
 Zclusterj<-Zij[observ]
 nk[Sj[j]]<-0 # we force Sj candidate to be different of the current value
 nk<-c(nk,alpha)
 probSj<-nk/sum(nk)
 Sprop<-rDiscreta(probSj)
 Sold<-Sj[j]
 Zprop<-numeric(Ij[j])
 Zold<-numeric(Ij[j])
 lprior<-0
 lfunctrans<-0
 lpriorold<-0
 lfunctransold<-0
 taujcand<-tauj
 taujcand[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]<-rep(K+1,Ij[j])
 nj<-sum(Sj[-j]==Sold)
 if (Sprop <= K){
  for (i in 1:Ij[j]){
   l<-1:Lk[Sprop]
   Zvector<-c(Zij[which(tauj==Sprop)],Zprop[which(Zprop<=Lk[Sprop] & Zprop>0)])
   mlk<-table(Zvector)
   probZij<-mlk*dnorm(Yclusterj[i],theta.samp[(cumsum(Lk)[Sprop]-
Lk[Sprop]+l),1],sqrt(theta.samp[(cumsum(Lk)[Sprop]-Lk[Sprop]+l),2]))
   if (length(which(Zprop>Lk[Sprop]))>0){
    mlk2<-table(Zprop[which(Zprop>Lk[Sprop])])
    probZij2<-mlk2*dstudentt(Yclusterj[i],ni,mu,desvio)
    mlk<-c(mlk,mlk2)
    probZij<-c(probZij,probZij2)}
   mlk<-c(mlk,beta)
   probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,desvio))
   probZij<-probZij/sum(probZij)
   mlk<-mlk/sum(mlk)
   Zprop[i]<-rDiscreta(probZij)
   lprior<-lprior+log(mlk[Zprop[i]])
   lfunctrans<-lfunctrans+log(probZij[Zprop[i]])
   if (nj > 0){
    l<-1:Lk[Sold]
    Zvector<-c(Zij[which(taujcand==Sold)],Zold[which(Zold<=Lk[Sold] & Zold>0)])
    mlk<-rep(0,Lk[Sold])
    need<-data.frame(table(Zvector))
    categr<-as.numeric(as.character(need[,1]))
    mlk[categr]<-as.numeric(need[,2])
    probZij<-mlk*dnorm(Yclusterj[i],theta.samp[(cumsum(Lk)[Sold]-
Lk[Sold]+1),1],sqrt(theta.samp[(cumsum(Lk)[Sold]-Lk[Sold]+1),2]))
    if (length(which(mlk==0))==0){
     mlk<-c(mlk,beta)
     probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,desvio))}
else {
     empty<-which(mlk==0)
     if (mlk[Zclusterj[i]]==0){
      mlk[Zclusterj[i]]<-beta
      probZij[Zclusterj[i]]<-mlk[Zclusterj[i]]*dstudentt(Yclusterj[i],ni,mu,
desvio)} else {
      mlk[empty[1]]<-beta
      probZij[empty[1]]<-mlk[empty[1]]*dstudentt(Yclusterj[i],ni,mu,desvio)}}
    probZij<-probZij/sum(probZij)
    mlk<-mlk/sum(mlk)
    Zold[i]<-Zclusterj[i]
    lpriorold<-lpriorold+log(mlk[Zclusterj[i]])
    lfunctransold<-lfunctransold+log(probZij[Zclusterj[i]])} else {
    Zold[1]<-Zclusterj[1]
    Zvector<-c(Zij[which(taujcand==Sold)],Zold[which(Zold <= Lk[Sold] & Zold>0)])
    mlk<-rep(0,Lk[Sold])
    need<-data.frame(table(Zvector))
    categr<-as.numeric(as.character(need[,1]))
    if(length(categr)>0) mlk[categr]<-as.numeric(need[,2])
    probZij<-mlk*dstudentt(Yclusterj[i],ni,mu,desvio)
    if (length(which(mlk==0))==0){
     mlk<-c(mlk,beta)
     probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,
desvio))} else {
     empty<-which(mlk==0)
     if (mlk[Zclusterj[i]]==0){
      mlk[Zclusterj[i]]<-beta
      probZij[Zclusterj[i]]<-mlk[Zclusterj[i]]*dstudentt(Yclusterj[i],ni,mu,
desvio)} else {
      mlk[empty[1]]<-beta
      probZij[empty[1]]<-mlk[empty[1]]*dstudentt(Yclusterj[i],ni,mu,desvio)}}
    probZij<-probZij/sum(probZij)
    mlk<-mlk/sum(mlk)
    Zold[i]<-Zclusterj[i]
    lpriorold<-lpriorold+log(mlk[Zclusterj[i]])
    lfunctransold<-lfunctransold+log(probZij[Zclusterj[i]])}}}
 if (Sprop > K){
  Zprop[1]<-1
  for (i in 2:Ij[j]){
   mlk<-table(Zprop[which(Zprop>0)])
   probZij<-mlk*dstudentt(Yclusterj[i],ni,mu,desvio)
   mlk<-c(mlk,beta)
   probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,desvio))
   probZij<-probZij/sum(probZij)
   mlk<-mlk/sum(mlk)
   Zprop[i]<-rDiscreta(probZij)
   lprior<-lprior+log(mlk[Zprop[i]])
   lfunctrans<-lfunctrans+log(probZij[Zprop[i]])
   if (nj > 0){
    l<-1:Lk[Sold]
    Zvector<-c(Zij[which(taujcand==Sold)],Zold[which(Zold<=Lk[Sold] & Zold>0)])
    mlk<-rep(0,Lk[Sold])
    need<-data.frame(table(Zvector))
    categr<-as.numeric(as.character(need[,1]))
    mlk[categr]<-as.numeric(need[,2])
    probZij<-mlk*dnorm(Yclusterj[i],theta.samp[(cumsum(Lk)[Sold]-Lk[Sold]+l),1],
sqrt(theta.samp[(cumsum(Lk)[Sold]-Lk[Sold]+l),2]))
    if (length(which(mlk==0))==0){
     mlk<-c(mlk,beta)
     probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,
desvio))} else {
     empty<-which(mlk==0)
     if (mlk[Zclusterj[i]]==0){
      mlk[Zclusterj[i]]<-beta
      probZij[Zclusterj[i]]<-mlk[Zclusterj[i]]*dstudentt(Yclusterj[i],ni,mu,
desvio)} else {
      mlk[empty[1]]<-beta
      probZij[empty[1]]<-mlk[empty[1]]*dstudentt(Yclusterj[i],ni,mu,desvio)}}
    probZij<-probZij/sum(probZij)
    mlk<-mlk/sum(mlk)
    Zold[i]<-Zclusterj[i]
    lpriorold<-lpriorold+log(mlk[Zclusterj[i]])
    lfunctransold<-lfunctransold+log(probZij[Zclusterj[i]])} else {
    Zold[1]<-Zclusterj[1]
    Zvector<-c(Zij[which(taujcand==Sold)],Zold[which(Zold <= Lk[Sold] & Zold>0)])
    mlk<-rep(0,Lk[Sold])
    need<-data.frame(table(Zvector))
    categr<-as.numeric(as.character(need[,1]))
    if(length(categr)>0) mlk[categr]<-as.numeric(need[,2])
    probZij<-mlk*dstudentt(Yclusterj[i],ni,mu,desvio)
    if (length(which(mlk==0))==0){
     mlk<-c(mlk,beta)
     probZij<-c(probZij,mlk[length(mlk)]*dstudentt(Yclusterj[i],ni,mu,desvio))}
else {
     empty<-which(mlk==0)
     if (mlk[Zclusterj[i]]==0){
      mlk[Zclusterj[i]]<-beta
      probZij[Zclusterj[i]]<-mlk[Zclusterj[i]]*dstudentt(Yclusterj[i],ni,mu,
desvio)} else {
      mlk[empty[1]]<-beta
      probZij[empty[1]]<-mlk[empty[1]]*dstudentt(Yclusterj[i],ni,mu,
desvio)}}
    probZij<-probZij/sum(probZij)
    mlk<-mlk/sum(mlk)
    Zold[i]<-Zclusterj[i]
    lpriorold<-lpriorold+log(mlk[Zclusterj[i]])
    lfunctransold<-lfunctransold+log(probZij[Zclusterj[i]])}}}
 Sjcand<-Sj
 Sjcand[j]<-Sprop
 taujcand<-tauj
 taujcand[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]<-rep(Sprop,Ij[j])
 Zijcand<-Zij
 Zijcand[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]<-Zprop
 return(list(Sjcand, taujcand, Zijcand, lprior, lfunctrans, Sprop, Zprop,
Yclusterj, Zclusterj,lpriorold,lfunctransold))}
#
############
# calculate the acceptance rate of the pair Sj and all Zij
############
prob.accept<-function(dataset, Yclusterj, Sj, Zij, tauj, Lk, theta.samp, j,
Zijcand, taujcand, Sprop, Zprop, lambda, mu, agam, bgam, lprior,
lfunctrans, lpriorold, lfunctransold){
 marglikel<-0
 marglikelold<-0
 #
 Yclusterk<-dataset[which(tauj==Sprop),2]
 Zclusterk<-Zij[which(tauj==Sprop)]
 Lsprop<-length(table(c(Zclusterk,Zprop)))
 if (length(Zclusterk)==0){
  mlk<-rep(0,Lsprop)
  Ymeanlk<-rep(0,Lsprop)
  Yvarlk<-rep(0,Lsprop)}
 if (length(Zclusterk)>0){
  mlk<-table(Zclusterk)
  Ymeanlk<-aggregate(Yclusterk, by = list(Zclusterk,rep(1,length(Zclusterk))),
FUN = "mean")[,3]
  Ymeanlk[is.na(Ymeanlk)] <- 0
  Yvarlk<-aggregate(Yclusterk, by = list(Zclusterk,rep(1,length(Zclusterk))),
FUN = "var")[,3]*(mlk-1)
  Yvarlk[is.na(Yvarlk)] <- 0
  dif<-Lsprop-length(mlk)
  if (dif > 0){
   mlk<-c(mlk,rep(0,dif))
   Ymeanlk<-c(Ymeanlk,rep(0,dif))
   Yvarlk<-c(Yvarlk,rep(0,dif))}}
 mupost<-((mlk*Ymeanlk)+(lambda*mu))/(lambda+mlk)
 lambdapost<-lambda+mlk
 agampost<-agam+(mlk/2)
 bgampost<-bgam+(Yvarlk+((mlk*lambda*(Ymeanlk-mu)**2)/(lambda+mlk)))/2
 for (l in 1:Lsprop){
  Yclusterjl<-Yclusterj[Zprop==l]
  if (length(Yclusterjl)>0) marglikel<-marglikel+dmarglikeli(Yclusterjl,
lambdapost[l],agampost[l],bgampost[l],mupost[l])}
 #
 Yclusterk<-dataset[which(taujcand==Sj[j]),2]
 Zclusterk<-Zijcand[which(taujcand==Sj[j])]
 if (length(Zclusterk)==0){
  mlk<-rep(0,Lk[Sj[j]])
  Ymeanlk<-rep(0,Lk[Sj[j]])
  Yvarlk<-rep(0,Lk[Sj[j]])}
 if (length(Zclusterk)>0){
  mlk<-table(Zclusterk)
  Ymeanlk<-aggregate(Yclusterk, by = list(Zclusterk,rep(1,length(Zclusterk))),
FUN = "mean")[,3]
  Ymeanlk[is.na(Ymeanlk)] <- 0
  Yvarlk<-aggregate(Yclusterk, by = list(Zclusterk,rep(1,length(Zclusterk))),
FUN = "var")[,3]*(mlk-1)
  Yvarlk[is.na(Yvarlk)] <- 0
  dif<-Lk[Sj[j]]-length(mlk)
  if (dif > 0){
   mlk<-c(mlk,rep(0,dif))
   Ymeanlk<-c(Ymeanlk,rep(0,dif))
   Yvarlk<-c(Yvarlk,rep(0,dif))}}
 mupost<-((mlk*Ymeanlk)+(lambda*mu))/(lambda+mlk)
 lambdapost<-lambda+mlk
 agampost<-agam+(mlk/2)
 bgampost<-bgam+(Yvarlk+((mlk*lambda*(Ymeanlk-mu)**2)/(lambda+mlk)))/2
 Zold<-Zij[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]
 for (l in 1:Lk[Sj[j]]){
  Yclusterjl<-Yclusterj[Zold==l]
  if (length(Yclusterjl)>0) marglikelold<-marglikelold+dmarglikeli(Yclusterjl,
lambdapost[l],agampost[l],bgampost[l],mupost[l])}
 #
 paccept<-exp(marglikel+lprior+lfunctransold-(marglikelold+lpriorold+lfunctrans))
 return(list(paccept,lprior,lpriorold,lfunctrans,lfunctransold,marglikel,
marglikelold))}
#
########################
# Model initialization
########################
```

```
#
# data set needs to be ordered by elements and replications of each element
# column 1 identifies the elements and the column 2 has the observations for
# each element
#
caminho<-"/xxx/xxxxx/" # this directory is where the files with MCMC output
# will be saved, then you need to change it for a directory of your computer
J<-max(dataset[,1])
Nt<-nrow(dataset)
alpha<-1 # total mass parameter
beta<-1 # total mass parameter
agam<-3 # alpha hyperparameter of Inverse gamma distribution (G0)
bgam<-5 # beta hyperparameter of Inverse gamma distribution (G0)
mu<-0 # mu hyperparameter of Inverse gamma distribution (G0)
lambda<-0.01 # lambda hyperparameter of Inverse gamma distribution (G0)
ni<-2*agam # parameter of student-t distribution
desvio<-sqrt((2*bgam*(1+lambda))/(ni*lambda)) # parameter of t distribution
Ij<-numeric(J)
for (i in 1:J) Ij[i]<-sum(dataset[,1]==i)
#
# Initialize Sj, Zij and theta
#
Sj<-rep(1,J) # distributional cluster membership indicator of elements
Zij<-rep(1,Nt) # observational cluster membership indicator
K<-length(table(Sj))
Lk<-numeric(K)
tauj<-numeric()
for (j in 1:length(Sj)) tauj<-c(tauj,rep(Sj[j],Ij[j]))
for (k in 1:K) Lk[k]<-length(table(Zij[which(tauj==k)]))
set.seed(1000)
theta.samp<-posteriori.theta(dataset, Zij, tauj, lambda, agam, bgam, mu)
#
indrejtotal<-0 #initialize vector indrejtotal
probacetotal<-1 #initialize vector probacetotal
#
amostrasfin<-1000 # MCMC sample size after burn in and jumps
burnin<-1000 # burn in size
saltos<-10 # jumps size
```

```
AmostrasTotal<-burnin+amostrasfin*saltos # number of iterations to be run
set.seed(300)
#
library(compiler)
enableJIT(3)
#
for (int in 1:AmostrasTotal){
#
######### update Sj and Zij by a MH step of one j
#
j<-sample(1:J,1)
#
cat('\n', int, j, K)
candidato<-sample.tau.xi(dataset, Ij, Sj, Zij, tauj, K, Lk, theta.samp, j,
alpha,beta, ni, mu, desvio)
paccept<-prob.accept(dataset, candidato[[8]], Sj, Zij, tauj, Lk, theta.samp,
j,candidato[[3]], candidato[[2]], candidato[[6]], candidato[[7]], lambda,
mu,agam, bgam, candidato[[4]], candidato[[5]], candidato[[10]],
candidato[[11]]) [[1]]
probacetotal<-c(probacetotal,paccept)
aux2<-runif(1)
if (aux2<paccept){
 indrejtotal<-c(indrejtotal,0)
 Sj[j]<-candidato[[6]]
 tauj[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]<-rep(candidato[[6]],Ij[j])
 Zij[(cumsum(Ij)[j]-Ij[j]+1):cumsum(Ij)[j]]<-candidato[[7]]
 while (length(table(Sj))<max(Sj)){ # exclude empty distributional clusters
  categr<-as.numeric(as.character(data.frame(table(Sj))[,1]))
  categd<-seq(1:length(table(Sj)))
  dif<-which(categr!=categd)
  for (i in 1:length(Sj)) if (Sj[i]>dif[1]) Sj[i]<-Sj[i]-1
  tauj<-numeric()
  for (n in 1:length(Sj)) tauj<-c(tauj,rep(Sj[n],Ij[n]))}
 K<-length(table(Sj))
 Lk<-numeric(K)
 for (k in 1:K){
  observ<-which(tauj==k)
  Lk[k]<-length(table(Zij[observ]))
  while (Lk[k]<max(Zij[observ])){ # exclude empty observational clusters
   categr<-as.numeric(as.character(data.frame(table(Zij[observ]))[,1]))
   categd<-seq(1:Lk[k])
   dif<-which(categr!=categd)
   for (i in observ) if (Zij[i]>dif[1]) Zij[i]<-Zij[i]-1}}
 theta.samp<-posteriori.theta(dataset, Zij, tauj, lambda, agam, bgam, mu)
 Zij<-posteriori.Zij2(dataset, Zij, tauj, K, theta.samp, ni, mu, desvio, Lk,
beta, j, Sj)
 Lk<-numeric(K)
 for (n in 1:K) Lk[n]<-length(table(Zij[which(tauj==n)]))
 theta.samp<-posteriori.theta(dataset, Zij, tauj, lambda, agam, bgam, mu)}
if (aux2>=paccept) indrejtotal<-c(indrejtotal,1)
#
if (int>burnin & int%%saltos==0){
 cat('',Sj,file=paste(caminho,"Sj_simu1.txt",sep=""),append=T)
 cat('',theta.samp,file=paste(caminho,"theta_simu1.txt",sep=""),append=T)
 cat('',Zij,file=paste(caminho,"Zij_simu1.txt",sep=""),append=T)
 cat('',K,file=paste(caminho,"K_simu1.txt",sep=""),append=T)
 cat('',Lk,file=paste(caminho,"Lk_simu1.txt",sep=""),append=T)}
}
cat('',indrejtotal,file=paste(caminho,"indrejtotal_simu1.txt",sep=""),append=T)
```

Chapter

7

Big Data Clustering using mixture model [1]

In this chapter, we propose two nonparametric Bayesian methods to cluster big data and apply them to cluster genes by patterns of gene-gene interaction. Both approaches define model-based clustering with nonparametric Bayesian priors and include an implementation that remains feasible for big data.

The first method is based on a predictive recursion which requires a single (or two) cycle of simple deterministic calculations for each observation under study. The second scheme is an exact method that divides the data into smaller subsamples and involves local partitions that can be determined in parallel. In a second step, the method requires only the sufficient statistics of each local cluster to derive global clusters.

Under simulated and benchmark data sets the proposed methods compare favorably with other clustering algorithms, including k-means, DBSCAN, SUGS and EM algorithm.

We apply the proposed approaches to cluster a large data set of gene-gene interactions extracted from the online search tool “Zodiac”. The original genetic data consisting of distributions of regression coefficients is pre-processed by summarizing each distribution by a high order Jacobi polynomial.

[1] This chapter is based on the manuscript “Big Data Clustering” submitted for publication (Zuanetti *et al.* , Submitted b). The manuscript is a joint work supervised by Prof. Dr. Peter Müller and with colaboration of Dr. Yitan Zhu, Dr. Shengjie Yang and Prof. Dr. Yuan Ji.

7.1 Introduction

7.1.1 Clustering methods

We develop model-based clustering methods that are suitable for large data sets. The discussion is motivated by an application to cluster around 20,000 genes by their interaction patterns with other genes. The interaction between a pair of genes is represented as a coefficient in an auto-logistic model of gene expression, copy number variation, methylation, and protein activation (when available) for pairs of genes. See Mitra *et al.* (2013) for details. For the upcoming discussion the data are the estimated gene-gene interaction coefficients in these models. That is, we have a histogram of 20,000 reported coefficients for each gene. Genetic interactions play a critical role in cancer development. Therefore, identifying a group of genes that has, systematically, the highest correlation with many others genes is helpful to understand the relative importance of genes.

A traditional and useful classification of clustering methods is into partitional and hierarchical clustering methods. Hierarchical clustering algorithms recursively find nested clusters, starting from either all singleton clusters and merging clusters, or starting from a single large cluster and then proceeding with recursive splits of clusters. Partitional clustering algorithms find all clusters simultaneously as a partition of the data and do not impose a hierarchical structure. Xu *et al.* (2005) reviews the most commonly used hierarchical and partitional clustering algorithms in statistics, computer science and machine learning. Jain (2010) summarizes well known partitional clustering methods and discusses major challenges in designing such clustering algorithms.

One of the most popular partitional algorithms is k-means and variations for various types of data. It is typically used with an Euclidean metric to find spherical or ball-shaped clusters. Versions with Mahalanobis distance, Itakura-Saito distance, L1 distance and Bregman divergence have been used to detect different shaped clusters (Banerjee *et al*, 2005; Kashima *et al*, 2008; Linde *et al*, 1980; Mao & Jain, 1996). Other variants of k-means include ISODATA (which consider the effect of outliers in clustering), FOGGY, Fuzzy c-means (which allows multiple cluster membership), and many others. See, for example, Jain (2010) for a review and more references. Kulis & Jordan (2012) propose another variant of k-means algorithm, known as DP-means, which adds a penalty based on the number of clusters in the objective function.

Although k-means is a deterministic method, it can be shown that, under suitable conditions, the solution is equivalent to the solution under some model-based clustering. Kulis & Jordan (2012) show that the estimated partition under DP-means is approximately the maximum *a posteriori* (MAP) solution under model-based clustering using a Dirichlet Process (DP) mixture of normals in the limit, as the kernel standard deviation goes to zero.

Another popular partitional deterministic clustering algorithm is DBSCAN (Ester *et al.*

, 1996) which requires two input parameters and discovers cluster of arbitrary shape. It is a density based method where clusters can be defined as high density regions in the feature space, separated by low density regions. Although it seems to be more efficient than k-means for large databases, results in simulated and real data sets show that DBSCAN can provide a large number of singleton clusters.

In addition to deterministic schemes, a number of probabilistic model-based approaches have been developed for data clustering. Many approaches assume that the data is generated by a mixture model. Sampling from a mixture model implies a prior on a random partition of the data, using the following construction. First we rewrite the mixture model as a hierarchical model with latent indicator variables that select terms in the mixture. Interpreting these latent variables as cluster membership indicators a mixture model implies a random partition. Fraley & Raftery (2002) describe and review a methodological framework. The approach comprises three core elements: initialization via model-based hierarchical agglomerative clustering, maximum likelihood estimation via the EM algorithm, and selection of the model and the number of clusters using approximate Bayes factors. The latter can be implemented, for example, using an approximation with the Bayesian information criteria (BIC).

7.1.2 Big data clustering

Most of the above mentioned methods fail for big data, due to computational constraints and the need to access all data simultaneously. Most clustering algorithms that handle large-size data sets can be characterized as data summarization, distributed computing, incremental clustering or sampling-based methods. Data summarization methods first summarize a big data set into a relatively small subset and then apply the clustering algorithms to the summarized data set. Dimension reduction not only makes it possible to work with big data and reduces the computational cost, but can also provide users with a visual examination of the data of interest (Xu *et al*, 2005). Distributed computing methods divide each step of a data clustering algorithm into a number of procedures that can be computed independently, in parallel. Incremental clustering algorithms are designed to work with a single (or few) pass over all data points. Sampling-based methods subsample a large data set selectively, and perform clustering for the smaller subset and later transfer the results to the larger data set.

Some examples of distributed computing algorithms are the following methods. Guha *et al.* (2003) propose a simple algorithm that splits the data set into random smaller data subsets, finds clusters for each one using a k-medians algorithm and then clusters all the intermediate medians into K final medians. Pennell & Dunson (2007) proposed a 2-stage method for fitting semiparametric random effects models to large data sets. In the first stage, they construct $G << n$ (where n is the sample size) clusters using a method similar to a k-means method. In the second stage, they use a Dirichlet process prior (DP) for the G cluster means. Inference under the DP prior implies that some of the earlier G cluster means are merged. Zhao *et al.*

(2009) adapt the k-means algorithm to the MapReduce framework to make the clustering method applicable to large scale data.

Unrelated to clustering, many Bayesian model-based methods for big data are variations of a strategy known as consensus Monte Carlo. Huang & Gelman (2005) suggest four alternative methods for consensus Monte Carlo. The first variation is combining separate subset-specific inference using a normal approximation. The second variation consists in carrying out prior to *a posteriori* updating sequentially; that is, fitting the model for the first subset and using the *a posteriori* distribution of this subset as a prior to estimate the model for second subset and so on. The third option is starting inference with one subset and adding other data using importance sampling. The final suggestion generalizes the second option by combining importance sampling and a birth-death process. The problem with importance sampling methods is the potential risk that the Monte Carlo sample could collapse to a single high dimensional point. Scott *et al.* (2016) also propose a consensus Monte Carlo approach to divide the data across multiple machines, with each machine implementing separate Monte Carlo simulation from the subset *a posteriori*. The *a posteriori* draws are then combined using weighted averages to form a consensus Monte Carlo sample.

Sequential importance sampling (SIS) (MacEachern *et al*, 1999) and sequential updating and greedy search (SUGS) (Wang & Dunson, 2011) are examples of incremental clustering algorithms with nonparametric mixture models in big data. MacEachern *et al.* (1999) did not originally develop an approach for clustering, but the method involves latent variables that can be used to define a random partition on the basis of ties of these variables. The method exploits the similarities between the collapsed Gibbs sampler and SIS and propose a method which clusters observations sequentially without running a Markov chain. Instead, many independent and identically distributed replicates are simulated to create an importance sample. Wang & Dunson (2011) propose a similar scheme, similar to SIS in the sense of sequential updating but different in that it adopts a random assignment of allocation instead of finding the allocation that maximizes the *a posteriori* probability of cluster membership indicators. A common limitation of these methods is potentially sticky cluster allocation.

Building on these methods, in this chapter we propose two nonparametric model-based methods to cluster big data. The first scheme is an incremental clustering algorithm called predictive recursion clustering (PRC). It is based on a predictive recursion algorithm by Newton *et al.* (1998), which is usually applied to approximate the *a posteriori* mean of a DP mixture model. The approximation takes the form of a mixture of kernels that is incrementally updated on a grid. We use the terms of the mixture approximation to define clusters and add a step to allocate observations to the cluster that maximizes the marginal *a posteriori* after all observations have been included in the model. It is an approximate method that avoids a full Markov chain Monte Carlo *a posteriori* simulation.

The second method is a distributed computing algorithm called subset nonparametric Bayesian (SNOB) clustering. It is an exact method that divides the data into smaller groups, referred to as shards, across multiple machines and identifies local clusters. In a second step, we combine the local clusters to determine global clusters in a MapReduce or Hadoop framework. The method is simulation exact, in the sense that the global clusters are built on the basis of a probability model for the full original data. Importantly, in the second step we need only sufficient statistics for cluster-specific parameters from the first step. We apply the two proposed big data model-based clustering approaches to group genes by gene-gene interactions.

The chapter is organized as follows: in Section 7.2 we briefly define the predictive recursion algorithm of Newton *et al.* (1998) and develop the PRC method for clustering big data. Section 7.3 proposes the SNOB method. Section 7.4 applies both methods in two benchmark data sets and one simulated data set to explore the performance of both schemes in clustering big data. We compare their performance with DP-means, DBSCAN, SUGS and EM clustering. Section 7.5 clusters the gene data set using the two proposed methods. In Section 7.6 we discuss the results and in Section 7.7 we show the appendices.

7.2 Predictive recursion clustering (PRC)

7.2.1 Predictive recursion

We introduce a big data clustering algorithm based on the recursive algorithm of Newton *et al.* (1998). The algorithm is usually applied to approximate the *a posteriori* expectation for the unknown mixture distribution in a Dirichlet process (DP) mixture model. A major advantage of the approach is the computationally efficient implementation, in particular, the method processes the data one by one and needs only a single pass over the entire data set.

Let $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ be the observed data where $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{ip})$ for $i = 1, \ldots, n$. Assume $\boldsymbol{y}_i \sim F$, i.i.d., for some unknown distribution F. The widely used DP mixture model writes F as a mixture, $F = \int p(\boldsymbol{y} \mid \boldsymbol{\theta})\, dG(\boldsymbol{\theta})$ of some kernel $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ with respect to a mixing measure G which in turn is assigned a DP prior, $G \sim \text{DP}(\alpha G_0)$. Here $\text{DP}(\mu)$ denotes a DP prior with base measure $\mu = \alpha G_0$, where G_0 is a normalized probability measure and $\alpha > 0$ is known as the total mass parameter. The DP prior is a nonparametric Bayesian prior for a discrete random probability measure (Ferguson, 1973). See, for example, Ghoshal (2010) for a review. The DP mixture model can be equivalently written as a hierarchical model with latent variables $\boldsymbol{\theta}_i \sim G$, as

$$\boldsymbol{y}_i \mid \boldsymbol{\theta}_i \overset{indep}{\sim} p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i), \quad \boldsymbol{\theta}_i \mid G \overset{iid}{\sim} G \text{ and } G \sim \text{DP}(\alpha G_0). \tag{7.1}$$

Ties in the $\boldsymbol{\theta}_i$ give rise to the desired partition of experimental units with clusters defined by matching unique values of the $\boldsymbol{\theta}_i$. Let $g_n(\boldsymbol{\theta}_{n+1}) \equiv p(\boldsymbol{\theta}_{n+1} \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ denote the *a posteriori* predictive distribution. The *a posteriori* predictive $g_n(\boldsymbol{\theta})$ is identical to the *a posteriori* mean $E(G \mid \boldsymbol{y})$. This is easily seen by considering $p(\boldsymbol{\theta}_{n+1} \mid \boldsymbol{y}) = E\left\{p(\boldsymbol{\theta}_{n+1} \mid G) \mid \boldsymbol{y}\right\} =$

$E\{G(\boldsymbol{\theta}_{n+1}) \mid \boldsymbol{y}\}$. The predictive recursion algorithm of Newton *et al.* (1998) approximates $g_i(\boldsymbol{\theta})$ as

$$g_i(\boldsymbol{\theta}) = (1-\omega_i)g_{i-1}(\boldsymbol{\theta}) + \omega_i \frac{p(\boldsymbol{y}_i \mid \boldsymbol{\theta})g_{i-1}(\boldsymbol{\theta})}{c(\boldsymbol{y}_i, g_{i-1})}, \tag{7.2}$$

starting with $g_0 = G_0$. Here, $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$ is a sequence of weights and $c(\boldsymbol{y}_i, g_{i-1}) = \int p(\boldsymbol{y}_i \mid \boldsymbol{\theta})g_{i-1}(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ are normalization constant. Updating is exact for $i = 1$ and $w_1 = 1/(1+\alpha)$, and is an approximation beyond that. That is, for $i = 1$, (7.2) defines exactly the *a posteriori* predictive distribution under the DP mixture model. Convergence holds as long as the positive weights satisfy $\sum_i \omega_i = \infty$ and $\lim_{i\to\infty} w_i = 0$ (Newton *et al*, 1998). Model (7.1) naturally suggests $\omega_i = 1/(i+\alpha)$, however sequences that vanish a bit slower are also allowed. The updated estimate g_i is a mixture of the current estimate g_{i-1} and new information $\boldsymbol{y}_i$ in the form of an *a posteriori* density on $\boldsymbol{\theta}_i$ under a model with prior g_{i-1} and likelihood $p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i)$.

Under (7.2) the (approximate) predictive g_i is a mixture of 2^i components

$$g_i(\boldsymbol{\theta}) = \sum_{k=1}^{2^i} \pi_k^{(i)} f_k^{(i)}(\boldsymbol{\theta}), \tag{7.3}$$

with weights $\pi_k^{(i)}$ and components $f_k^{(i)}$:

$$\pi_k^{(i)} = \begin{cases} (1-\omega_i)\pi_k^{(i-1)} \\ \omega_i \pi_{k-2^{i-1}}^{(i-1)} \end{cases} \quad \text{and } f_k^{(i)}(\boldsymbol{\theta}) = \begin{cases} f_k^{(i-1)}(\boldsymbol{\theta}), & \text{for } k = 1, \ldots, 2^{i-1} \\ \propto f_{k-2^{i-1}}^{(i-1)}(\boldsymbol{\theta})\, p(\boldsymbol{y}_i \mid \boldsymbol{\theta}), & \text{for } k = 2^{i-1}+1, \ldots, 2^i, \end{cases}$$

starting with $\pi_1^{(0)} = 1$ and $f_1^{(0)}(\boldsymbol{\theta}) = G_0(\boldsymbol{\theta})$. When G_0 and $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ are conjugate updating of $f_k^{(i)}$ is straightforward. In the remaining discussion we assume such conjugate setup, and let $\boldsymbol{\eta}_k^{(i)}$ denote the parameters that index $f_k^{(i)}$, and we will write $f_k^{(i)}(\boldsymbol{\theta} \mid \boldsymbol{\eta}_k^{(i)})$ if we want to highlight this.

Calculation of $g_n(\boldsymbol{\theta})$ only requires a single cycle of simple deterministic updates for each observation under study and remains therefore feasible also for large data sets. However, it is impractical to record all 2^i components. Instead we later merge and reduce the number of components. See later for details. For the moment we only assume that g_n is a mixture with known weights $\pi_k^{(i)}$ and components $f_k^{(i)}$.

Let $\mathrm{N}(y \mid \boldsymbol{\theta} = (\mu, \sigma^2))$ denote a normal distribution for y and let $\mathrm{NIG}(\boldsymbol{\theta} = (\mu, \sigma^2) \mid \mu_0, \lambda_0, a_0, b_0)$ denote a normal inverse gamma distribution for $\boldsymbol{\theta}$. That is, a gamma distribution for $1/\sigma^2$, $p(1/\sigma^2) = \mathrm{Ga}(a_0, b_0)$, and a conditional normal distribution for μ, $p(\mu \mid \sigma^2) = \mathrm{N}(\mu, \sigma^2/\lambda_0)$. Here the gamma distribution is parametrized such that $E(1/\sigma^2) = a_0/b_0$. Defining the sampling model $p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i)$ we assume that y_{ij} and $y_{ij'}$ are independent given $\boldsymbol{\theta}_i$, and $j \neq j'$. The independence restriction is not a strong assumption as it could always be achieved by using transformed data $\tilde{\boldsymbol{y}} = \left(A^{-1}\boldsymbol{y}^T\right)^T$, with a factorization of an arbitrary non-diagonal covariance matrix AA'. We therefore assume a normal sampling model, $p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i = (\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)) = \prod_{j=1}^p \mathrm{N}(y_{ij} \mid \mu_{ij}, \sigma_{ij}^2)$, and a conjugate baseline measure $G_0(\boldsymbol{\theta}) = \prod_{j=1}^p \mathrm{NIG}(\boldsymbol{\theta}_j \mid \mu_{0j}, \lambda_{0j}, a_{0j}, b_{0j})$. Under this model, the updating in (7.3) becomes

$$f_k^{(i)}(\boldsymbol{\theta}_i \mid \boldsymbol{\eta}_k^{(i)}, \boldsymbol{y}_i) = \prod_{j=1}^p \mathrm{NIG}\left(\boldsymbol{\theta}_j \mid \mu_{kj}^{(i)}, \lambda_{kj}^{(i)}, a_{kj}^{(i)}, b_{kj}^{(i)}\right) \text{ for } k = 2^{i-1}+1, \ldots, 2^i \tag{7.4}$$

with

$$\mu_{kj}^{(i)} = \frac{y_{ij} + \lambda_{k-2^{i-1}j}^{(i-1)}\mu_{k-2^{i-1}j}^{(i-1)}}{\lambda_{kj}^{(i)}},\ b_{kj}^{(i)} = b_{k-2^{i-1}j}^{(i-1)} + \frac{\lambda_{k-2^{i-1}j}^{(i-1)}\left(y_{ij} - \mu_{k-2^{i-1}j}^{(i-1)}\right)^2}{2\lambda_{kj}^{(i)}}, \tag{7.5}$$

$\lambda_{kj}^{(i)} = \lambda_{k-2^{i-1}j}^{(i-1)} + 1$, and $a_{kj}^{(i)} = a_{k-2^{i-1}j}^{(i-1)} + 1/2$. That is, $\boldsymbol{\eta}_k^{(i)} = (\mu_{kj}^{(i)}, \lambda_{kj}^{(i)}, a_{kj}^{(i)}, b_{kj}^{(i)};\ j = 1, \ldots, p)$. The recursion starts with $\boldsymbol{\eta}_1^{(0)} = (\mu_{0j}, \lambda_{0j}, a_{0j}, b_{0j}, j = 1, \ldots, p)$.

7.2.2 Merging similar components and removing order dependence

For a large data set it is impractical to keep all components and weights in memory. In addition, most weights are negligible. Hennig (2010) reviews and compares different methods for merging normal mixture components. One method that compares favorably is based on thresholding a dissimilarity measure between pairs of components. Hennig (2010) suggests using Bhattacharyya distance. Here we propose to use Mahalanobis distance, a special case of Bhattacharyya distance, since for Mahalanobis distance the cutoff-value to merge a pair of components can be easily determined as a χ^2-quantile, whereas the cutoff-value for a Bhattacharyya distance is, in general, a subjective choice.

In our case $g_i(\boldsymbol{\theta})$ is a mixture of normal inverse gamma distributions. To reduce it to a mixture of normals, we sample $1/\sigma_{kj}^{2(i)} \sim \mathrm{Ga}(a_{kj}^{(i)}, b_{kj}^{(i)})$, $j = 1, \ldots, p$, and define

$$d(\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\mu}_{k'}^{(i)}) = \sum_{j=1}^p \frac{\left(\mu_{kj}^{(i)} - \mu_{k'j}^{(i)}\right)^2}{\left(\sigma_{kj}^{2(i)}/\lambda_{kj}^{(i)} + \sigma_{k'j}^{2(i)}/\lambda_{k'j}^{(i)}\right)/2}. \tag{7.6}$$

We use the Mahalanobis distance (7.6) to merge terms as follows. First, we fix a threshold $d^\star$ as a $(1-q)$ tail cutoff in a χ^2 distribution with p degrees of freedom. In our implementation we use $(1-q) = 75\%$ for $i = 1, \ldots, n-1$, and $(1-q) = 95\%$ for the last step, $i = n$, of updating (7.3). In each step, before recording the new approximation $g_i(\cdot)$ we merge the components with the smallest distance $d(\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\mu}_{k'}^{(i)}) < d^\star$ by dropping the component with the smaller weight, rescaling the remaining weights and continue dropping components while there are pairs with $d(\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\mu}_{k'}^{(i)}) < d^\star$. Instead of dropping one of the components, we could combine their parameters and build a new component with new parameters. However, in that case, we would need to recompute Mahalanobis distances for all pairs.

Finally, we add one more element to the algorithm. Because of sequential updating, the predictive recursion estimate $g_i(\boldsymbol{\theta})$ depends on the order in which observations are added. Although this dependence can be weak (Newton *et al*, 1998) we prefer to mitigate it. We repeat the predictive recursion clustering for multiple permutations of the ordering and use an approximate pseudo-marginal likelihood (PML) to select the best order. See Gelfand & Dey (1994) and Pettit (1990) for details about PML and Wang & Dunson (2011) for an application of PML to select the best estimate in sequential updating.

PML is defined as product of conditional predictive ordinates as

$$\begin{aligned}\mathrm{PML}(\boldsymbol{y}) &= \prod_{i=1}^{n} p(\boldsymbol{y}_i \mid \boldsymbol{y}_{-i}) = \prod_{i=1}^{n} \int p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i) p(\boldsymbol{\theta}_i \mid \boldsymbol{y}_{-i}) \mathrm{d}\boldsymbol{\theta}_i \\ &\approx \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k^{(n)} \int p\left(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i\right) f_k^{(n)}(\boldsymbol{\theta}_i)\, \mathrm{d}\boldsymbol{\theta}_i, \end{aligned} \tag{7.7}$$

where

$$\int p\left(\boldsymbol{y}_i \mid \boldsymbol{\theta}_i\right) f_k^{(n)}(\boldsymbol{\theta}_i)\, \mathrm{d}\boldsymbol{\theta}_i = \prod_{j=1}^{p} \mathrm{t}\left(y_{ij} \mid 2a_{kj}^{(n)}, \mu_{kj}^{(n)}, \left(\frac{b_{kj}^{(n)}(1+\lambda_{kj}^{(n)})}{a_{kj}^{(n)} \lambda_{kj}^{(n)}} \right)^{1/2} \right) \equiv p(\boldsymbol{y}_i \mid \boldsymbol{\eta}_k^{(n)}). \tag{7.8}$$

In (7.8) $\mu_{kj}^{(n)}, a_{kj}^{(n)}, b_{kj}^{(n)}$ and $\lambda_{kj}^{(n)}$ are defined in (7.5), and $\mathrm{t}(y \mid \nu, m, s)$ denotes a t-distribution with ν degrees of freedom and location and scale parameters m and s. We refer to (7.8) as $p(\boldsymbol{y}_i \mid \boldsymbol{\eta}_k^{(n)})$, the marginal model for $\boldsymbol{y}_i$, marginalizing $\boldsymbol{\theta}_i$ with respect to the k-th term in $g_n(\cdot)$. We will use (7.8) later again to approximate the desired clustering under the DP mixture (7.1).

The approximation in (7.7) is that $\pi_k^{(n)}$ and $f_k^{(n)}(\boldsymbol{\theta}_i)$ should be computed without considering observation $\boldsymbol{y}_i$ for $i = 1, \ldots, n$. However, to speed up computation we evaluate them once over all observations. The approximation is negligible for a large data sets. We select the order with the largest PML.

7.2.3 PRC algorithm

At this moment of the construction we have an approximation $g_n(\boldsymbol{\theta}_{n+1}) \approx p(\boldsymbol{\theta}_{n+1} \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ as a mixture of K NIG terms. Here, K is the number of terms left after merging. We now use the latter to impute the $\boldsymbol{\theta}_i$ in (7.1). That is, we exploit the mixture model $g_n(\cdot)$ to approximate the desired clustering under the DP mixture. We use $p(\boldsymbol{\theta}_i \mid \boldsymbol{y}_{-i}) \approx g_n(\boldsymbol{\theta}_i)$ to get $p(\boldsymbol{\theta}_i \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \propto p(y_i \mid \boldsymbol{\theta}_i)\, p(\boldsymbol{\theta}_i \mid \boldsymbol{y}_{-i}) \approx p(y_i \mid \boldsymbol{\theta}_i)\, g_n(\boldsymbol{\theta}_i)$. Next, let s_i denote a latent cluster membership indicator to select a term of the mixture g_n in the approximation. Then, denoting $\boldsymbol{\pi}^{(n)} = (\pi_1^{(n)}, \ldots, \pi_K^{(n)})$, we have

$$p\left(s_i = k \mid \boldsymbol{\pi}^{(n)}, \boldsymbol{\eta}^{(n)}, \boldsymbol{y}\right) \propto \pi_k^{(n)}\, p(\boldsymbol{y}_i \mid \boldsymbol{\eta}_k^{(n)}) \tag{7.9}$$

for $k = 1, \dots, K$, using $p(\boldsymbol{y}_i \mid \boldsymbol{\eta}_k^{(n)})$ from (7.8).

Let $\boldsymbol{\theta}_k^\star$, $k = 1, \dots, K$, denote the unique values of $\boldsymbol{\theta}_i$, with $\boldsymbol{\theta}_i = \boldsymbol{\theta}_k^\star$ when $s_i = k$. Using the *a posteriori* for $\boldsymbol{\theta}_k^\star$ defined as

$$p\left(\boldsymbol{\theta}_k^\star \mid \boldsymbol{s}, \boldsymbol{y}\right) \propto G_0(\boldsymbol{\theta}_k^\star) \prod_{i:s_i=k} p(\boldsymbol{y}_i \mid \boldsymbol{\theta}_k^\star), \tag{7.10}$$

we generate $\boldsymbol{\theta}_k^\star$, $k = 1, \dots, K$. In our case (7.10) is a normal inverse gamma distribution. Finally, we update the cluster membership indicators s_i, now using the updated cluster-specific $\boldsymbol{\theta}_k^\star$,

$$p\left(s_i = k \mid \boldsymbol{s}_{-i}, \boldsymbol{y}, \boldsymbol{\pi}^{(n)}, \boldsymbol{\theta}^\star\right) \propto \pi_k^{(n)} \prod_{j=1}^{p} p(y_{ij} \mid \boldsymbol{\theta}_{kj}^\star), \tag{7.11}$$

for $k = 1, \dots, K$. While the last step is not strictly necessary – we already have s_i – we found that it improved observed mis-classification rates in simulation experiments.

```
Algorithm 2 PRC algorithm
Input: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ (data) and $d^\star$ (cutoff-value for merging components)
Output: Cluster membership indicators $s_1, ..., s_n$ and number of clusters $K$

  1. For $o = 1, \dots, O$, where $O$ is the number of different permutations of data.
      (a) Reorder the data.
      (b) For $i = 1, \dots, n$
            i. Compute the elements of $g_i(\boldsymbol{\theta})$.
           ii. Merge pairs of components with $d < d^\star$.
      (c) Calculate pseudo-marginal likelihood (PML) using (7.7).
  2. Select the ordering with highest PML.
  3. Draw $\boldsymbol{s}$ as in (7.9).
  4. Update $\boldsymbol{\theta}_k^\star$ using (7.10), $j = 1, \dots, p$ and $k = 1, \dots, K$.
  5. Update $\boldsymbol{s}$ as in (7.11).
```

7.3 Subset nonparametric Bayesian (SNOB)

For large sample sizes, the usual MCMC algorithms for *a posteriori* computations in DP mixture models as in (7.1) are computationally impractical. The recursive scheme of Section 7.2 is a computationally attractive approximation of *a posteriori* inference on the random partition. Alternatively, we propose the following exact nonparametric scheme for big data clustering. The approach is exact in the sense of following *a posteriori* inference in the DP mixture model (7.1) for the entire data.

The idea of the proposed scheme is to split the data into B smaller subsets, called *shards*, and perform nonparametric clustering in each shard. We refer to the resulting partition in each subset as local clusters. In a second step, we cluster the local clusters to create global clusters. The shards can be processed in parallel. The *a posteriori* draws are then combined to identify the global clusters. Importantly, both steps are carried out to follow *a posteriori* inference under one common underlying DP mixture model (7.1) for the entire data. In that sense the method is simulation exact. In summary, we cluster the full data $\boldsymbol{y}$ by splitting the data as $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_B)$ where $\boldsymbol{y}_b = (\boldsymbol{y}_{b1}, \ldots, \boldsymbol{y}_{bI_b})$ denotes shard $b = 1, \ldots, B$, I_b is the number of observations in the b-th subset and $\sum_{b=1}^{B} I_b = n$. Let $\boldsymbol{y}_{bh}$ be the h-th observation of b-th shard for $h = 1, \ldots, I_b$. We can rewrite model (7.1) as

$$\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh} \overset{indep}{\sim} p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh}),\ \boldsymbol{\theta}_{bh} \mid G \overset{iid}{\sim} G \text{ and } G \sim \text{DP}(\alpha G_0), \tag{7.12}$$

$b = 1, \ldots, B$ and $h = 1, \ldots, I_b$.

7.3.1 Clustering each shard and estimating local clusters

The discrete nature of G in model (7.12) gives rise to ties among $\boldsymbol{\theta}_{bh}$. Let $\{\boldsymbol{\theta}^\star_{b1}, \ldots, \boldsymbol{\theta}^\star_{bL_b}\}$ denote the unique elements among $\boldsymbol{\theta}_{bh}$ for the specific shard b and $\boldsymbol{r}_b = (r_{b1}, \ldots, r_{bI_b})$ the local cluster membership indicators for b-th shard with $r_{bh} = \ell$ if $\boldsymbol{\theta}_{bh} = \boldsymbol{\theta}^\star_{b\ell}$, for $h = 1, \ldots, I_b$. Let $R_{b\ell} = \{h : r_{bh} = \ell\}$ denote the local clusters and $n_{b\ell} = |R_{b\ell}|$ denote the sizes of the L_b local clusters. For later reference we state that the DP prior in (7.12) implies

$$p(\boldsymbol{r}_b) \propto \alpha^{L_b} \prod_{\ell=1}^{L_b} (n_{b\ell} - 1)! \tag{7.13}$$

The random partition (7.13) is known as the Polya urn (Blackwell & MacQueen, 1973) and implies

$$p(r_{bh} = \ell \mid \boldsymbol{r}_b^-) \propto \begin{cases} n_{b\ell}^-, \text{ if } \ell = 1, \ldots, L_b^- \\ \alpha, \text{ if } \ell = L_b^- + 1, \end{cases} \tag{7.14}$$

where the superscript xx^- represents the appropriate quantity xx with observation bh excluded from the sample. See, for example, Ghoshal (2010) for a review, including *a posteriori* simulation schemes for $\boldsymbol{r}_b$. Let $p(\boldsymbol{y}_{bh} \mid r_{bh} = \ell, \boldsymbol{r}_b^-, \boldsymbol{y}_b^-) = \int p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}^{\star -}_{b\ell}) p(\boldsymbol{\theta}^{\star -}_{b\ell} \mid \boldsymbol{r}_b^-, \boldsymbol{y}_b^-) \mathrm{d}\boldsymbol{\theta}^{\star -}_{b\ell}$ and $h_0(\boldsymbol{y}_{bh}) = \int p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh}) G_0(\boldsymbol{\theta}_{bh}) \mathrm{d}\boldsymbol{\theta}_{bh}$. The *a posteriori* simulation includes resampling of the local cluster membership indicators r_{bh}, $h = 1, \ldots, I_b$, using the complete conditional *a* *posteriori* probabilities

$$p(r_{bh} = \ell \mid \boldsymbol{r}_b^-, \boldsymbol{y}) \propto \begin{cases} n_{b\ell}^- \, p(\boldsymbol{y}_{bh} \mid r_{bh} = \ell, \boldsymbol{r}_b^-, \boldsymbol{y}_b^-), & \text{for } \ell = 1, \ldots, L_b^- \\ \alpha \, h_0(\boldsymbol{y}_{bh}), & \text{for } \ell = L_b^- + 1. \end{cases} \tag{7.15}$$

For conjugate DP mixtures $p(\boldsymbol{\theta}_{b\ell}^{\star\,-} \mid \boldsymbol{r}_b^-, \boldsymbol{y}_b^-)$, $p(\boldsymbol{y}_{bh} \mid r_{bh} = \ell, \boldsymbol{r}_b^-, \boldsymbol{y}_b^-)$ and $h_0(\boldsymbol{y}_{bh})$ are available in closed form. For example, consider a normal sampling model, $p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh} = (\boldsymbol{\mu}_{bh}, \boldsymbol{\sigma}_{bh}^2)) = \prod_{j=1}^p \mathrm{N}(y_{bhj} \mid \boldsymbol{\theta}_{bhj} = (\mu_{bhj}, \sigma_{bhj}^2))$, and a conjugate baseline measure $G_0(\boldsymbol{\theta}) = \prod_{j=1}^p \mathrm{NIG}(\boldsymbol{\theta}_j \mid \mu_{0j}, \lambda_{0j}, a_{0j}, b_{0j})$. The complete conditional *a posteriori* distribution of $\boldsymbol{\theta}_{b\ell}^{\star\,-}$ is

$$p(\boldsymbol{\theta}_{b\ell}^{\star\,-} \mid \boldsymbol{r}_b^-, \boldsymbol{y}_b^-) = \prod_{j=1}^p \mathrm{NIG}(m_{b\ell j}, \lambda_{b\ell j}, a_{b\ell j}, b_{b\ell j}), \tag{7.16}$$

with $\lambda_{b\ell j} = \lambda_{0j} + n_{b\ell}^-$, $a_{b\ell j} = \frac{n_{b\ell}^-}{2} + a_{0j}$,

$$m_{b\ell j} = \frac{n_{b\ell}^- \bar{y}_{b\ell j} + \lambda_{0j}\mu_{0j}}{\lambda_{0j} + n_{b\ell}^-}, \text{ and } b_{b\ell j} = b_{0j} + \frac{s_{b\ell j}^2 + \frac{n_{b\ell}^- \lambda_{0j} (\bar{y}_{b\ell j} - \mu_{0j})^2}{\lambda_{0j} + n_{b\ell}^-}}{2},$$

where $\bar{y}_{b\ell j} = \sum_{h \in R_{b\ell}^-} y_{bhj}/n_{b\ell}^-$ and $s_{b\ell j}^2 = \sum_{h \in R_{b\ell}^-} (y_{bhj} - \bar{y}_{b\ell j})^2$ are cluster-specific sample means and (scaled) variances evaluated without observation bh. The marginal likelihood

$$h_0(\boldsymbol{y}_{bh}) = \prod_{j=1}^p \mathrm{t}\left(y_{bhj} \mid 2a_{0j}, \mu_{0j}, \left(\frac{b_{0j}(1 + \lambda_{0j})}{a_{0j}\lambda_{oj}} \right)^{1/2} \right) \tag{7.17}$$

and the marginal likelihood of $\boldsymbol{y}_{bh}$ in first line of (7.15) is equal to (7.17), just changing $(\mu_{0j}, \lambda_{0j}, a_{0j}, b_{0j})$ for $(m_{b\ell j}, \lambda_{b\ell j}, a_{b\ell j}, b_{b\ell j})$.

We use (7.15) and (7.17) to implement *a posteriori* MCMC simulation for the cluster membership indicators for the local partitions $\boldsymbol{r}_b$. So far we only described standard *a posteriori* simulation for each shard. Based on the *a posteriori* simulation output we can then record estimated partitions $\boldsymbol{r}_b$. In our implementation we use thresholding of *a posteriori* co-clustering probabilities $\pi_{bhh'}^\star = p(r_{bh} = r_{bh'} \mid \boldsymbol{y}_b)$. We report clusters using the following heuristic, starting with $r_{b1} = 1$ and using a threshold $p^\star$. For $h > 1$, let $m_h = \arg\max_{h' < h} \pi_{bhh'}^\star$, $K_h = \max\{r_{b1}, \ldots, r_{bh}\}$ and

$$r_{bh} = \begin{cases} r_{bm_h}, & \text{if } \pi_{bhm_h}^\star \geq p^\star \\ K_{h-1} + 1, & \text{otherwise }, \end{cases} \tag{7.18}$$

for $h = 2, \ldots, I_b$. We fix $p^\star = 0.5$, that is, we allocate two observations in the same cluster if they are allocated in the same cluster in more than half of the MCMC sample. Alternatively, any other scheme to report a point estimate for $\boldsymbol{r}_b$ could be used, for example Dahl (2006).

In summary, the inputs to cluster each shard are $\boldsymbol{y}_b$, a starting point for $\boldsymbol{r}_b$ and a threshold $p^\star$ to define final clustering. The outputs are a point estimate for $\boldsymbol{r}_b$ and sufficient statistics of each local cluster: $n_{b\ell}$, $M_{b\ell j} = \sum_{h \in R_{b\ell}} y_{bhj}$ and $S^2_{b\ell j} = \sum_{h \in R_{b\ell}} y^2_{bhj}$, for $\ell = 1, \ldots, L_b$ and $j = 1, \ldots, p$. In next section we will see that $\boldsymbol{\eta}_{b\ell} = (n_{b\ell}, \boldsymbol{M}_{b\ell}, \boldsymbol{S}^2_{b\ell})$ are enough to identify global clusters. We do not need the original data $\boldsymbol{y}_b$ anymore.

7.3.2 Estimating global clusters

Recall now the original inference goal of clustering $\{1, \ldots, n\}$, that is inference on $\boldsymbol{s}$. We construct $\boldsymbol{s}$ by merging local clusters with matching $\boldsymbol{\theta}^\star_{b\ell}$. Similar to inference for r_{bh} in (7.15) we can carry this out marginalizing w.r.t. $\boldsymbol{\theta}^\star_{b\ell}$. Under (7.12) all required *a posteriori* probabilities are available in closed form if the kernel $p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh})$ and $G_0(\boldsymbol{\theta}_{bh})$ are chosen as a conjugate pair.

Let $\boldsymbol{\theta}^{\star\star} = \{\boldsymbol{\theta}^{\star\star}_1, \ldots, \boldsymbol{\theta}^{\star\star}_K\}$ denote the unique elements among $\boldsymbol{\theta}_{bh}$ across all shards and let K denote the number of global clusters. Our interest is to find the global clusters defined by $\boldsymbol{\theta}^{\star\star}$. We will never need to record the $\boldsymbol{\theta}^{\star\star}_k$ – they are only used here to define the notion of global clusters. Let $\tilde{S}_k = \{(b, \ell) : \boldsymbol{\theta}^\star_{b\ell} = \boldsymbol{\theta}^{\star\star}_k\}$ denote all local clusters in the k-th global cluster and let $\tilde{s}_{b\ell} = k$ when $(b, \ell) \in \tilde{S}_k$ denote global cluster membership indicators for each local cluster. Let $S_k = \{(b, h) : \tilde{s}_{br_{bh}} = k\}$ denote all observations allocated in the k-th global cluster and let $s_{bh} = k$ when $(b, h) \in S_k$ denote the global cluster membership indicator for each observation. The implied size of the k-th global cluster is $n_k = |S_k| = \sum_{(b,\ell) \in \tilde{S}_k} n_{b\ell}$. The main inference target is $\boldsymbol{s} = \{s_{bh}, b = 1, \ldots, B, h = 1, \ldots, I_b\}$, the cluster arrangement of the entire data, which is determined by $\tilde{\boldsymbol{s}}$ and $\boldsymbol{r} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_B)$ as $s_{bh} = \tilde{s}_{b, r_{bh}}$. Deriving conditional prior and *a posteriori* probabilities for global clusters, that is for $\tilde{s}_{b\ell}$, parallels the earlier discussion for (7.13). The only change is that a change in $\tilde{s}_{b\ell}$ involves reallocation of multiple experimental units, namely all $h \in R_{b\ell}$. Keeping this detail in mind, similar to (7.13), the joint prior on $\boldsymbol{s}$ is

$$p(\boldsymbol{s}) \propto \alpha^K \prod_{k=1}^{K} (n_k - 1)! \tag{7.19}$$

and implies the conditional prior

$$\begin{aligned} p(\tilde{s}_{b\ell} = k \mid \tilde{\boldsymbol{s}}^-, \boldsymbol{r}) &= \frac{p(\tilde{\boldsymbol{s}}^-, \tilde{s}_{b\ell} = k \mid \boldsymbol{r})}{p(\tilde{s}^- \mid \boldsymbol{r})} \\ &\propto \begin{cases} \frac{\alpha^{K^-} \left(\prod_{m=1 \& m \neq k}^{K^-} \left(n_m^- - 1\right)!\right) \left(n_k^- + n_{b\ell} - 1\right)!}{\alpha^{K^-} \left(\prod_{m=1 \& m \neq k}^{K^-} \left(n_m^- - 1\right)!\right) \left(n_k^- - 1\right)!}, \text{ if } k = 1, \ldots, K^- \\ \frac{\alpha^{K^-} \left(\prod_{m=1}^{K^-} \left(n_m^- - 1\right)!\right) \alpha (n_{b\ell} - 1)!}{\alpha^{K^-} \left(\prod_{m=1}^{K^-} \left(n_m^- - 1\right)!\right)}, \text{ if } k = K^- + 1 \end{cases} \\ &= \begin{cases} \frac{\left(n_k^- + n_{b\ell} - 1\right)!}{\left(n_k^- - 1\right)!}, \text{ if } k = 1, \ldots, K^- \\ \alpha\,(n_{b\ell} - 1)!, \text{ if } k = K^- + 1, \end{cases} \end{aligned} \tag{7.20}$$

where the superscript xx^- represents the appropriate quantity xx with local cluster bl excluded from the sample.

Similar to before, let $p(\boldsymbol{y}_{bl} \mid \tilde{s}_{b\ell} = k, \tilde{\boldsymbol{s}}^-, \boldsymbol{r}^-, \boldsymbol{y}^-) = \int \prod_{h \in R_{b\ell}} p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_k^{\star\star-}) p(\boldsymbol{\theta}_k^{\star\star-} \mid \tilde{s}^-, \boldsymbol{r}^-, \boldsymbol{y}^-) \mathrm{d}\boldsymbol{\theta}_k^{\star\star-}$, and let $h_0(\boldsymbol{y}_{b\ell}) = \int \prod_{h \in R_{b\ell}} p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh}) G_0(\boldsymbol{\theta}_{bh}) \mathrm{d}\boldsymbol{\theta}_{bh}$. The global cluster membership indicator $\tilde{s}_{b\ell}$, for $b = 1, \ldots, B$ and $\ell = 1, \ldots, L_b$, is drawn using its marginal conditional *a posteriori* distribution defined as

$$p(\tilde{s}_{b\ell} = k \mid \tilde{s}^-, \boldsymbol{r}^-, \boldsymbol{y}) \propto \begin{cases} \frac{\left(n_k^- + n_{b\ell} - 1\right)!}{\left(n_k^- - 1\right)!} p(\boldsymbol{y}_{bl} \mid \tilde{s}_{b\ell} = k, \tilde{\boldsymbol{s}}^-, \boldsymbol{r}^-, \boldsymbol{y}^-), & \text{for } k = 1, \ldots, K^- \\ \alpha \Gamma\left(n_{b\ell}\right) \, h_0(\boldsymbol{y}_{b\ell}), & \text{for } k = K^- + 1. \end{cases} \tag{7.21}$$

For conjugate models $p(\boldsymbol{\theta}_k^{\star\star-} \mid \tilde{s}^-, \boldsymbol{r}^-, \boldsymbol{y}^-)$ and $h_0(\boldsymbol{y}_{b\ell}) = \int \prod_{h \in R_{b\ell}} p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh}) G_0(\boldsymbol{\theta}_{bh}) \mathrm{d}\boldsymbol{\theta}_{bh}$ are available in closed form and usually are function of the summary statistics $\boldsymbol{\eta}_{b\ell}$ for $R_{b\ell}$ only, without need to access the original data $\boldsymbol{y}_b$. For example, under a normal sampling model, $p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh} = (\boldsymbol{\mu}_{bh}, \boldsymbol{\sigma}^2_{bh})) = \prod_{j=1}^p \mathrm{N}(y_{bhj} \mid \boldsymbol{\theta}_{bhj} = (\mu_{bhj}, \sigma^2_{bhj}))$, and a conjugate NIG baseline measure $G_0(\boldsymbol{\theta}) = \prod_{j=1}^p \mathrm{NIG}(\boldsymbol{\theta}_j \mid \mu_{0j}, \lambda_{0j}, a_{0j}, b_{0j})$, the complete conditional *a posteriori* distribution of $\boldsymbol{\theta}_k^{\star\star-}$ is similar to (7.16), now including all observations allocated in global cluster S_k except observations of local cluster $R_{b\ell}$. It can be evaluated using the summaries $\boldsymbol{\eta}_{b'\ell'}$ only. The marginal likelihood for $(b\ell)$ starting a new global (singleton) cluster, $h_0(\boldsymbol{y}_{b\ell}) = \int \prod_{h \in R_{b\ell}} p(\boldsymbol{y}_{bh} \mid \boldsymbol{\theta}_{bh}) G_0(\boldsymbol{\theta}_{bh}) \mathrm{d}\boldsymbol{\theta}_{bh}$, is

$$\begin{aligned} h_0(\boldsymbol{y}_{b\ell}) = \prod_{j=1}^p (2\pi)^{-\frac{n_{b\ell}}{2}} \left(\frac{\lambda_{0j}}{n_{b\ell} + \lambda_{0j}}\right)^{1/2} & \frac{\Gamma\left(\frac{n_{b\ell}}{2} + a_{0j}\right)}{\Gamma\left(a_{0j}\right)} \\ & \times \frac{b_{0j}^{a_{0j}}}{\left(b_{0j} + \frac{s^2_{b\ell j}}{2} + \frac{\lambda_{0j} n_{b\ell} \left(\bar{y}_{b\ell j} - \mu_{0j}\right)^2}{2(n_{b\ell} + \lambda_{0j})}\right)^{\frac{n_{b\ell}}{2} + a_{0j}}}, \end{aligned} \tag{7.22}$$

where $\bar{y}_{b\ell j} = \sum_{h \in R_{b\ell}} y_{bhj} / n_{b\ell}$ and $s^2_{b\ell j} = \sum_{h \in R_{b\ell}} (y_{bhj} - \bar{y}_{b\ell j})^2$ are local cluster-specific sample means and (scaled) variances. Both are functions of the summary $\boldsymbol{\eta}_{b\ell}$. The marginal likelihood when (b, ℓ) joins global cluster k, that is $p(\boldsymbol{y}_{bl} \mid \tilde{s}_{b\ell} = k, \tilde{\boldsymbol{s}}^-, \boldsymbol{r}^-, \boldsymbol{y}^-)$, is equal to (7.22), replacing μ_{0j}, λ_{0j}, a_{0j} and b_{0j} by the parameters of the complete conditional *a posteriori* distribution of $\boldsymbol{\theta}_k^{\star\star-}$. Again, only functions of the summary $\boldsymbol{\eta}_{b\ell}$ are used.

We implement *a posteriori* MCMC for $\tilde{\boldsymbol{s}}$ using (7.21). Finally, we report a point estimate for $\tilde{\boldsymbol{s}}$ using a similar scheme as before for $\boldsymbol{r}_b$. First we evaluate the *a posteriori* co-clustering probabilities $\pi^{\star\star}_{b\ell, b'\ell'} = p(\tilde{s}_{b\ell} = \tilde{s}_{b'\ell'} \mid \boldsymbol{y})$ as appropriate Monte Carlo averages. Next we define a threshold $p^{\star\star}$ and use a rule as in (7.18). We use $p^{\star\star} = 0.50$, that is, we allocate two local clusters in the same global cluster if they co-cluster in more than half of the MCMC samples.

In summary, the inputs to estimate global clusters are the sufficient statistics $\boldsymbol{\eta}_{b\ell}$ for all local clusters and a starting point for $\tilde{\boldsymbol{s}}$. The output is an estimated global partition as a set of clusters membership indicators for all local clusters, $\tilde{\boldsymbol{s}}$, which combined with $\boldsymbol{r}$ provide the global cluster for each observation.

In summary, Algorithm 3 describes the SNOB procedure.

Algorithm 3 SNOB algorithm

```
1. Local clustering of shards $b = 1, \dots, B$
   Input: data $\boldsymbol{y}_b$ for the $b$-th shard, a starting point for $\boldsymbol{r}_b$, and a threshold $p^\star$
   Output: Local cluster membership indicators $\boldsymbol{r}_b$ and sufficient statistics of each local
   cluster of $b$-th shard, $\boldsymbol{\eta}_{b\ell}$

   (a) Draw $r_{bh}$, for $h = 1, \dots, I_b$, $M$ times, where $M$ defines the MCMC chain size.
   (b) Calculate a posteriori co-clustering probability for each pair of observations.
   (c) Compute final estimate of $\boldsymbol{r}_b$ and the sufficient statistics for each local cluster.

2. Global clustering
   Input: sufficient statistics $\boldsymbol{\eta}_{b\ell}$ for each local cluster, a starting point for $\tilde{\boldsymbol{s}}$, and a threshold
   $p^{\star\star}$ to define global clusters
   Output: Global cluster membership indicators $\tilde{\boldsymbol{s}}$

   (a) Draw $\tilde{s}_{b\ell}$, for $b = 1, \dots, B$ and $\ell = 1, \dots, L_b$, $M$ times, where $M$ defines the MCMC
       chain size.
   (b) Calculate a posteriori co-clustering probability for each pair of local cluster.
   (c) Compute final estimate of $\tilde{\boldsymbol{s}}$.
```

7.4 Simulation

We apply the proposed PRC and SNOB approach to clustering in a simulation example and two benchmark data sets. R codes used to carry out PRC and SNOB are available in the appendices of this chapter. The first data set is far from the large sample size for which we specifically developed PRC and SNOB. We use it because the analyses provide a useful comparison with known (non-big data) clustering algorithms. The first data set is the iris flower data set or Fisher's *iris data* (Fisher, 1936). It consists of 50 samples from each of $K = 3$ species of iris (iris setosa, iris virginica and iris versicolor), for a total sample size of $n = 150$. Four features ($p = 4$) were measured from each sample: the length and the width of the sepals and petals, in centimeters. The second data set, *wine quality data* (Cortez *et al*, 2009), represents a white wine sample collection of Vinho Verde wines. It consists of $n = 4898$ samples and $p = 11$ physiochemical variables: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and a quality rating. The quality rating is based on a sensory taste test carried out by at least three sommeliers and scaled in 11 quality classes from 0 - very bad to 10 - very excellent. We combine wine quality classes into $K = 3$ levels: low (rating < 6), moderate (rating equal to 6) and high (rating > 6) as (Cortez *et al*, 2009). We also simulate another large data set with $n = 5000$ observations. The simulation truth is a mixture of $K = 4$ multivariate normal distributions ($p = 8$), with parameters chosen to mimic the data in the application to genes (see the next section).

We carry out clustering for these three data sets using DP-means (Kulis & Jordan, 2012), DBSCAN (Ester *et al*, 1996), SUGS (Wang & Dunson, 2011) and an EM algorithm for Bayesian regularization for multivariate normal mixtures proposed by Fraley & Raftery (2007). See the introduction for a brief review of these methods. For all methods we cluster the de-correlated transformed data $\tilde{\boldsymbol{y}} = \left(A^{-1}\boldsymbol{y}^T\right)^T$, where AA' is a (Cholesky) factorization of the empirical covariance matrix of $\boldsymbol{y}$.

For SUGS, PRC and SNOB schemes we fix the total mass parameters as $\alpha = 1$ and the hyperparameters of the NIG baseline measure as $\mu_{0j} = 0$, $\lambda_{0j} = 0.01$, $a_{0j} = 5$ and $b_{0j} = 3$, for $j = 1, \ldots, p$, implying that, a priori, $E(\sigma^2) = 0.75$, $SD(\sigma^2) = 0.45$, $E(\mu \mid \sigma^2) = 0$ and $Var(\mu \mid \sigma^2) = 100\sigma^2$. The hyperparameters for σ^2 are chosen because of the unit sample standard deviation of the pre-processed data. For PRC, we use weights $\omega_i = 1/(1 + \alpha\sqrt{i}) = 1/(1 + \sqrt{i})$, for $i = 1, \ldots, n$, that is, weights that vanish slightly slower than nominal weight $\omega_i = 1/(i + \alpha) = 1/(1 + i)$.

We run DP-means for different numbers of clusters. For choosing the best DP-means clustering, we use the summary SSE/SST for each value of K, where SSE is the error sum of squares and SST is the total sum of squares. The best value of K is the smallest value before SSE/SST starts to sharply rise. For the iris, wine and simulated data set we choose clustering with $K = 8$, 19 and 10 clusters based on SSE/SST=0.32, 0.59 and 0.54, respectively. R code for DP-means clustering is available at `https://github.com/johnmyleswhite/bayesian_nonparametrics/tree/master/code/dp-means`.

For DBSCAN, we set the minimum number of neighbors a point should have to be included into a cluster to 5 for all data sets and choose the maximum distance for a point be considered a part of a cluster as suggested by Ester *et al.* (1996). For the iris, wine and simulated data set, the estimated partition includes $K = 2$, 1 and 5 non-singleton clusters and 23, 0 and 135 singleton clusters, respectively. The DBSCAN algorithm is available as an R-package `dbscan` (Ester *et al*, 1996).

The EM algorithm proposed by Fraley & Raftery (2007) classifies data using finite mixture models with a random number of clusters and allows clusters with different sizes, shapes and orientations. They propose to select the best clustering using Bayesian information criterion (BIC). For the iris, wine and simulated data sets, the best partition has $K = 2$, 16 and 4 clusters, respectively. This EM algorithm is available as the R-package `mclust` (Fraley & Raftery, 2002; Fraley *et al*, 2012).

For SUGS and PRC we used 10 permutations of the data set and report the permutation with maximum PML. SUGS estimates K to be 3, 19 and 5 for the iris, wine and simulated data set, respectively. We note sticky cluster allocation in SUGS – for all three data sets most data are allocated in the first and second clusters only. Under PRC the estimate for K is 5, 19 and 12 clusters for the iris, wine and simulated data set, respectively.

We do not use subsamples in SNOB for the iris data set since it is already so small and only carry out the step of estimating local clusters. The wine and simulated data set are broken into $B = 5$ shards with same size. For all MCMC chains we run 6000 iterations, discarding the first 1000 iterations and thinning out to save only every 5-th iteration. All chains is initialized with singleton clusters. SNOB estimates $K = 10$ clusters for the iris data set (5 of them being singleton clusters), $K = 18$ global clusters for the wine data and $L_b = 4$ local clusters for each shard and $K = 4$ global clusters for the simulated data. The estimated number of local clusters in wine data is 23, 23, 23, 21 and 28 for each shard.

As our interest is to identify clusters and correctly classify the data, we use misclassification rate to compare methods. We define misclassification rate as

$$MCR = \frac{\sum_{i=1}^{n} I(s_i^{(T)} = s_i)}{n} 100, \tag{7.23}$$

where $s_i^{(T)}$ is the true cluster membership indicator. When the clusters' label of $\boldsymbol{s}$ is switched compared with $\boldsymbol{s}^{(T)}$, we relabel $\boldsymbol{s}$ and choose the labels' permutation which minimize MCR. When any method estimates $K = 1$ we record a misclassification rate of 100%.

Table 7.1 shows the misclassification rate of each method for each data set. We observe that PRC and SNOB have favorable misclassification rates for all analyzed data sets. DP-means is slightly better than PRC and SNOB for the iris data set. However it identifies a higher number of clusters. For the simulated data set, the EM algorithm is comparable to SNOB and very similar to PRC, but less favorable for the iris data set.

Comparing only PRC and SNOB, we observe that PRC usually reports a larger number of clusters, but most observations are concentrated in a smaller number of groups while under SNOB the clusters are more balanced.

Table 7.1: Misclassification rate of clustering.

	DP-means	DBSCAN	SUGS	EM	PRC	SNOB
Iris data set	18.0	38.7	34.7	33.0	21.3	28.7
Wine data set	51.2	100.0	52.8	47.9	50.8	45.0
Simulated data set	25.4	2.7	26.0	0.0	2.8	0.0

In terms of computation time, the fastest algorithms are PRC and SUGS since they only read each observation once (except for a small number of repetitions to evaluate alternative permutations to mitigate order dependence). Although SNOB involves MCMC simulation for each shard and another MCMC simulation to find the global clusters, it remains computation efficient since the shards are small, and can even be analyzed in parallel. For estimating global clustering we need to save only the sufficient statistics.

7.5 Clustering genes by their GE-GE interactions

7.5.1 Data

Our goal is clustering $n = 19,304$ genes according to their GE-GE interaction pattern with other genes. The data set is from Zodiac (Zhu *et al*, 2015), an online search engine for visualizing gene interaction based on statistical analysis of data collected by The Cancer Genome Atlas (TCGA). Statistical inference in Zodiac is based on auto-logistic models and the coefficients between pairs of gene-specific latent indicators are used to describe the strength of gene-gene interactions. We do not make use of the detail methods of how Zodiac derives them (using a model proposed in Mitra et al. 2013). For the upcoming discussion these coefficients are the data. That is, for each gene we record a histogram of $19,303$ gene-gene interaction coefficients, and the formal goal becomes to identify a cluster of high interaction genes.

Available clustering algorithms, as well as the proposed PRC and SNOB scheme are developed to cluster p-dimensional outcomes, rather than histograms (or distributions). We therefore pre-process the data, replacing the histograms of gene-gene interaction coefficients by an approximation with high Jacobi polynomials, that is, essentially by a sequence of empirical moments. This choice is motivated by Arbel *et al.* (2015) who show examples where distributions are well fitted by high order Jacobi polynomials (with the first seven moments). We summarize the histogram of significant coefficients by their first seven moments $\boldsymbol{M}_i = (M_{i1}, ..., M_{i7})$ and add the frequency f_{i0} of nonsignificant auto-logistic coefficients. Significant coefficients are identified by thresholding false discovery rate at 10%. See Mitra *et al.* (2013) for details. The r-th empirical moment is defined as $M_{ir} = \sum_{j=1}^{c_i} y_{ij}^r \Big/ c_i$ where c_i is the number of significant coefficients for i-th gene. In summary, we define summarized data as $\boldsymbol{y}_i^\star = (\boldsymbol{M}_i, \log(f_{i0}))$.

Finally, we implement PRC and SNOB clustering with transformed data $\tilde{\boldsymbol{y}}$, where $\tilde{\boldsymbol{y}} = \left(A^{-1}\boldsymbol{y}^{\star T}\right)^T$ and A is the lower triangular matrix of a Cholesky decomposition of the empirical covariance matrix of the $\boldsymbol{y}^\star$. For PRC and SNOB we fix the total mass parameters as $\alpha = 1$ and the hyperparameters of the NIG baseline measure as $\mu_{0j} = 0$, $\lambda_{0j} = 0.01$, $a_{0j} = 5$ and $b_{0j} = 3$, for $j = 1, \ldots, 8$.

7.5.2 PRC results

We implement the PRC algorithm with weights $\omega_i = 1/(1 + \alpha\sqrt{i}) = 1/(1 + \sqrt{i})$, which decrease slightly slower than nominal weights $\omega_i = 1/(i + \alpha) = 1/(1 + i)$. To mitigate the dependency on the order, we use 10 permutations of the data set and select the one with the largest PML value.

The final number of components in (7.3) is 104, but the genes are allocated to only 60 of these. Only 6 clusters include more than 1% of all genes and together they represent 96.4% of the total number of genes. The remaining 3.6% genes are allocated to the remaining 54 small clusters. Table 7.2 shows the frequency (%) of the 6 most representative clusters. We observe one larger cluster with frequency close to 0.69, called cluster 2, and two moderate size clusters, called clusters 1 and 3, which frequency is 0.07 and 0.15, respectively. Similar to inference for the simulated data set, we find a large number of clusters, but most observations are concentrated in a small number of groups.

Table 7.2: Frequency of the most 6 representative PRC clusters.

Cluster	%	Cluster	%	Cluster	%
1	6.5	3	14.6	5	1.3
2	69.8	4	1.5	6	2.7

Figure 7.1 shows the 6 most representative PRC clusters. With the aim of identifying groups of gene with the systematically highest interactions with other genes, we represent each cluster by a boxplot of the percentiles of the (original) interaction coefficients, showing the 5%, 10%, 25%, 50% (median), 75%, 90% and 95% percentiles and the $10f_{0i}$. Figure 7.1 shows the boxplot of these 8 summary statistics arranged by clusters. Genes in cluster 3, which represent 14.6% of all genes, have only highly positive interaction coefficients with other genes. Genes of cluster 3 are of most interest to investigators. Cluster 1, which represent almost 7% of genes, is characterized by genes that have negative and positive significant interactions. Cluster 2 which is the largest cluster consists of genes with positive and moderately significant coefficients. The remaining clusters 4, 5 and 6 include the genes with the lowest interactions since all percentiles are close to zero.

7.5.3 SNOB results

In SNOB, we split the data set into $B = 20$ shards. For all MCMC chains we run 6000 iterations, discarding the first 1000 iterations and thinning out to save only every 5-th iteration. MCMC chains is initialized with singleton clusters.

The average number of local clusters in each shard is 17 with a total of 341 local clusters. The number of global clusters is $K = 26$, but only 12 clusters have more than 1% of all genes, and together they represent 97.3% of all genes. The remaining 2.7% genes (around 500 genes) are distributed across the remaining 14 smaller global clusters. Table 7.3 shows the frequency (%) of the 12 most representative clusters. We observe two larger clusters with frequency higher than 0.20 (clusters 2 and 4), and two moderate size clusters (1 and 5).

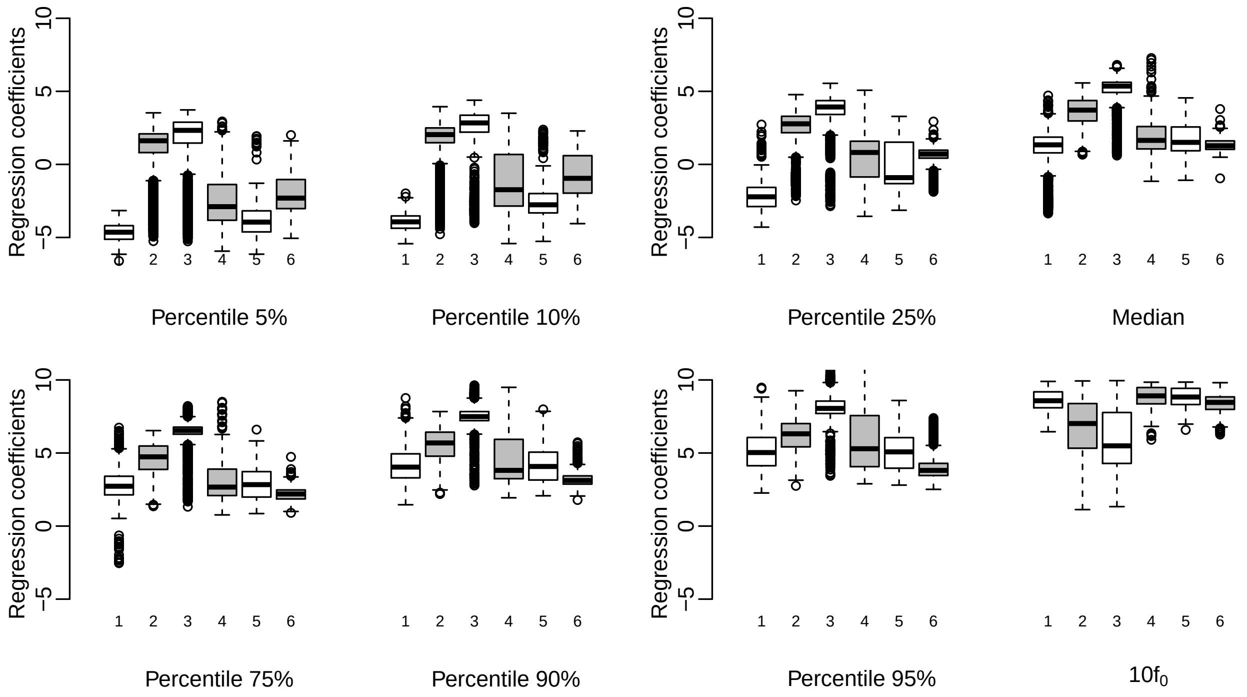

Percentile 5%
Percentile 10%
Percentile 25%
Median
Percentile 75%
Percentile 90%
Percentile 95%
$10f_0$

Figure 7.1: PRC clusters. Each boxplot shows the distribution of the respective quantile across all genes in the cluster.

Table 7.3: Frequency of the most 12 representative SNOB clusters.

Cluster	%	Cluster	%	Cluster	%	Cluster	%
1	10.7	4	21.4	7	3.6	10	1.2
2	2.4	5	7.3	8	1.8	11	1.0
3	42.4	6	2.2	9	2.3	12	1.1

Figure 7.2 shows the boxplot of percentiles of the original auto-logistic gene-gene interaction coefficients, arranged by clusters. The boxplots shows percentiles for 5%, 10%, 25%, 50% (median), 75%, 90% and 95%, plus $10f_{0i}$. Genes in clusters 4 and 9, which represent 23.7% of genes, have only highly positive significant interactions with other genes. These clusters are the desired report of genes of interest. Clusters 5 and 6, which represent almost 10% of genes, are characterized by genes that have negative and positive significant interactions. Cluster 8 consists of genes with the largest number of significant coefficients. The remaining clusters (1, 2, 10, 11 and 12) have the genes with the lowest interactions since all percentiles are close to zero.

7.6 Discussion

We introduced two nonparametric model-based methods to cluster big data and identify groups across thousands of genes that have similar patterns of genetic interaction with other genes. The first scheme, PRC, is an incremental clustering algorithm based on predictive

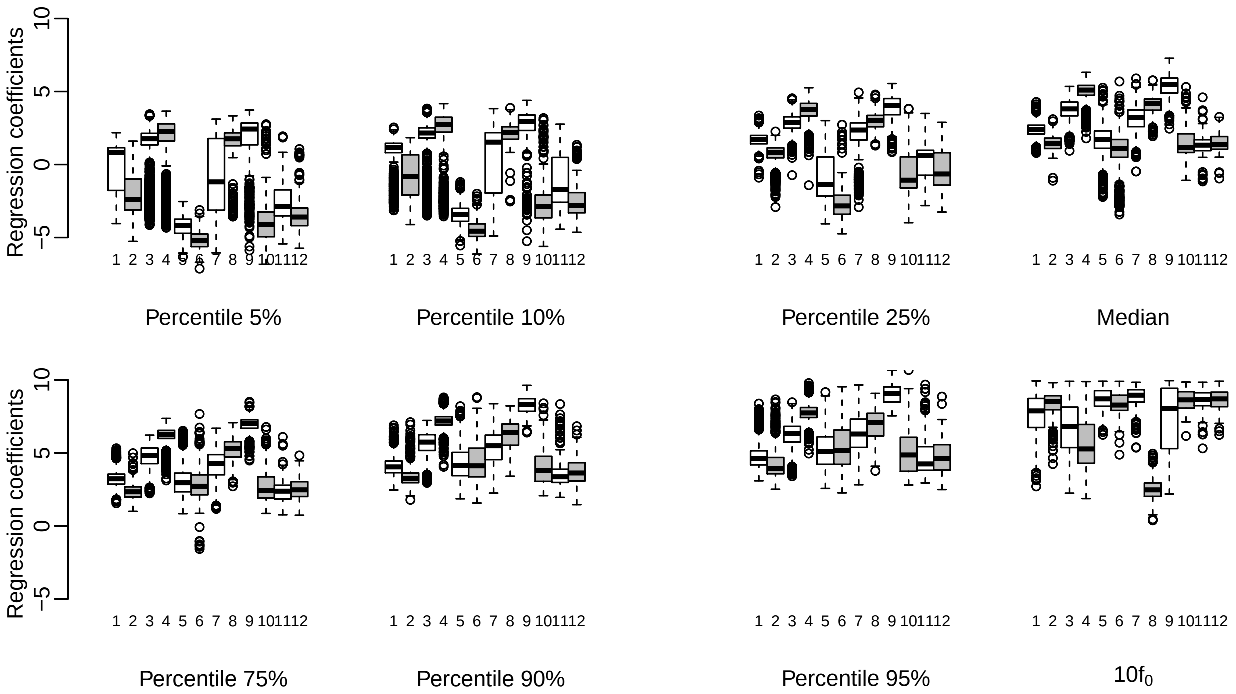


Figure 7.2: SNOB clusters. Each boxplot shows the distributions of the respective quantile across all genes in the cluster.

recursion algorithm of Newton *et al.* (1998). It is an approximate method which requires a single cycle of simple deterministic calculations for each observation under study. The other method is a distributed computing algorithm, called SNOB. It is an exact method that divides the data in smaller groups across multiple machines and identifies local clusters. In a second step, the method combines the local clusters to find global clusters in a MapReduce or Hadoop framework. The algorithm is simulation exact, in the sense that it uses probabilities under the *a posteriori* random partition of a DP mixture model. The second step is implemented as one step of a Markov chain Monte Carlo simulation. It needs to only access the sufficient statistics of the earlier determined local clusters.

Both methods show good performance to classify simulated and two benchmark smaller data sets. Compared with traditional clustering algorithms including k-means, DBSCAN, SUGS and the EM algorithm, PRC and SNOB have favorable misclassification rates. The clusters that are reported under the PRC and SNOB methods have clear distinct features; we can interpret the desired groups of gene with the highest interactions with other genes.

PRC is more computation efficient than SNOB since it does not require full Monte Carlo simulations. However, in the examples we found that it estimates more concentrated clusters. SNOB estimates more balanced clusters. Also, it is easier to implement with nonconjugate distributions. For the PRC algorithm, the required normalizing constants complicate the implemenation in nonconjugate problems.

7.7 Appendices

7.7.1 R codes to carry out PRC for simulated data set

In this section, we show R codes for clustering the simulated data set using PRC method.

```
#
#########################
# useful functions
#########################
#
############
# sample n=1 from a Multinomial(p) distribution
############
rDiscreta<-function(p){
 u<-runif(1)
 P<-cumsum(p)
 val<-sum(P<u)+1
 val}
#
############
# calculate mean, lambda, alfa and beta of the posterior distribution given
# one observation yi when the variance of Yi is sigma^2/lambda
############
postparamvarunknown<-function(yi,mui,lambdai,alfai,betai){
 munew<-(yi+(lambdai*mui))/(lambdai+1)
 lambdanew<-lambdai+1
 agamnew<-alfai+(1/2)
 bgamnew<-betai+((lambdai*(yi-mui)**2)/(lambdai+1))/2
list(munew,lambdanew,agamnew,bgamnew)}
#
############
# calculate the density function of a t distribituion (yij, ni, mu, sigma)
############
dstudentt<-function(yij,ni,mu,desvio){
 dens<-exp(lgamma((ni+1)/2)-lgamma(ni/2))/(sqrt(pi*ni)*desvio)*((1+(1/ni)*
((yij-mu)/desvio)**2)**(-(ni+1)/2))
 return(dens)}
#
```

```
############
# calculate mahalanobis distance between two multidim. normal distributions
############
MALAfunc<-function(mu0,mu1,sigma0,sigma1){
 sigmap<-(sigma0+sigma1)/2
 distmu<-t(mu1-mu0)
 MALA<-sqrt(distmu%*%diag(1/sigmap)%*%t(distmu))}
#
#########################
# simulated data set
#########################
#
set.seed(100)
muvec<-matrix(c(0,0.5,1.1,-11.6,-8,1.7,3.1,1.9,-0.5,1.2,1.7,-11.7,-6.3,0.1,
2.3,2.4,0.5,-1.2,2.7,-11.6,-7,0.7,2.6,2.2,1,1.2,3.7,-11.6,-7.7,1.2,2.9,2.1),
nrow=4,byrow=TRUE)
sigmavec<-matrix(c(0.002140429, 0.001531872, 0.225287998, 0.053739488,
0.126138910, 0.112735647, 0.133862779, 0.002140429, 0.001798247,
0.006215775, 0.315767179, 0.231431127, 0.472012711,0.129508527,
1.008442513, 0.001798247,0.002172256, 0.015160350, 2.142055552,
0.885205346, 0.549862208, 3.155843303, 0.887398965, 0.002172256,
0.004303235, 0.011570390, 1.311111853, 1.026574670, 1.325584473,
0.319953786, 1.874197742, 0.004303235),nrow=4,byrow=TRUE)
Nt<-5000
compon<-numeric(Nt)
Y<-matrix(0,ncol=ncol(muvec),nrow=Nt)
for (i in 1:Nt){
 compon[i]<-rDiscreta(rep(1/nrow(muvec),nrow(muvec)))
 Y[i,]<-rnorm(ncol(muvec),muvec[compon[i],],sqrt(sigmavec[compon[i],]))}
#
covmat<-cov(Y)
decomp<-t(chol(covmat))
round(decomp%*%t(decomp),5)==round(covmat,5)
Ytransf<-t(solve(decomp)%*%t(Y))
media<-apply(Ytransf,2,mean)
for (j in 1:ncol(Ytransf)) Ytransf[,j]<-Ytransf[,j]-media[j]
Y<-Ytransf
Nt<-nrow(Y)
```

```
#
#########################
# Run PRC
#########################
#
library(compiler)
enableJIT(3)
#
Yorig<-Y
componorig<-compon
replic<-10
vero<-numeric(replic)
#
for (mc in 1:replic){
 #
 set.seed(100)
 alpha<-1
 weight<-1/((1+alpha*sqrt(1:Nt)))
 p<-ncol(Y) # dimension of multivariate distribution
 muvec<-matrix(0,nrow=p,ncol=1) # G0 mean
 lambdavec<-matrix(0.1,nrow=p,ncol=1) # G0 variance
 alfavec<-matrix(5,nrow=p,ncol=1) # G0
 betavec<-matrix(3,nrow=p,ncol=1) # G0
 mixweigold<-1
 #
 dcomp<-sqrt(qchisq(.75, df=p))
 for (j in 2:(Nt+1)){
  for (k in 1:ncol(muvec)){
   newpar<-postparamvarunknown(Y[j-1,],muvec[,k],lambdavec[,k],alfavec[,k],
betavec[,k])
   muvec<-cbind(muvec,newpar[[1]])
   lambdavec<-cbind(lambdavec,newpar[[2]])
   alfavec<-cbind(alfavec,newpar[[3]])
   betavec<-cbind(betavec,newpar[[4]])}
  mixweigold<-c(mixweigold*(1-weight[j-1]),mixweigold*weight[j-1])
  sigmavec<-matrix(0,ncol=ncol(muvec),nrow=nrow(muvec))
  for (i in 1:nrow(muvec)){
   for (k in 1:ncol(muvec)) sigmavec[i,k]<-1/(rgamma(1,alfavec[i,k],
betavec[i,k]))/lambdavec[i,k]}
  #
  if (j>2){
   MALA<-matrix(0,ncol=ncol(muvec)-1,nrow=ncol(muvec)-1)
   for (i in 2:ncol(muvec)){
    for (l in 2:ncol(muvec)){
     MALA[i-1,l-1]<-MALAfunc(muvec[,i],muvec[,l],sigmavec[,i],sigmavec[,l])}}
   for (i in 1:(ncol(muvec)-1)) MALA[i,i]<-100
   #
   while (min(MALA) < dcomp & ncol(muvec)>3){
    index<-which(MALA==min(c(MALA)), arr.ind=TRUE)[1,]
    i<-index[order(mixweigold[index])[2]]+1
    l<-index[order(mixweigold[index])[1]]+1
    muvec<-muvec[,-l]
    lambdavec<-lambdavec[,-l]
    alfavec<-alfavec[,-l]
    betavec<-betavec[,-l]
    sigmavec<-sigmavec[,-l]
    mixweigold<-mixweigold[-l]
    MALA<-MALA[,-(l-1)]
    MALA<-MALA[-(l-1),]
    mixweigold<-mixweigold/sum(mixweigold)}}}
 #
 dcomp<-sqrt(qchisq(.95, df=p))
 MALA<-matrix(0,ncol=ncol(muvec),nrow=ncol(muvec))
 for (i in 1:ncol(muvec)){
  for (l in 1:ncol(muvec)){
   MALA[i,l]<-MALAfunc(muvec[,i],muvec[,l],sigmavec[,i],sigmavec[,l])}}
 for (i in 1:ncol(muvec)) MALA[i,i]<-100
 while (min(MALA) < dcomp & ncol(muvec)>3){
  index<-which(MALA==min(c(MALA)), arr.ind=TRUE)[1,]
  if (min(index)==1){
   i<-max(index)
   l<-min(index)}
  if (min(index)!=1){
   i<-index[order(mixweigold[index])[2]]
   l<-index[order(mixweigold[index])[1]]}
  muvec<-muvec[,-l]
  lambdavec<-lambdavec[,-l]
  alfavec<-alfavec[,-l]
  betavec<-betavec[,-l]
  sigmavec<-sigmavec[,-l]
  mixweigold<-mixweigold[-l]
  MALA<-MALA[,-l]
  MALA<-MALA[-l,]
  mixweigold<-mixweigold/sum(mixweigold)}
 #
 ####### calculating the pseudo-marginal likelihood
 #
 prob_fim<-numeric()
 prob<-numeric()
 Sj<-numeric(Nt)
 for (j in 1:Nt){
  for (k in 1:length(mixweigold)){
   ni<-2*alfavec[,k]
   mu<-muvec[,k]
   desvio<-sqrt((2*betavec[,k]*(1+lambdavec[,k]))/(ni*lambdavec[,k]))
   prob[k]<-mixweigold[k]*prod(dstudentt(Y[j,],ni,mu,desvio))}
  Sj[j]<-which((prob/sum(prob))==max((prob/sum(prob))))
  prob_fim<-c(prob_fim,prob[Sj[j]])}
 vero[mc]<-sum(log(prob_fim))
 print(table(compon,Sj))
 #
 set.seed(310+mc)
 neworder<-sample(1:nrow(Yorig),nrow(Yorig))
 Y<-Yorig[neworder,]
 compon<-componorig[neworder]
 Nt<-nrow(Y)}
#
best<-which(vero==max(vero))-1
set.seed(310+best)
neworder<-sample(1:nrow(Yorig),nrow(Yorig))
Y<-Yorig[neworder,]
compon<-componorig[neworder]
Nt<-nrow(Y)
#
```

```
########### run again the model with best order
#
set.seed(100)
alpha<-1
weight<-1/((1+alpha*sqrt(1:Nt)))
p<-ncol(Y) # dimension of multivariate distribution
muvec<-matrix(0,nrow=p,ncol=1) # G0 mean
lambdavec<-matrix(0.1,nrow=p,ncol=1) # G0 variance
alfavec<-matrix(5,nrow=p,ncol=1) # G0
betavec<-matrix(3,nrow=p,ncol=1) # G0
mixweigold<-1
#
dcomp<-sqrt(qchisq(.75, df=p))
for (j in 2:(Nt+1)){
 for (k in 1:ncol(muvec)){
  newpar<-postparamvarunknown(Y[j-1,],muvec[,k],lambdavec[,k],alfavec[,k],
betavec[,k])
  muvec<-cbind(muvec,newpar[[1]])
  lambdavec<-cbind(lambdavec,newpar[[2]])
  alfavec<-cbind(alfavec,newpar[[3]])
  betavec<-cbind(betavec,newpar[[4]])}
 mixweigold<-c(mixweigold*(1-weight[j-1]),mixweigold*weight[j-1])
 sigmavec<-matrix(0,ncol=ncol(muvec),nrow=nrow(muvec))
 for (i in 1:nrow(muvec)){
  for (k in 1:ncol(muvec)) sigmavec[i,k]<-1/(rgamma(1,alfavec[i,k],
betavec[i,k]))/lambdavec[i,k]}
 #
 if (j>2){
  MALA<-matrix(0,ncol=ncol(muvec)-1,nrow=ncol(muvec)-1)
  for (i in 2:ncol(muvec)){
   for (l in 2:ncol(muvec)){
    MALA[i-1,l-1]<-MALAfunc(muvec[,i],muvec[,l],sigmavec[,i],sigmavec[,l])}}
  for (i in 1:(ncol(muvec)-1)) MALA[i,i]<-100
  #
  while (min(MALA) < dcomp & ncol(muvec)>3){
   index<-which(MALA==min(c(MALA)), arr.ind=TRUE)[1,]
   i<-index[order(mixweigold[index])[2]]+1
   l<-index[order(mixweigold[index])[1]]+1
   muvec<-muvec[,-l]
   lambdavec<-lambdavec[,-l]
   alfavec<-alfavec[,-l]
   betavec<-betavec[,-l]
   sigmavec<-sigmavec[,-l]
   mixweigold<-mixweigold[-l]
   MALA<-MALA[,-(l-1)]
   MALA<-MALA[-(l-1),]
   mixweigold<-mixweigold/sum(mixweigold)}}}
#
dcomp<-sqrt(qchisq(.95, df=p))
MALA<-matrix(0,ncol=ncol(muvec),nrow=ncol(muvec))
for (i in 1:ncol(muvec)){
 for (l in 1:ncol(muvec)){
  MALA[i,l]<-MALAfunc(muvec[,i],muvec[,l],sigmavec[,i],sigmavec[,l])}}
for (i in 1:ncol(muvec)) MALA[i,i]<-100
while (min(MALA) < dcomp & ncol(muvec)>3){
 index<-which(MALA==min(c(MALA)), arr.ind=TRUE)[1,]
 if (min(index)==1){
  i<-max(index)
  l<-min(index)}
 if (min(index)!=1){
  i<-index[order(mixweigold[index])[2]]
  l<-index[order(mixweigold[index])[1]]}
 muvec<-muvec[,-l]
 lambdavec<-lambdavec[,-l]
 alfavec<-alfavec[,-l]
 betavec<-betavec[,-l]
 sigmavec<-sigmavec[,-l]
 mixweigold<-mixweigold[-l]
 MALA<-MALA[,-l]
 MALA<-MALA[-l,]
 mixweigold<-mixweigold/sum(mixweigold)}
#
prob_fim<-numeric()
prob<-numeric()
for (j in 1:Nt){
 for (k in 1:length(mixweigold)){
  ni<-2*alfavec[,k]
  mu<-muvec[,k]
  desvio<-sqrt((2*betavec[,k]*(1+lambdavec[,k]))/(ni*lambdavec[,k]))
  prob[k]<-mixweigold[k]*prod(dstudentt(Y[j,],ni,mu,desvio))}
 prob_fim<-rbind(prob_fim,prob/sum(prob))}
Sj<-numeric(Nt)
for (j in 1:Nt) Sj[j]<-which(prob_fim[j,]==max(prob_fim[j,]))
table(compon,Sj)
#
# Updating parameters
#
K<-ncol(muvec)
Dim<-ncol(Y)
mu0<-matrix(0,ncol=1,nrow=Dim) # G0
lambda0<-matrix(0.1,nrow=Dim,ncol=1)
alfa0<-matrix(5,nrow=Dim,ncol=1)
beta0<-matrix(3,nrow=p,ncol=1)
#
nk<-numeric(K)
ykmean<-matrix(0,nrow=Dim,ncol=K)
ykvar<-matrix(0,nrow=Dim,ncol=K)
mupost<-matrix(0,nrow=Dim,ncol=K)
lambdapost<-matrix(0,nrow=Dim,ncol=K)
agampost<-matrix(0,nrow=Dim,ncol=K)
bgampost<-matrix(0,nrow=Dim,ncol=K)
for (k in 1:K){
 nk[k]<-sum(Sj==k)
 if (nk[k]==0){
  ykmean[,k]<-matrix(0,ncol=1,nrow=Dim)
  ykvar[,k]<-matrix(0,ncol=1,nrow=Dim)}
 if (nk[k]>1){
  ykmean[,k]<-apply(Y[which(Sj==k),],2,mean)
  ykvar[,k]<-apply(Y[which(Sj==k),],2,var)*(nk[k]-1)}
 if (nk[k]==1){
  ykmean[,k]<-Y[which(Sj==k),]
  ykvar[,k]<-matrix(0,ncol=1,nrow=Dim)}
 mupost[,k]<-(ykmean[,k]*nk[k]+(lambda0*mu0))/(lambda0+nk[k])
 lambdapost[,k]<-lambda0+nk[k]
 agampost[,k]<-alfa0+(nk[k]/2)
 bgampost[,k]<-beta0+(ykvar[,k]+((nk[k]*lambda0*(ykmean[,k]-mu0)**2)/
(lambda0+nk[k])))/2}
muvec<-mupost
lambdavec<-lambdapost
alfavec<-agampost
betavec<-bgampost
sigma2vec<-matrix(0,ncol=ncol(agampost),nrow=nrow(agampost))
for (i in 1:nrow(sigma2vec)){
 for (j in 1:ncol(sigma2vec)) sigma2vec[i,j]<-1/rgamma(1,alfavec[i,j],
betavec[i,j])}
#
prob_fim<-numeric()
prob<-numeric()
for (j in 1:Nt){
 for (k in 1:length(mixweigold)) prob[k]<-mixweigold[k]*
prod(dnorm(Y[j,],muvec[,k],sqrt(sigma2vec[,k])))
 prob_fim<-rbind(prob_fim,prob/sum(prob))}
Sj<-numeric(Nt)
for (j in 1:Nt) Sj[j]<-which(prob_fim[j,]==max(prob_fim[j,]))
table(compon,Sj)
```

7.7.2 R codes to carry out SNOB for simulated data set

In this section, we show R codes for clustering the simulated data set using SNOB method.

```
#
#########################
# useful functions
#########################
#
############
# sample n=1 from a Multinomial(p) distribution
############
rDiscreta<-function(p){
 u<-runif(1)
 P<-cumsum(p)
 val<-sum(P<u)+1
 val}
#
############
# calculate the density function of a t distribituion (yij, ni, mu, sigma)
############
dstudentt<-function(yij,ni,mu,desvio){
 dens<-exp(lgamma((ni+1)/2)-lgamma(ni/2))/(sqrt(pi*ni)*desvio)*((1+(1/ni)*
((yij-mu)/desvio)**2)**(-(ni+1)/2))
 return(dens)}
#
############
# compute mean, lambda, alpha and beta of the posterior distribution
# given one observation yi when the variance of Yi is sigma^2
############
postparamvarunknown<-function(yi,mui,lambdai,alfai,betai){
 munew<-(yi+(lambdai*mui))/(lambdai+1)
 lambdanew<-lambdai+1
 agamnew<-alfai+(1/2)
 bgamnew<-betai+((lambdai*(yi-mui)**2)/(lambdai+1))/2
 list(munew,lambdanew,agamnew,bgamnew)}
#
############
# compute mean, lambda, alfa and beta of the posterior distribution
# given many observations yi when the variance of Yi is sigma^2
############
postparamvarunyis<-function(yi,mui,lambdai,alfai,betai,nki){
 if (nki>1){
  ykmean<-apply(yi,2,mean)
  yksumsquar<-apply(yi**2,2,sum)}
 if (nki==1){
  ykmean<-yi
  yksumsquar<-yi**2}
 munew<-(ykmean*nki+lambdai*mui)/(lambdai+nki)
 lambdanew<-lambdai+nki
 agamnew<-alfai+(nki/2)
 s2<-yksumsquar-(nki*ykmean**2)
 bgamnew<-betai+(s2+((nki*lambdai*(ykmean-mui)**2)/(lambdai+nki)))/2
 list(munew,lambdanew,agamnew,bgamnew,ykmean,yksumsquar)}
```

```
#
############
# compute the marginalized likelihood of yi
############
dmarglikeli<-function(mlkj,media,varia,lambda,agam,bgam,mu){
 logdens<-(lgamma(mlkj/2+agam)-lgamma(agam))+((-mlkj/2)*log(2*pi))+
(0.5*(log(lambda)-log(mlkj+lambda)))+(agam*log(bgam))+((-(mlkj/2+agam))*
log(bgam+(varia/2)+((lambda*mlkj*((media-mu)**2))/(2*(mlkj+lambda)))))
 return(logdens)}
#
############
# transform data using Cholesky decomposition
############
transfdata<-function(Y){ # Y is a matrix: rows represent different observations
# and columns represent different variables
 covmat<-cov(Y)
 decomp<-t(chol(covmat))
 Ytransf<-t(solve(decomp)%*%t(Y))
 media<-apply(Ytransf,2,mean)
 for (j in 1:ncol(Ytransf)) Ytransf[,j]<-Ytransf[,j]-media[j]
 return(Ytransf)}
#
############
# Gibbs to identify local cluster for each batch
############
Gibbs_local<-function(Y,Sj,burnin,amostrasfin,saltos,alpha,mu0,lambda0,
alfa0,beta0){
 Nt<-nrow(Y)
 Dim<-ncol(Y)
 AmostrasTotal<-burnin+amostrasfin*saltos
 nk<-table(Sj)
 K<-length(nk)
 ni0<-2*alfa0
 mu0<-mu0
 desvio0<-sqrt((2*beta0*(1+lambda0))/(ni0*lambda0))
 #
 ykmean<-matrix(0,nrow=Dim,ncol=K)
 yksumsquar<-matrix(0,nrow=Dim,ncol=K)
 mupost<-matrix(0,nrow=Dim,ncol=K)
 lambdapost<-matrix(0,nrow=Dim,ncol=K)
 alfapost<-matrix(0,nrow=Dim,ncol=K)
 betapost<-matrix(0,nrow=Dim,ncol=K)
 for (k in 1:K){
  param<-postparamvarunyis(Y[which(Sj==k),],mu0,lambda0,alfa0,beta0,nk[k])
  mupost[,k]<-param[[1]]
  lambdapost[,k]<-param[[2]]
  alfapost[,k]<-param[[3]]
  betapost[,k]<-param[[4]]
  ykmean[,k]<-param[[5]]
  yksumsquar[,k]<-param[[6]]}
 #
 for (int in 1:AmostrasTotal){
  cat('\n',int)
 for (i in 1:Nt){
  nk<-numeric(K)
  for (k in 1:K)  nk[k]<-sum(Sj[-i]==k)
  mupostnew<-mupost
  lambdapostnew<-lambdapost
  alfapostnew<-alfapost
  betapostnew<-betapost
  ykmeannew<-ykmean
  yksumsquarnew<-yksumsquar
  if (nk[Sj[i]]>0){
   param<-postparamvarunyis(Y[-c(which(Sj!=Sj[i]),i),],mu0,lambda0,alfa0,
beta0,nk[Sj[i]])
   mupostnew[,Sj[i]]<-param[[1]]
   lambdapostnew[,Sj[i]]<-param[[2]]
   alfapostnew[,Sj[i]]<-param[[3]]
   betapostnew[,Sj[i]]<-param[[4]]
   ykmeannew[,Sj[i]]<-param[[5]]
   yksumsquarnew[,Sj[i]]<-param[[6]]}
  prob<-numeric(K)
  for (k in 1:K){
   ni<-2*alfapostnew[,k]
   mu<-mupostnew[,k]
   desvio<-sqrt((2*betapostnew[,k]*(1+lambdapostnew[,k]))/(ni*
lambdapostnew[,k]))
   prob[k]<-nk[k]*prod(dstudentt(Y[i,],ni,mu,desvio))}
  prob<-c(prob,alpha*prod(dstudentt(Y[i,],ni0,mu0,desvio0)))
  Snew<-rDiscreta(prob/sum(prob))
 #
  if (Snew != Sj[i] & Snew <= K){
   Sj[i]<-Snew
   nk[Sj[i]]<-nk[Sj[i]]+1
   mupost<-mupostnew
   lambdapost<-lambdapostnew
   alfapost<-alfapostnew
   betapost<-betapostnew
   ykmean<-ykmeannew
   yksumsquar<-yksumsquarnew
   param<-postparamvarunyis(Y[which(Sj==Sj[i]),],mu0,lambda0,alfa0,
beta0,nk[Sj[i]])
   mupost[,Sj[i]]<-param[[1]]
   lambdapost[,Sj[i]]<-param[[2]]
   alfapost[,Sj[i]]<-param[[3]]
   betapost[,Sj[i]]<-param[[4]]
   ykmean[,Sj[i]]<-param[[5]]
   yksumsquar[,Sj[i]]<-param[[6]]}
  if (Snew != Sj[i] & Snew > K){
   Sj[i]<-Snew
   nk<-c(nk,1)
   mupost<-mupostnew
   lambdapost<-lambdapostnew
   alfapost<-alfapostnew
   betapost<-betapostnew
   ykmean<-ykmeannew
   yksumsquar<-yksumsquarnew
   param<-postparamvarunyis(Y[which(Sj==Sj[i]),],mu0,lambda0,alfa0,beta0,
nk[Sj[i]])
   mupost<-cbind(mupost,param[[1]])
   lambdapost<-cbind(lambdapost,param[[2]])
   alfapost<-cbind(alfapost,param[[3]])
   betapost<-cbind(betapost,param[[4]])
   ykmean<-cbind(ykmean,param[[5]])
   yksumsquar<-cbind(yksumsquar,param[[6]])}
  while (length(table(Sj))<max(Sj)){ # exclude empty clusters
   categr<-as.numeric(as.character(data.frame(table(Sj))[,1]))
   categd<-seq(1:length(table(Sj)))
   dif<-which(categr!=categd)
   Sj[which(Sj>dif[1])]<-Sj[which(Sj>dif[1])]-1
   mupost<-matrix(c(mupost[,-dif[1]]),nrow=Dim)
   lambdapost<-matrix(c(lambdapost[,-dif[1]]),nrow=Dim)
   alfapost<-matrix(c(alfapost[,-dif[1]]),nrow=Dim)
   betapost<-matrix(c(betapost[,-dif[1]]),nrow=Dim)
   ykmean<-matrix(c(ykmean[,-dif[1]]),nrow=Dim)
   yksumsquar<-matrix(c(yksumsquar[,-dif[1]]),nrow=Dim)
   K<-ncol(mupost)}
  if (length(table(Sj))<K){
   mupost<-matrix(c(mupost[,-K]),nrow=Dim)
   lambdapost<-matrix(c(lambdapost[,-K]),nrow=Dim)
   alfapost<-matrix(c(alfapost[,-K]),nrow=Dim)
   betapost<-matrix(c(betapost[,-K]),nrow=Dim)
   ykmean<-matrix(c(ykmean[,-K]),nrow=Dim)
   yksumsquar<-matrix(c(yksumsquar[,-K]),nrow=Dim)
   K<-ncol(mupost)}
  K<-ncol(mupost)}
  if (int>burnin & int%%saltos==0) cat(’’,Sj,file=paste(caminho,"Sj_batch",
b,".txt",sep=""),append=T)}}
#
############
# Define final local cluster for each batch
############
local_clust<-function(caminho,arqu,Nt,thresh){
 Sj<-scan(file=paste(caminho,arqu,sep=""))
 Sj<-matrix(Sj,ncol=Nt,byrow=TRUE)
 #
 Sj.j<-Sj
 prob.eq<-matrix(0,nrow=ncol(Sj.j),ncol=ncol(Sj.j))
 for (i in 1:ncol(Sj.j)){
  for (j in 1:ncol(Sj.j)){
   prob.eq[i,j]<-round(sum(Sj.j[,i]==Sj.j[,j])/length(Sj.j[,i]),4)*100}}
 clust<-c(1,rep(0,(ncol(Sj.j)-1)))
 for (i in 2:ncol(Sj.j)){
  if (max(prob.eq[i,1:(i-1)])>thresh) clust[i]<-clust[which(prob.eq[i,1:(i-1)]
==max(prob.eq[i,1:(i-1)]))[1]] else clust[i]<-max(clust[1:(i-1)]+1)}
 thesing<-0.3
 singl<-which(clust %in% which(table(clust)==1))
 if (length(singl)>1){
  prob.eq.sin<-prob.eq[singl,]
  for (i in 1:nrow(prob.eq.sin)){
   prob.eq.sin[i,singl[i]]<-0
   if (max(prob.eq.sin[i,])>thesing) clust[singl[i]]<-clust[which(prob.eq.sin[i,]
==max(prob.eq.sin[i,]))[1]]}
  while (length(table(clust))<max(clust)){ # exclude empty clusters
   categr<-as.numeric(as.character(data.frame(table(clust))[,1]))
   categd<-seq(1:length(table(clust)))
   dif<-which(categr!=categd)
   clust[which(clust>dif[1])]<-clust[which(clust>dif[1])]-1}}
 return(clust)}
#
############
# Gibbs to identify global clusters
############
Gibbs_global<-function(batches,name_Yb,name_clustb,Zj,burnin,amostrasfin,
saltos,alpha,mu0,lambda0,alfa0,beta0){
 AmostrasTotal<-burnin+amostrasfin*saltos
 Klocal<-numeric()
 for (b in 1:batches) Klocal<-c(Klocal,length(table(get(paste(name_clustb,
b,sep="")))))
 nk<-numeric(sum(Klocal))
 Dim<-ncol(get(paste(name_Yb,b,sep="")))
 ykmean<-matrix(0,nrow=Dim,ncol=sum(Klocal))
 yksumsquar<-matrix(0,nrow=Dim,ncol=sum(Klocal))
 clus<-1
 for (b in 1:batches){
  for (k in 1:Klocal[b]){
   nk[clus]<-length(which(get(paste(name_clustb,b,sep=""))==k))
   if (nk[clus]>1){
    ykmean[,clus]<-apply(get(paste(name_Yb,b,sep=""))[which(get(paste(
name_clustb,b,sep=""))==k),],2,mean)
```

```
    yksumsquar[,clus]<-apply(get(paste(name_Yb,b,sep=""))[which(get(
paste(name_clustb,b,sep=""))==k),]**2,2,sum)}
   if (nk[clus]==1){
    ykmean[,clus]<-get(paste(name_Yb,b,sep=""))[which(get(paste(
name_clustb,b,sep=""))==k),]
    yksumsquar[,clus]<-get(paste(name_Yb,b,sep=""))[which(get(paste(
name_clustb,b,sep=""))==k),]**2}
   clus<-clus+1}}
 #
 Ynk<-nk
 Ymean<-ykmean
 Ysum<-matrix(0,ncol=ncol(Ymean),nrow=nrow(Ymean))
 for (i in 1:length(Ynk)) Ysum[,i]<-Ymean[,i]*Ynk[i]
 Ysumsquar<-yksumsquar
 Nt<-ncol(Ymean)
 #
 K<-length(table(Zj))
 nk<-numeric(K)
 for (k in 1:K)  nk[k]<-sum(Ynk[which(Zj==k)])
 yksum<-matrix(0,ncol=K,nrow=nrow(Ymean))
 ykmean<-matrix(0,ncol=K,nrow=nrow(Ymean))
 yksumsquar<-matrix(0,ncol=K,nrow=nrow(Ymean))
 for (k in 1:K){
  if (length(which(Zj==k))>1){
   yksum[,k]<-apply(Ysum[,which(Zj==k)],1,sum)
   yksumsquar[,k]<-apply(Ysumsquar[,which(Zj==k)],1,sum)}
  if (length(which(Zj==k))==1){
   yksum[,k]<-Ysum[,which(Zj==k)]
   yksumsquar[,k]<-Ysumsquar[,which(Zj==k)] }
  ykmean[,k]<-yksum[,k]/nk[k]}
 mupost<-matrix(0,nrow=Dim,ncol=K)
 lambdapost<-matrix(0,nrow=Dim,ncol=K)
 alfapost<-matrix(0,nrow=Dim,ncol=K)
 betapost<-matrix(0,nrow=Dim,ncol=K)
 for (k in 1:K){
  mupost[,k]<-(ykmean[,k]*nk[k]+lambda0*mu0)/(lambda0+nk[k])
  lambdapost[,k]<-lambda0+nk[k]
  alfapost[,k]<-alfa0+(nk[k]/2)
  s2<-yksumsquar[,k]-(nk[k]*ykmean[,k]**2)
  betapost[,k]<-beta0+(s2+((nk[k]*lambda0*(ykmean[,k]-mu0)**2)/
(lambda0+nk[k])))/2}
 #
 for (int in 1:AmostrasTotal){
  cat('\n', int)
 for (i in 1:Nt){
  nk<-numeric(K)
  for (k in 1:K)  nk[k]<-sum(Ynk[-c(which(Zj!=k),i)])
  mupostnew<-mupost
  lambdapostnew<-lambdapost
  alfapostnew<-alfapost
  betapostnew<-betapost
  yksumnew<-yksum
  ykmeannew<-ykmean
  yksumsquarnew<-yksumsquar
  if (nk[Zj[i]]>0){
   yksumnew[,Zj[i]]<-yksum[,Zj[i]]-Ysum[,i]
   ykmeannew[,Zj[i]]<-yksumnew[,Zj[i]]/(nk[Zj[i]])
   yksumsquarnew[,Zj[i]]<-yksumsquar[,Zj[i]]-Ysumsquar[,i]
   mupostnew[,Zj[i]]<-(ykmeannew[,Zj[i]]*nk[Zj[i]]+lambda0*mu0)/(lambda0+nk[Zj[i]])
   lambdapostnew[,Zj[i]]<-lambda0+nk[Zj[i]]
   alfapostnew[,Zj[i]]<-alfa0+(nk[Zj[i]]/2)
   s2<-yksumsquarnew[,Zj[i]]-(nk[Zj[i]]*ykmeannew[,Zj[i]]**2)
   betapostnew[,Zj[i]]<-beta0+(s2+((nk[Zj[i]]*lambda0*(ykmeannew[,Zj[i]]-mu0)**2)/
(lambda0+nk[Zj[i]])))/2}
  prob<-numeric(K)
  varia<-Ysumsquar[,i]-(Ynk[i]*Ymean[,i]**2)
  for (k in 1:K) prob[k]<-sum(dmarglikeli(Ynk[i],Ymean[,i],varia,
lambdapostnew[,k],alfapostnew[,k],betapostnew[,k],mupostnew[,k]))
  prob<-c(prob,sum(dmarglikeli(Ynk[i],Ymean[,i],varia,lambda0,alfa0,beta0,mu0)))
  mk<-nk
  if (nk[Zj[i]]==0) mk[Zj[i]]<-1
  mk<-c(lgamma(mk+Ynk[i])-lgamma(mk),(log(alpha)+lgamma(Ynk[i])))
  prob<-mk+prob
  prob<-exp(prob-max(prob))
  if (nk[Zj[i]]==0) prob[Zj[i]]<-0
  Snew<-rDiscreta(prob/sum(prob))
  #
  if (Snew != Zj[i] & Snew <= K){
   Zj[i]<-Snew
   nk[Zj[i]]<-nk[Zj[i]]+Ynk[i]
   mupost<-mupostnew
   lambdapost<-lambdapostnew
   alfapost<-alfapostnew
   betapost<-betapostnew
   yksum<-yksumnew
   ykmean<-ykmeannew
   yksumsquar<-yksumsquarnew
   yksum[,Zj[i]]<-yksum[,Zj[i]]+Ysum[,i]
   ykmean[,Zj[i]]<-yksum[,Zj[i]]/nk[Zj[i]]
   yksumsquar[,Zj[i]]<-yksumsquar[,Zj[i]]+Ysumsquar[,i]
   mupost[,Zj[i]]<-(ykmean[,Zj[i]]*nk[Zj[i]]+lambda0*mu0)/(lambda0+nk[Zj[i]])
   lambdapost[,Zj[i]]<-lambda0+nk[Zj[i]]
   alfapost[,Zj[i]]<-alfa0+(nk[Zj[i]]/2)
   s2<-yksumsquar[,Zj[i]]-(nk[Zj[i]]*ykmean[,Zj[i]]**2)
   betapost[,Zj[i]]<-beta0+(s2+((nk[Zj[i]]*lambda0*(ykmean[,Zj[i]]-mu0)**2)/
(lambda0+nk[Zj[i]])))/2}
  if (Snew != Zj[i] & Snew > K){
   Zj[i]<-Snew
   nk<-c(nk,Ynk[i])
   yksum<-cbind(yksumnew,Ysum[,i])
   ykmean<-cbind(ykmeannew,Ymean[,i])
   yksumsquar<-cbind(yksumsquarnew,Ysumsquar[,i])
   mupost<-cbind(mupostnew,(ykmean[,Zj[i]]*nk[Zj[i]]+lambda0*mu0)/
(lambda0+nk[Zj[i]]))
   lambdapost<-cbind(lambdapostnew,lambda0+nk[Zj[i]])
   alfapost<-cbind(alfapostnew,alfa0+(nk[Zj[i]]/2))
   s2<-yksumsquar[,Zj[i]]-(nk[Zj[i]]*ykmean[,Zj[i]]**2)
   betapost<-cbind(betapostnew,beta0+(s2+((nk[Zj[i]]*lambda0*
(ykmean[,Zj[i]]-mu0)**2)/(lambda0+nk[Zj[i]])))/2)}
  while (length(table(Zj))<max(Zj)){ # exclude empty clusters
   categr<-as.numeric(as.character(data.frame(table(Zj))[,1]))
   categd<-seq(1:length(table(Zj)))
   dif<-which(categr!=categd)
   Zj[which(Zj>dif[1])]<-Zj[which(Zj>dif[1])]-1
   mupost<-matrix(c(mupost[,-dif[1]]),nrow=Dim)
   lambdapost<-matrix(c(lambdapost[,-dif[1]]),nrow=Dim)
   alfapost<-matrix(c(alfapost[,-dif[1]]),nrow=Dim)
   betapost<-matrix(c(betapost[,-dif[1]]),nrow=Dim)
   yksum<-matrix(c(yksum[,-dif[1]]),nrow=Dim)
   ykmean<-matrix(c(ykmean[,-dif[1]]),nrow=Dim)
   yksumsquar<-matrix(c(yksumsquar[,-dif[1]]),nrow=Dim)
   K<-ncol(mupost)}
  if (length(table(Zj))<K){
   mupost<-matrix(c(mupost[,-K]),nrow=Dim)
   lambdapost<-matrix(c(lambdapost[,-K]),nrow=Dim)
   alfapost<-matrix(c(alfapost[,-K]),nrow=Dim)
   betapost<-matrix(c(betapost[,-K]),nrow=Dim)
   yksum<-matrix(c(yksum[,-K]),nrow=Dim)
   ykmean<-matrix(c(ykmean[,-K]),nrow=Dim)
   yksumsquar<-matrix(c(yksumsquar[,-K]),nrow=Dim)
   K<-ncol(mupost)}
  K<-ncol(mupost)}
 if (int>burnin & int%%saltos==0) cat(’’,Zj,file=paste(caminho,
"Global_cluster.txt",sep=""),append=T)}}
#
#########################
# data set
#########################
#
set.seed(100)
muvec<-matrix(c(0,0.5,1.1,-11.6,-8,1.7,3.1,1.9,-0.5,1.2,1.7,-11.7,-6.3,0.1,
2.3,2.4,0.5,-1.2,2.7,-11.6,-7,0.7,2.6,2.2,1,1.2,3.7,-11.6,-7.7,1.2,2.9,2.1),
nrow=4,byrow=TRUE)
sigmavec<-matrix(c(0.002140429, 0.001531872, 0.225287998, 0.053739488,
0.126138910, 0.112735647, 0.133862779, 0.002140429, 0.001798247, 0.006215775,
0.315767179, 0.231431127, 0.472012711, 0.129508527, 1.008442513, 0.001798247,
0.002172256, 0.015160350, 2.142055552, 0.885205346, 0.549862208, 3.155843303,
0.887398965, 0.002172256,0.004303235, 0.011570390, 1.311111853, 1.026574670,
1.325584473, 0.319953786, 1.874197742, 0.004303235),nrow=4,byrow=TRUE)
Nt<-5000
compon<-numeric(Nt)
Y<-matrix(0,ncol=ncol(muvec),nrow=Nt)
```

```
for (i in 1:Nt){
 compon[i]<-rDiscreta(rep(1/nrow(muvec),nrow(muvec)))
 Y[i,]<-rnorm(ncol(muvec),muvec[compon[i],],sqrt(sigmavec[compon[i],]))}
dados<-transfdata(Y)
#
batches<-5 # number of shards
amostra<-1:nrow(dados)
for (b in 1:batches){
 assign(paste("b",b,sep=""),sort(sample(amostra,nrow(dados)/batches)))
 assign(paste("Yb",b,sep=""),dados[get(paste("b",b,sep="")),])
 assign(paste("componb",b,sep=""),compon[get(paste("b",b,sep=""))])
 amostra<-amostra[-which(amostra %in% get(paste("b",b,sep="")))]}
#
caminho<-"/XXXX/XXX/" # this is the location of the results’ files.
#
#########################
# Local clusters
#########################
#
library(compiler)
enableJIT(3)
#
for (b in 1:batches){ # can be parallelized
 Y<-get(paste("Yb",b,sep=""))
 Nt<-nrow(Y)
 Dim<-ncol(Y)
 #
 # Initialize Sj and hyperparameters
 #
 Sj<-rep(1,nrow(Y)) # local cluster’s membership indicator starting point
 amostrasfin<-5000 # sample size after burnin and jumps
 burnin<-5000 # burnin size
 saltos<-5 # jumps size
 set.seed(100)
 alpha<-1 # total mass parameter of DP
 mu0<-matrix(0,ncol=1,nrow=Dim) # G0
 lambda0<-matrix(0.1,nrow=Dim,ncol=1) # G0
 alfa0<-matrix(5,nrow=Dim,ncol=1) # G0
 beta0<-matrix(3,nrow=Dim,ncol=1) # G0
 #
 # run Gibbs to estimate local cluster
 #
 Gibbs_local(Y,Sj,burnin,amostrasfin,saltos,alpha,mu0,lambda0,alfa0,beta0)}
#
# Determine final local clusters
#
batches<-5
Nt<-c(rep(1000,batches)) # number of observations in each batch
thresh<-0.5*100
#
library(compiler)
enableJIT(3)
for (b in 1:batches) assign(paste("clust",b,sep=""),local_clust(caminho,
paste("Sj_batch",b,".txt",sep=""),Nt[b],thresh))
#
#########################
# Global clusters
#########################
#
library(compiler)
enableJIT(3)
#
# Initialize Zj and posterior parameters
#
set.seed(100)
alpha<-1
mu0<-matrix(0,ncol=1,nrow=Dim) # G0
lambda0<-matrix(0.1,nrow=Dim,ncol=1) # G0
alfa0<-matrix(5,nrow=Dim,ncol=1) # G0
beta0<-matrix(3,nrow=Dim,ncol=1) # G0
#
Klocal<-numeric()
for (b in 1:batches) Klocal<-c(Klocal,length(table(get(paste(
"clust",b,sep="")))))
Zj<-seq(1:sum(Klocal)) # global cluster’s membership indicator starting point
amostrasfin<-5000 # sample size after burnin and jumps
```

```
burnin<-5000 # burnin size
saltos<-5 # jumps size
#
# Run Gibbs sampler to find global clustes
#
Gibbs_global(batches,"Yb","clust",Zj,burnin,amostrasfin,saltos,alpha,mu0,
lambda0,alfa0,beta0)
#
# Determine final global clusters
#
Nt<-sum(Klocal)
thresh<-0.5*100
clust_f<-local_clust(caminho,paste("Global_cluster.txt",sep=""),Nt,thresh)
#
# Allocate all observation in the final global clusters
#
clust_local<-get(paste("clust",1,sep=""))
for (b in 2:batches) clust_local<-c(clust_local,get(paste(
"clust",b,sep=""))+max(clust_local))
for (i in 1:length(clust_local)) clust_local[i]<-clust_f[clust_local[i]]
```

Chapter

8

Conclusions

We describe a generalization of the dependence structure of a mixture model and propose data-driven Bayesian procedure to estimate models with unknown dependence relationship and number of components. Some important and useful models in Genetics and Molecular Biology: HMM and QTL mapping are special cases.

The proposed methodology for QTL mapping in pedigree data also estimates missing parents genotype and considers the correlation between close *loci* on the same chromosome to estimate nonfounder genotype, simplify likelihood definition and improve the Mendelian probability of inheritance.

These methods present a good performance under tested situations and applications to simulated and real data illustrate situations where the proposed models are usually more suitable than the conventional models. The proposed methods improve the mixing of the MCMC, implying in faster convergence. This permits to work with shorter sequences in the simulation process which is a consequence of better proposals in transdimensional moves. We also discuss some Bayesian diagnostic measures and suggest applying them in QTL mapping model checking.

For clustering genes, we propose three nonparametric Bayesian model-based algorithm: the marginal NDP scheme, the PRC algorithm and the SNOB method. The first method is able to cluster distributions and the two latter algorithms cluster big data. The PRC is an approximate method which requires a single cycle of simple deterministic calculations for each observation under study. The SNOB is an exact method that finds global clusters in a parallel and distributed algorithm.

All proposed clustering methods show good performance to classify simulated and real data sets. Compared with traditional clustering algorithms including k-means, DBSCAN, SUGS and the EM algorithm, proposed algorithms have favorable misclassification rates and the clusters that are reported under proposed methods have clear distinct features.

Bibliography

Abney, M. (2008). Identity-by-descent estimation and mapping of qualitative traits in large, complex pedigrees. *Genetics*, **179**(3), 1577–1590.

Almasy, L. & Blangero, J. (1998). Multipoint quantitative-trait linkage analysis in general pedigrees. *The American Journal of Human Genetics*, **62**(5), 1198–1211.

Arbel, J., Lijoi, A. & Nipoti, B. (2015). Bayesian survival model based on moment characterization. In *Bayesian Statistics from Methods to Models and Applications*, pages 3–14. Springer.

Balding, D. J., Bishop, M. & Cannings, C. (2008). *Handbook of statistical genetics*, volume 1. John Wiley & Sons.

Banerjee, A., Merugu, S., Dhillon, I. S. & Ghosh, J. (2005). Clustering with Bregman divergences. *The Journal of Machine Learning Research*, **6**, 1705–1749.

Basten, C. J., Weir, B. S. & Zeng, Z.-B. (1997). QTL cartographer: a reference manual and tutorial for QTL mapping. *Department of Statistics, North Carolina State University, Raleigh, NC*.

Bauman, L. E., Almasy, L., Blangero, J., Duggirala, R., Sinsheimer, J. S. & Lange, K. (2005). Fishing for pleiotropic QTLs in a polygenic sea. *Annals of Human Genetics*, **69**(5), 590–611.

Beckett, L. & Diaconis, P. (1994). Spectral analysis for discrete longitudinal data. *Advances in Mathematics*, **103**(1), 107–128.

Bink, M., Jansen, J., Madduri, M., Voorrips, R., Durel, C.-E., Kouassi, A., Laurens, F., Mathis, F., Gessler, C., Gobbin, D. *et al.* (2014). Bayesian QTL analyses using pedigreed families of an outcrossing species, with application to fruit firmness in apple. *Theoretical and Applied Genetics*, **127**(5), 1073–1090.

Blackwell, D. & MacQueen, J. B. (1973). Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, **1**, 353–355.

Boys, R. J. & Henderson, D. (2002). On determining the order of Markov dependence of an observed process governed by a hidden Markov model. *Scientific Programming*, **10**(3), 241–251.

Boys, R. J. & Henderson, D. A. (2004). A Bayesian approach to DNA sequence segmentation. *Biometrics*, **60**(3), 573–581.

Boys, R. J., Henderson, D. A. & Wilkinson, D. J. (2000). Detecting homogeneous segments in DNA sequences by using hidden Markov models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **49**(2), 269–285.

Broman, K. W. (2006). Use of hidden markov models for QTL mapping. In *Johns Hopkins University, Dept. of Biostatistics Working Papers*. bepress.

Broman, K. W. & Speed, T. (1999). A review of methods for identifying QTLs in experimental crosses. *Lecture Notes-Monograph Series*, **33**, 114–142.

Brooks, S. P., Giudici, P. & Roberts, G. O. (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(1), 3–39.

Brown, M. D., Glazner, C. G., Zheng, C. & Thompson, E. A. (2012). Inferring coancestry in population samples in the presence of linkage disequilibrium. *Genetics*, **190**(4), 1447–1460.

Cannings, C., Thompson, E. & Skolnick, M. (1978). Probability functions on complex pedigrees. *Advances in Applied Probability*, pages 26–61.

Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. *Bulletin of mathematical biology*, **51**(1), 79–94.

Churchill, G. A. (1992). Hidden Markov chains and the analysis of genome structure. *Computers & chemistry*, **16**(2), 107–115.

Comuzzie, A. G., Hixson, J. E., Almasy, L., Mitchell, B. D., Mahaney, M. C., Dyer, T. D., Stern, M. P., MacCluer, J. W. & Blangero, J. (1997). A major quantitative trait locus determining serum leptin levels and fat mass is located on human chromosome 2. *Nature Genetics*, **15**(3), 273–276.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. & Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, **47**(4), 547–553.

Cox, A., Ackert-Bicknell, C. L., Dumont, B. L., Ding, Y., Bell, J. T., Brockmann, G. A., Wergedal, J. E., Bult, C., Paigen, B., Flint, J. *et al.* (2009). A new standard genetic map for the laboratory mouse. *Genetics*, **182**(4), 1335–1344.

Dahl, D. B. (2006). Model-based clustering for expression data via a Dirichlet process mixture model. In M. Vannucci, K.-A. Do, & P. Müller, editors, *Bayesian Inference for Gene Expression and Proteomics*. Cambridge University Press.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **39**(1), 1–38.

Druet, T. & Farnir, F. P. (2011). Modeling of identity-by-descent processes along a chromosome between haplotypes and their genotyped ancestors. *Genetics*, **188**(2), 409–419.

Druet, T. & Georges, M. (2010). A hidden Markov model combining linkage and linkage disequilibrium information for haplotype reconstruction and quantitative trait locus fine mapping. *Genetics*, **184**(3), 789–798.

Elston, R. C. & Stewart, J. (1971). A general model for the genetic analysis of pedigree data. *Human Heredity*, **21**(6), 523–542.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery in Databases*, volume 96, pages 226–231.

Fan, Y., Peters, G. W. & Sisson, S. A. (2009). Automating and evaluating reversible jump MCMC proposal distributions. *Statistics and Computing*, **19**(4), 409–421.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, **1**, 209–230.

Fishelson, M. & Geiger, D. (2002). Exact genetic linkage computations for general pedigrees. *Bioinformatics*, **18**(Suppl 1), S189–S198.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**(2), 179–188.

Fraley, C. & Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, **97**(458), 611–631.

Fraley, C. & Raftery, A. E. (2007). Bayesian regularization for normal mixture estimation and model-based clustering. *Journal of Classification*, **24**(2), 155–181.

Fraley, C., Raftery, A. E. & Scrucca, L. (2012). Normal mixture modeling for model-based clustering, classification, and density estimation. *Department of Statistics, University of Washington, Available online at http://cran.r-project.?org/web/packages/mclust/index.html*, **23**, 2012.

Geisser, S. (1980). Discussion on sampling and Bayes' inference in scientific modeling and robustness (by GEP Box). *Journal of the Royal Statistical Society: Series A (General)*, **143**, 416–417.

Gelfand, A. E. & Dey, D. K. (1994). Bayesian model choice: Asymptotics and exact calculations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **56**, 501–514.

Gelman, A., Meng, X.-L. & Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760.

Gelman, A., Hwang, J. & Vehtari, A. (2014). Understanding predictive information criteria for bayesian models. *Statistics and Computing*, **24**(6), 997–1016.

Ghoshal, S. (2010). *The Dirichlet process, related priors and posterior asymptotics*, volume 2. Chapter.

Gough, J., Karplus, K., Hughey, R. & Chothia, C. (2001). Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *Journal of Molecular Biology*, **313**(4), 903–919.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**(4), 711–732.

Green, P. J. & Mira, A. (2001). Delayed rejection in reversible jump Metropolis–Hastings. *Biometrika*, **88**(4), 1035–1053.

Green, P. J. & Richardson, S. (2001). Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, **28**(2), 355–375.

Guha, S., Meyerson, A., Mishra, N., Motwani, R. & O'Callaghan, L. (2003). Clustering data streams: Theory and practice. *Knowledge and Data Engineering, IEEE Transactions on*, **15**(3), 515–528.

Habier, D., Totir, L. R. & Fernando, R. L. (2010). A two-stage approximation for analysis of mixture genetic models in large pedigrees. *Genetics*, **185**(2), 655–670.

Haldane, J. (1919). The combination of linkage value and the calculation of distances between the loci of linkage factors. *Genetics*, **8**, 299–309.

Haley, C. S. & Knott, S. A. (1992). A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity*, **69**(4), 315–324.

Heath, S. C. (1997). Markov chain Monte Carlo segregation and linkage analysis for oligogenic models. *The American Journal of Human Genetics*, **61**(3), 748–760.

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

Huang, Z. & Gelman, A. (2005). Sampling for Bayesian computation with large datasets. *Available at SSRN 1010107*.

Jackson, C. H., Sharples, L. D., Thompson, S. G., Duffy, S. W. & Couto, E. (2003). Multistate Markov models for disease progression with classification error. *Journal of the Royal Statistical Society: Series D (The Statistician)*, **52**(2), 193–209.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, **31**(8), 651–666.

Jain, S. & Neal, R. M. (2004). A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, **13**(1), 158–182.

Jain, S. & Neal, R. M. (2007). Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, **2**(3), 445–472.

Jansen, R. C. (1993). Interval mapping of multiple quantitative trait loci. *Genetics*, **135**(1), 205–211.

Jansen, R. C. & Stam, P. (1994). High resolution of quantitative traits into multiple loci via interval mapping. *Genetics*, **136**(4), 1447–1455.

Kang, H. M., Zaitlen, N. A. & Eskin, E. (2010). EMINIM: an adaptive and memory-efficient algorithm for genotype imputation. *Journal of Computational Biology*, **17**(3), 547–560.

Kao, C.-H., Zeng, Z.-B. & Teasdale, R. D. (1999). Multiple interval mapping for quantitative trait loci. *Genetics*, **152**(3), 1203–1216.

Kashima, H., Hu, J., Ray, B. & Singh, M. (2008). K-means clustering of proportional data using L1 distance. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

Kass, R. E., Carlin, B. P., Gelman, A. & Neal, R. M. (1998). Markov chain Monte Carlo in practice: a roundtable discussion. *The American Statistician*, **52**(2), 93–100.

Kranz, J. (2003). *Comparative epidemiology of plant diseases*. Springer.

Kulis, B. & Jordan, M. I. (2012). Revisiting k-means: New algorithms via Bayesian nonparametrics. *Proceedings of 29th International Conference on Machine Learning (ICML), Edinburgh*.

Lander, E. S. & Botstein, D. (1989). Mapping mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics*, **121**(1), 185–199.

Lander, E. S. & Green, P. (1987). Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences*, **84**(8), 2363–2367.

Lange, K. & Sobel, E. (2006). Variance component models for X-linked QTLs. *Genetic Epidemiology*, **30**(5), 380–383.

Lange, K., Westlake, J. & Spence, M. (1976). Extensions to pedigree analysis III. Variance components by the scoring method. *Annals of Human Genetics*, **39**(4), 485–491.

Lange, K., Boehnke, M. & Opitz, J. M. (1983). Extensions to pedigree analysis. IV. Covariance components models for multivariate traits. *American Journal of Medical Genetics*, **14**(3), 513–524.

Lange, K., Papp, J. C., Sinsheimer, J. S., Sripracha, R., Zhou, H. & Sobel, E. M. (2013). Mendel: the Swiss army knife of genetic analysis programs. *Bioinformatics*, **29**(12), 1568–1570.

Lauritzen, S. L. & Sheehan, N. A. (2003). Graphical models for genetic analyses. *Statistical Science*, **18**(4), 489–514.

Leal, S. M., Yan, K. & Müller-Myhsok, B. (2005). SimPed: a simulation program to generate haplotype and genotype data for pedigree structures. *Human Heredity*, **60**(2), 119–122.

Lee, J., Müller, P., Zhu, Y. & Ji, Y. (2013). A nonparametric Bayesian model for local clustering with application to proteomics. *Journal of the American Statistical Association*, **108**(503), 775–788.

Lee, S. H., Goddard, M. E., Visscher, P. M. & van der Werf, J. H. (2010). Using the realized relationship matrix to disentangle confounding factors for the estimation of genetic variance components of complex traits. *Genetics Selection Evolution*, **42**(1), 1.

Lee, S. Y., Lee, J. Y., Jung, K. S. & Ryu, K. H. (2009). A 9-state hidden Markov model using protein secondary structure information for protein fold recognition. *Computers in Biology and Medicine*, **39**(6), 527–534.

Lesaffre, E. & Lawson, A. B. (2012). *Bayesian biostatistics*. John Wiley & Sons.

Li, Y., Willer, C., Sanna, S. & Abecasis, G. (2009). Genotype imputation. *Annual Review of Genomics and Human Genetics*, **10**, 387.

Linde, Y., Buzo, A. & Gray, R. M. (1980). An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, **28**(1), 84–95.

Liu, J. S. (1996). Nonparametric hierarchical Bayes via sequential imputations. *The Annals of Statistics*, **24**(3), 911–930.

Lund, M. S., Sahana, G., de Koning, D.-J., Su, G. & Carlborg, Ö. (2009). Comparison of analyses of the QTLMAS XII common dataset. I: Genomic selection. *BMC Proceedings*, **3**(Suppl1), S1.

MacEachern, S. N., Clyde, M. & Liu, J. S. (1999). Sequential importance sampling for nonparametric Bayes models: The next generation. *Canadian Journal of Statistics*, **27**(2), 251–267.

Madden, L. & Hughes, G. (1995). Plant disease incidence: distributions, heterogeneity, and temporal analysis. *Annual Review of Phytopathology*, **33**(1), 529–564.

Mao, J. & Jain, A. K. (1996). A self-organizing network for hyperellipsoidal clustering (hec). *Neural Networks, IEEE Transactions on*, **7**(1), 16–29.

McLachlan, G. & Peel, D. (2004). *Finite mixture models*. John Wiley & Sons.

Meira, S. A. (2014). Modelo de mistura com dependência Markoviana de primeira ordem. *PhD Thesis. Departamento de Estatística - Universidade Federal de São Carlos*.

Mitra, R., Müller, P., Liang, S., Yue, L. & Ji, Y. (2013). A Bayesian graphical model for chip-seq data on histone modifications. *Journal of the American Statistical Association*, **108**(501), 69–80.

Muri, F. (1998). Modelling bacterial genomes using hidden Markov models. In R. Payne & P. Green, editors, *Compstat'98 Proceedings in Computational Statistics*, pages 89–100. Physica-Verlag HD.

Newton, M. A., Quintana, F. A. & Zhang, Y. (1998). Nonparametric Bayes methods using predictive updating. In *Practical nonparametric and semiparametric Bayesian statistics*, pages 45–61. Springer.

Pandolfi, S., Bartolucci, F. & Friel, N. (2014). A generalized multiple-try version of the reversible jump algorithm. *Computational Statistics & Data Analysis*, **72**, 298–314.

Pennell, M. L. & Dunson, D. B. (2007). Fitting semiparametric random effects models to large data sets. *Biostatistics*, **8**(4), 821–834.

Pettit, L. (1990). The conditional predictive ordinate for the normal distribution. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **52**, 175–184.

Provost, S. (2005). Moment-based density approximants. *Mathematica Journal*, **9**, 727–756.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

Rabiner, L. R. & Juang, B.-H. (1986). An introduction to hidden Markov models. *ASSP Magazine, IEEE*, **3**(1), 4–16.

Richardson, S. & Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **59**(4), 731–792.

Robert, C. P., Ryden, T. & Titterington, D. M. (2000). Bayesian inference in hidden Markov models through the reversible jump Markov chain Monte Carlo method. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **62**(1), 57–75.

Rodríguez, A. & Ghosh, K. (2012). Modeling relational data using nested infinite relational models. Technical report, Department of Applied Mathematics and Statistics, University of California, Santa Cruz.

Rodriguez, A., Dunson, D. B. & Gelfand, A. E. (2008). The nested Dirichlet process. *Journal of the American Statistical Association*, **103**(483), 1131–1143.

Rothman, K. J., Greenland, S. & Lash, T. L. (2008). *Modern epidemiology*. Lippincott Williams & Wilkins.

Rubin, D. B. *et al.* (1984). Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, **12**(4), 1151–1172.

Saraiva, E. F. & Milan, L. A. (2012). Clustering gene expression data using a posterior split-merge-birth procedure. *Scandinavian Journal of Statistics*, **39**(3), 399–415.

Satagopan, J. M. & Yandell, B. S. (1996). Estimating the number of quantitative trait loci via Bayesian model determination. In *Proceedings of the Joint Statistical Meetings*. Citeseer.

Satagopan, J. M., Yandell, B. S., Newton, M. A. & Osborn, T. C. (1996). A Bayesian approach to detect quantitative trait loci using Markov chain Monte Carlo. *Genetics*, **144**(2), 805–816.

Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I. & McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, **11**(2), 78–88.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, **4**, 639–650.

Shi, J., Murray-Smith, R. & Titterington, D. (2002). Birth-death MCMC methods for mixtures with an unknown number of components. Technical report, Citeseer.

Söding, J. (2005). Protein homology detection by HMM–HMM comparison. *Bioinformatics*, **21**(7), 951–960.

Spezia, L. (2010). Bayesian analysis of multivariate Gaussian hidden Markov models with an unknown number of regimes. *Journal of Time Series Analysis*, **31**(1), 1–11.

Stephens, D. & Fisch, R. (1998). Bayesian analysis of quantitative trait locus data using reversible jump Markov chain Monte Carlo. *Biometrics*, **54**(4), 1334–1347.

Stephens, D. & Smith, A. (1993). Bayesian inference in multipoint gene mapping. *Annals of Human Genetics*, **57**(1), 65–82.

Suzanne, J. R. (2008). QTL mapping technology using variance components in general pedigrees applied to the poultry industry. *PhD Thesis. The University of Edinburgh.*

Tierney, L. & Mira, A. (1999). Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*, **18**(1718), 2507–2515.

Trippa, L. (2011). Personal communication.

Visser, I., Speekenbrink, M. *et al.* (2010). **depmixS4**: An R-package for hidden markov models. *Journal of Statistical Software*, **36**(7), 1–21.

Wade, S., Mongelluzzo, S. & Petrone, S. (2011). An enriched conjugate prior for Bayesian nonparametric inference. *Bayesian Analysis*, **6**(3), 359–385.

Wang, L. & Dunson, D. B. (2011). Fast Bayesian inference in Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, **20**(1), 196–216.

Wergedal, J. E., Ackert-Bicknell, C. L., Tsaih, S.-W., Sheng, M. H.-C., Li, R., Mohan, S., Beamer, W. G., Churchill, G. A. & Baylink, D. J. (2006). Femur mechanical properties in the F2 progeny of an NZB/B1NJ× RF/J cross are regulated predominantly by genetic loci that regulate bone geometry. *Journal of Bone and Mineral Research*, **21**(8), 1256–1266.

Xu, R., Wunsch, D. *et al.* (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, **16**(3), 645–678.

Yang, J., Benyamin, B., McEvoy, B. P., Gordon, S., Henders, A. K., Nyholt, D. R., Madden, P. A., Heath, A. C., Martin, N. G., Montgomery, G. W. *et al.* (2010). Common SNPs explain a large proportion of the heritability for human height. *Nature genetics*, **42**(7), 565–569.

Yi, N. & Xu, S. (2002). Mapping quantitative trait loci with epistatic effects. *Genetical Research*, **79**(2), 185–198.

Yi, N., Yandell, B. S., Churchill, G. A., Allison, D. B., Eisen, E. J. & Pomp, D. (2005). Bayesian model selection for genome-wide epistatic quantitative trait loci analysis. *Genetics*, **170**(3), 1333–1344.

Zeng, Z.-B. (1994). Precision mapping of quantitative trait loci. *Genetics*, **136**(4), 1457–1468.

Zhao, W., Ma, H. & He, Q. (2009). Parallel k-means clustering based on MapReduce. In *Cloud Computing*, pages 674–679. Springer.

Zhu, Y., Xu, Y., Helseth, D. L., Gulukota, K., Yang, S., Pesce, L. L., Mitra, R., Müller, P., Sengupta, S., Guo, W. *et al.* (2015). Zodiac: A comprehensive depiction of genetic interactions in cancer by integrating TCGA data. *Journal of the National Cancer Institute*, **107**(8), djv129.

Zuanetti, D. A. (2006). Modelos HMM com dependência de segunda ordem: aplicação em genética. *Master thesis. Departamento de Estatística - Universidade Federal de São Carlos.*

Zuanetti, D. A. & Milan, L. A. (2015). Estimating dependent binomial mixture models through reversible jump MCMC. In *Proceedings of the 60th ISI World Statistics Congress*, pages 2529–2534.

Zuanetti, D. A. & Milan, L. A. (2016). Data-driven reversible jump for QTL mapping. *Genetics*, **202**(1), 25–36.

Zuanetti, D. A. & Milan, L. A. (Submitted). A model for QTL mapping of pedigree data.

Zuanetti, D. A. & Milan, L. A. (To appear a). Second-order autoregressive hidden Markov model. *Brazilian Journal of Probability and Statistics*.

Zuanetti, D. A. & Milan, L. A. (To appear b). A generalized mixture model applied to diabetes incidence data. *Biometrical Journal*.

Zuanetti, D. A., Müller, P., Zhu, Y., Yang, S. & Ji, Y. (Submitted a). A marginal NDP – clustering distributions.

Zuanetti, D. A., Müller, P., Zhu, Y., Yang, S. & Ji, Y. (Submitted b). Big data clustering.

Zucchini, W. & MacDonald, I. L. (2009). *Hidden Markov models for time series: an introduction using R*. CRC press.