ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

# SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

## DEPARTMENT OF STATISTICS

## POSTGRADUATE PROGRAM

# “Computer Intensive Methods for Statistical Models with Latent Structures”

By

Stavroula Gerontogianni

A THESIS
Submitted to the Department of Statistics
of the Athens University of Economics and Business
in partial fulfilment of the requirements for
the degree of Master of Science in Statistics

Athens, Greece
September 2017

# ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

## ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

## ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ

# “Computer Intensive Methods for Statistical Models with Latent Structures”

Σταυρούλα Δ. Γεροντογιάννη

ΔΙΑΤΡΙΒΗ
Που υποβλήθηκε στο Τμήμα Στατιστικής
του Οικονομικού Πανεπιστημίου Αθηνών
ως μέρος των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος Ειδίκευσης στη Στατιστική

Αθήνα
Σεπτέμβριος 2017

# DEDICATION

I dedicate it,

To my parents Dionysios and Spyridoula and to my brother Dimitrios-Michael for their patience and understanding throughout this effort.

To my supervisor Dr. Nikolaos Demiris for believing in me and giving me the chance to start a wonderful journey at next, as well as to Professors Dimitris Karlis and Ioannis Ntzoufras for their valuable help and support.

# ACKNOWLEDGEMENTS

I would like to thank first of all my supervisor Dr. Nikolaos Demiris who supported me, motivated me and provided me with priceless help throughout my academic years as an undergraduate and graduate student; without him I wouldn’t be able to make true my future dreams.

At next, I would like to thank the academic staff of the Department of Statistics of the Athens University of Economics and Business. In particular, the members of the three-member committee of Master Thesis evaluation, for their constructive comments in my presentation of the thesis, especially Ioannis Ntzoufras, who contributed to its improvement.

I would also like to thank the Staff of the Secretariat of the Master of Sciences Program of the Athens University of Economics and Business for being so helpful and supportive throughout the academic year.

# VITA

I completed my four-year undergraduate studies (BSc) in Statistics at the Athens University of Economics and Business (AUEB) / School of Information Sciences and Technology / Department of Statistics. My mother language is Hellenic (Greek) and I speak English in a high level [Certificate of Proficiency in English (ECPE)] and French in a basic level (DIPLOME D'ETUDES EN LANGUE FRANCAISE, DELF B1). I am a member of: (a) Hellenic (Greek) Mathematical Society (HMS), (b) Hellenic (Greek) Chamber of Economics (c) Hellenic Ministry of Education Affairs / Department of Colleges as a Teacher of Statistics in Colleges (after high school education), (d) National (Hellenic) Organization for the Certification of Qualifications and Vocational Guidance (NOCQVG - EOPPEP) as a Teacher of Statistics (in high school education), (e) National (Hellenic) Organization for the Certification of Qualifications and Vocational Guidance (NOCQVG - EOPPEP) as a Teacher of English Language (in high school education). My activities & personal interests are: WTF Taekwondo [National champion (teen)], Chess [National champion (teen)], Gymnastics (resistances - aerobics), Literature, Philosophy and History.

# ABSTRACT

**Stavroula Gerontogianni**

## “Computer Intensive Methods for Statistical Models with Latent Structures”

September 2017

Several methods have been proposed and applied to different data problems in order to make Bayesian inference for the unknown density of the parameters in interest. The most widely used so far are the Markov Chain Monte Carlo methods, including Gibbs Sampling which will be one of the algorithms in focus on this project. Moreover, contemporary methods such as Variational Inference and Hamiltonian Monte Carlo promise respectable results as regards accuracy and time speed. On account of this, they are considered to be quite useful alternatives in cases where their advantages tend to play a greater role than their disadvantages. In this project the three methods mentioned above, giving greater emphasis on the theory behind Variational Inference, are implemented in mixture models of Gaussians aiming at their comparison, in terms of accuracy, statistical efficiency and computational cost. At this point, it is of crucial importance to highlight the different softwares used to implement the algorithms; Gibbs sampling run through OpenBugs and Variational Inference as well as Hamiltonian Monte Carlo through the new probabilistic language Stan. Consequently, any differences occurring in the results may also be derivatives of the different softwares usage. At this stage, it is important to mention that Stan is on experimental level, especially for the Variational Inference algorithm; hence some inaccuracies in the results may occur. Nevertheless, it is interesting to test its capabilities and the way it works, since it could be a quite useful tool soon. The interface for both softwares is chosen to be R; hence the R code is provided in the Appendix. It must be also noted that Stan provides a black box procedure for the aforementioned algorithms, which is discussed in detail for each method.

# ΠΕΡΙΛΗΨΗ

**Σταυρούλα Δ. Γεροντογιάννη**

**“Computer Intensive Methods for Statistical Models with Latent Structures”**

Σεπτέμβριος 2017

Πολλές μέθοδοι έχουν προταθεί και εφαρμοστεί σε πολλές διαφορετικές περιπτώσεις δεδομένων, με σκοπό την Μπεϋζιανή συμπερασματολογία για την άγνωστη κατανομή των παραμέτρων υπό εξέταση. Οι πιο κοινώς διαδομένες μέθοδοι είναι οι Markov Chain Monte Carlo, συμπεριλαμβανομένου του Gibbs Sampling, που αποτελεί έναν από τους αλγορίθμους που θα εστιάσουμε σε αυτήν την εργασία. Επιπλέον, σύγχρονες μέθοδοι, όπως το Variational Inference και το Hamiltonian Monte Carlo υπόσχονται αξιόλογα αποτελέσματα, όσον αφορά την ακρίβεια των εκτιμήσεων και την ταχύτητα εκπλήρωσης του αλγορίθμου. Γι’ αυτό το λόγο, θεωρούνται ενδεχομένως ως χρήσιμες εναλλακτικές σε περιπτώσεις όπου τα πλεονεκτήματα τους παίζουν σημαντικότερο ρόλο από τα μειονεκτήματα τους. Σε αυτή την εργασία, οι τρεις μέθοδοι που αναφέρονται παραπάνω, δίνοντας μεγαλύτερη έμφαση στη θεωρία του Variational Inference, εφαρμόζονται σε μοντέλα μίξης κανονικών κατανομών με σκοπό την σύγκρισή τους σε όρους ακρίβειας, στατιστικής σημαντικότητας και υπολογιστικού κόστους. Σε αυτό το σημείο, είναι πολύ σημαντικό να τονισθούν τα διαφορετικά softwares που χρησιμοποιήθηκαν για την εφαρμογή των αλγορίθμων. Ο αλγόριθμος Gibbs sampling έτρεξε μέσω OpenBugs, ενώ οι Variational Inference και Hamiltonian Monte Carlo μέσω της probabilistic γλώσσας Stan. Συνεπώς, τυχόυσες διαφορές στα αποτελέσματα της εφαρμογής των τριών μεθόδων ενδεχομένως να οφείλονται σε κάποιο βαθμό στα διαφορετικά softwares που χρησιμοποιήθηκαν. Σε αυτό το σημείο, είναι σημαντικό να αναφέρουμε ότι η γλώσσα Stan είναι υπό επεξεργασία, κυρίως για τον αλγόριθμο Variational Inference, επομένως ανακρίβειες στα αποτελέσματα μπορεί να υπάρξουν. Παρόλα αυτά, παρουσιάζει ενδιαφέρον να εξετασθούν οι ικανότητες αυτής της γλώσσας, καθώς θα αποτελέσει σύντομα ένα πολύ χρήσιμο εργαλείο. Επιπλέον το interface και για τα δύο softwares επιλέχθηκε να είναι η R συνεπώς, παρέχεται ο κώδικας R στο Appendix. Πρέπει επίσης να σημειωθεί, ότι η γλώσσα Stan παρέχει μια αυτοματοποιημένη διαδικασία εφαρμογής των αλγορίθμων, η οποία αναλύεται με λεπτομέρεια στην παρούσα εργασία.

# TABLE OF CONTENTS

# LIST OF TABLES

# Chapter 1

# Introduction

In most of the times the density of the parameters in interest, called posterior density, may be difficult to be estimated due to the unknown denominator on the Bayes Theorem, which represents the marginal likelihood and plays the role of the normalizing constant.

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

Due to this problem, several methods have been developed to make Bayesian inference without having to calculate the intractable integral or summation on the denominator. The most common methods are the Markov Chain Monte Carlo, such as Gibbs Sampling, whose role is to sample from the true posterior by constructing a Markov Chain. This Markov Chain represents a sample from the true unknown density. Another method with the same purpose of existence is the Hamiltonian Monte Carlo, which remains a Markov Chain Monte Carlo method; however a significant difference exists. On the other hand, Variational Inference follows a different approach by approximating the unknown density and not sampling from it. Particularly, turns the sampling problem to an optimization problem. According to the case, the suitable method is the one which meets the most of the requirements such as low computational cost and statistical efficiency,

The aforementioned requirements, especially the low computational cost, seems to be essential in cases where the dimensions of the dataset are high, as well as the size. For instance, think of a virus survey where multiple relevant virus indexes are tracked for each individual and according to the result of an appropriate test, each one is assigned into one health status group; positive or negative to the certain virus. Consequently, it is of great importance, especially in the multidimensional case, to take advantage of the whole information being stored into a dataset, efficiently and fast.

The model being described previously, indicates a mixture of two distributions. A common assumption for those distributions i.e. for the Ebola virus, is to be univariate or multivariate Gaussians (Mbala P, Baguelin M, Demiris N et al., 2017), depending upon the number of observations for each individual. Therefore, the need of making Bayesian Inference in mixture models is considered to be indispensable.

# Chapter 2

# Theory behind the methods

## 2.1 Gibbs Sampling

The underlying logic of Markov Chain Monte Carlo (MCMC) sampling is that we can estimate any desired expectation by ergodic averages. That is, we can compute any statistic of a posterior distribution as long as we have N samples from that distribution:

$$E(f(s)) \approx \frac{1}{N}\sum_{i=1}^{N} f((s)^{i}$$

where $P$ is the posterior distribution of interest, $f(s)$ is the desired expectation and $f(s^{(i)})$ is the $i^{th}$ simulated sample from $P$.

Gibbs Sampling is one MCMC technique suitable for the task. Particularly, it generates posterior samples by sweeping through each variable to sample from its conditional distribution keeping fixed the remaining variables to their current values. The procedure continues until the samples are derived from the true posterior density. This situation is called “convergence”. The iterative scheme is illustrated below as Algorithm 1.

**Algorithm 1** Gibbs Sampler

```
$initialize\ \theta_1{}^{(0)}, \theta_2{}^{(0)}, \theta_3{}^{(0)}, \dots, \theta_D{}^{(0)}$
$\boldsymbol{for}\ iteration\ i = 1, 2, \dots\ \boldsymbol{do}$
$\theta_1{}^{(i)} \sim P(\theta_1 | \theta_2{}^{(i-1)}, \theta_3{}^{(i-1)}, \theta_4{}^{(i-1)}, \dots, \theta_D{}^{(i-1)})$
$\theta_2{}^{(i)} \sim P(\theta_2 | \theta_1{}^{(i-1)}, \theta_3{}^{(i-1)}, \theta_4{}^{(i-1)}, \dots, \theta_D{}^{(i-1)})$
 $\vdots$
$\theta_D{}^{(i)} \sim P(\theta_D | \theta_1{}^{(i-1)}, \theta_2{}^{(i-1)}, \theta_3{}^{(i-1)}, \dots, \theta_{D-1}{}^{(i-1)})$
$\boldsymbol{end\ for}$
```

According to Algorithm 1, consider the parameters $\theta_1,\ \theta_2,\ \theta_3, \dots,\ \theta_D$. First step is the initialization of the parameters : $\theta_1{}^{(0)}, \theta_2{}^{(0)}, \theta_3{}^{(0)}, \dots,\ \theta_D{}^{(0)}$. The algorithm begins by sampling the posterior conditionals; one parameter at a time. The number of iterations is considered to be large enough in order to hopefully reach the actual posterior density. The theory of MCMC guarantees the convergence to the true density under large number of iterations (Gilks et al., 1996).

At this project, Gibbs sampler was implemented via OpenBugs; an open-source software package for performing Bayesian Inference using Gibbs sampling. The user specifies a statistical model by simply stating the relationship between the related variables. Then, the software includes a system that performs an MCMC scheme based on the Gibbs sampler for analysing the specified model.

## 2.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) is a Markov Chain Monte Carlo method that uses the derivatives of the density function being sampled to generate efficient transitions spanning the posterior (Betancourt and Girolami, 2013; Neal, 2011). It uses an approximate Hamiltonian dynamics simulation based on numerical integration which is then corrected by performing a Metropolis acceptance step.

It is important to mention that this section translates the presentation of Hamiltonian Monte Carlo by Betancourt and Girolami (2013) into the notation of Gelman et al. (2013).

The algorithm introduces auxiliary momentum variables $\rho$, to the parameters of the unknown posterior density, θ, with the joint density

$$P(\rho,\theta) = P(\rho|\theta)P(\theta)$$

The density of the auxiliary variables is deemed to be the multivariate normal (d dimensions) in most of the cases, as well as in the automatic procedure the probabilistic language Stan offers;

$$\rho \sim \mathcal{N}_d(0,\Sigma)$$

where the covariance matrix Σ works as a Euclidian metric to rotate and scale the unknown-target density (Betancourt and Stein, 2011).

In Stan - in which the HMC algorithm is being implemented - the covariance matrix is usually replaced from the identity matrix or estimated from warmups samples and optionally restricted to a diagonal matrix.

After specifying the conditional density of the momentum variables, the joint density defines a Hamiltonian,

$$\begin{aligned} H(\rho,\theta) &= -logP(\rho,\theta) \\ &= -logP(\rho|\theta) - logP(\theta) \\ &= T(\rho|\theta) + V(\theta) \end{aligned}$$

with the “kinetic energy”

$$T(\rho|\theta) = -logP(\rho|\theta)$$

and the “potential energy”

$$V(\theta) = -logP(\theta)$$

This Hamiltonian function generates a transition by first sampling the auxiliary momentum variables $\rho \sim P(\rho|\theta)$, where $P(\rho|\theta)$ is a distribution independent from the parameters θ when a $\mathcal{N}_d(0, \Sigma)$ is assumed. At next, the joint system is evolving via Hamiltonian's equations,

$$\frac{d\theta}{dt} = +\frac{\partial H}{\partial \rho} = +\frac{\partial T}{\partial \rho}$$

$$\frac{d\rho}{dt} = -\frac{\partial H}{\partial \rho} = -\frac{\partial T}{\partial \theta} - \frac{\partial V}{\partial \theta}$$

The assumption that the momentum variables are independent from the parameters θ, instantly makes equal to 0 the derivative $\partial T / \partial \theta$, leading to the following

$$\frac{d\theta}{dt} = +\frac{\partial T}{\partial \rho}$$

$$\frac{d\rho}{dt} = -\frac{\partial V}{\partial \theta}$$

This two-state differential equation is being solved by using the leapfrog integrator, as Stan suggests. The leapfrog integrator is a numerical integration algorithm that is specifically adapted to provide stable results for Hamiltonian systems of equations. The leapfrog integrator takes discrete steps of some small time integral ε. It begins by drawing a momentum value from the $\rho$ density and then alternates half-step updates of the momentum and full-step updates of the position.

$$\begin{gathered} \rho \leftharpoonup \rho - \frac{\varepsilon}{2}\frac{\partial V}{\partial \theta} \\ \theta \leftharpoonup \theta + \varepsilon \Sigma \rho \\ \rho \leftharpoonup \rho - \frac{\varepsilon}{2}\frac{\partial V}{\partial \theta} \end{gathered}$$

L leapfrog steps are applied and a total of $L\varepsilon$ time is simulated. After the L repetitions of the above three steps, the derived result is two vectors; one for the momentum variable $\rho$ and the other for the parameters $\theta$. Leimkuhler and Reich (2004) provide a detailed analysis of the numerical integration of the Hamiltonian systems.

After the leapfrog implementation, in order to account for numerical errors, a Metropolis Hastings acceptance step is applied and a decision is made whether to update to the new state $(\rho^{new}, \theta^{new})$ or keep the existing state.

## 2.3 Variational Inference

Primarily, it is important to mention that the explanation of Variational Inference is mainly based on the papers of Ormerod and Wand (2010), as well as Kucukelbir et al. (2016).

Variational approximations are not widely known within the statistical community as the Monte Carlo methods, especially MCMC, for performing approximate inference, as well as Laplace approximation methods (Ormerod and Wand, 2010).

There are several approaches to variational approximations with the density transform approach to be possibly the most known one. This approach involves the approximation of intractable densities by others for which inference is more tractable. The Kullback – Leibler divergence constitutes the main ingredient in these approximations, working as the dissimilarity function which measures the distance between the true density and its approximation. The explanation is shown below in more details.

The Bayes' theorem defines the posterior density of the parameters as following,

$$\frac{P(y|\theta)P(\theta)}{P(y)}$$

where the denominator is called marginal likelihood or model evidence and it is an integral or a summation depending on the nature of the random variable. Throughout this section we discuss the continuous case; nonetheless, the discrete case has a similar treatment.

Let's assume a density function $q$ over the parameter space $\Theta$, with parameters $\varphi$**.** Note that $\theta$ and $\varphi$ are vectors. Then the logarithm of the marginal likelihood could be treated as follows,

**Equation 1**

$$
\begin{aligned}
logP(y) = logP(y)\int q(\theta;\varphi)d\theta &= \int q(\theta;\varphi)logP(y)d\theta \\
&= \int q(\theta;\varphi)log\left\{\frac{P(y,\theta)/q(\theta;\varphi)}{P(\theta|y)/q(\theta;\varphi)}\right\}d\theta \\
&= \int q(\theta;\varphi)log\left\{\frac{P(y,\theta)}{q(\theta;\varphi)}\right\}d\theta \;+\; \int \boldsymbol{q(\theta;\varphi)log\left\{\frac{q(\theta;\varphi)}{P(\theta|y)}\right\}d\theta}
\end{aligned}
$$

The Kullback – Leibler divergence is appearing as the second term on the right side of the Equation 1. According to Jensen's Inequality for concave functions, the Kullback – Leibler divergence is equal or greater than zero for all densities $q$, with equality if and only if $q(\theta;\varphi) = P(\theta|y)$

$$
KL(q(\theta;\varphi)||P(\theta|y)) = \int q(\theta;\varphi)log\left\{\frac{q(\theta;\varphi)}{P(\theta|y)}\right\}d\theta \;\geq\; 0
$$

Therefore, the inequality 2 is resulted:

**Inequality 2**

$$
logP(y) \geq \int q(\theta;\varphi)log\left\{\frac{P(y,\theta)}{q(\theta;\varphi)}\right\}d\theta
$$

$$
P(y) \geq \boldsymbol{exp}\left(\int \boldsymbol{q(\theta;\varphi)log\left\{\frac{P(y,\theta)}{q(\theta;\varphi)}\right\}d\theta}\right) \equiv \underline{P}(y;q)
$$

where the exponential integral on the right is called ELBO (Evidence Lower Bound).

To sum up, the variational approximations suggest instead of making inference for $P(\theta|y)$ for which the $P(y)$ is intractable, to make inference for the density $q(\theta;\varphi)$ which has a more tractable $\underline{P}(y;q)$.

The next step concerns the density $q$ and the family of distributions that it belongs. In particular, it is useful to restrict the family of densities to one that contains tractable densities and then find the one that minimizes the $KL(q(\theta;\varphi)||P(\theta|y))$ or equivalently maximizes the ELBO[1].

The choice of the aforementioned family is based on the paper of Kucukelbir et al. (2016) which proposes the Automatic Differentiation Variational Inference in the probabilistic language Stan.

First step is the transformation of the support space of the parameters $\theta$ in order to ensure that they live to the real coordinate space $\mathbb{R}^K$.

$$Z: supp(P(\theta)) \to \mathbb{R}^K$$

Then the transformed parameters are identified as $\omega = Z(\theta)$ for which the support space is the real coordinate space $\mathbb{R}^K$. Stan provides a library of transformations along with their Jacobians; hence any differentiable probability model can be represented by one with real-valued variables.

In the second step, assumptions about the family of the density $q(\omega;\varphi)$ are made. Specifically, the variational density $q$ is considered to be Gaussian in the real coordinate space. Concerning the original parameter space, the variational distributions $q(\theta;\varphi)$ may be non-Gaussian.

At this point, one option is to extend the analysis and apply the Mean – Field approximation by assuming that the variational density $q(\omega;\varphi)$ is a product of $M$ Gaussians; to wit the elements of the transformed vector $\theta, \omega,$ are independent. This approach is common in literature.

---

[1] $ELBO \equiv log\underline{P}(y;q) = -KL(q(\theta;\varphi)||P(\theta|y)) + terms\ not\ invoving\ q$

**Equation 3**

$$q(\omega;\varphi) = \mathcal{N}_M(\omega;\mu,\Sigma) = \prod_{m=1}^{M} \mathcal{N}(\omega_m;\mu_m,\sigma_m^2)$$

where $\Sigma$ is the diagonal covariance matrix with the variances $\sigma_1^2,\ \sigma_2^2, \dots,\ \sigma_M^2$ on the diagonal, $\mu$ is the mean vector of the of the Gaussian with elements $\mu_1,\ \mu_2, \dots,\ \mu_M,\ \omega$ is the transformed vector of the parameters $\theta$ and $\varphi = (\mu_1,\ \mu_2, \dots,\ \mu_M,\ \sigma_1^2,\ \sigma_2^2, \dots,\ \sigma_M^2)$ are considered to be the variational parameters. Particularly, the first equality in Equation 3 defines a multivariate Gaussian ($M$ dimensions) which is equivalent to the product of $M$ univariate due to the idependence assumption.

In the third step, the constrain regarding the space where the variational parameters $\varphi$ live, is removed by taking the logarithm of the standard deviations. This happens because the variances $\sigma_1^2,\ \sigma_2^2, \dots,\ \sigma_M^2$ must always be positive forcing the variational space to be $\Phi = \{\mathbb{R}^K,\ \mathbb{R}^K{}_{>0}\}$. Therefore, this constrain is removed if $\zeta = log(\sigma)$ and consequently, the variational $\varphi$ are $\mu_1,\ \mu_2, \dots,\ \mu_M,\ \zeta_1,\ \zeta_2, \dots,\ \zeta_M$ and their parameter space is unconstrained to $\mathbb{R}^{2K}$.

The next step concerns the optimization of the ELBO with respect to the variational parameters $\varphi$, which they live in an appropriate real coordinate space and thus, there is no need to worry about the support matching constrain issue. In Stan, this unconstrained optimization problem is solved via a stochastic gradient ascent algorithm that uses automatic differentiation to compute gradients and Monte Carlo integration to approximate expectations.

The automatic differentiation procedure cannot be used directly on the ELBO because it includes an unknown expectation (see Equation 4, first term on the right side)

**Equation 4**[2]

$$\underline{P}(y;q) = \mathbf{E}_{\boldsymbol{q(\omega;\varphi)}}\left[\boldsymbol{logP}\left(\boldsymbol{y},\boldsymbol{Z^{-1}(\omega)}\right) + \boldsymbol{log|detJ_{Z^{-1}}(\omega)|}\right] + \int q(\omega;\varphi)log[q(\omega;\varphi)]d\omega$$

---

[2] concerns the ELBO in the real coordinate space

Thus, the gradient operator is inserted into the expectation in order to differentiate automatically the functions stored inside. To accomplish that, a last transformation is required; the elliptical standardization. In particular, consider a transformation $T_\varphi$ that absorbs the variational parameters $\varphi$. This standardization works as a convertor of the Gaussian approximation to a standard Gaussian. In the Mean – Field approximation the standardization is $\lambda = T_\varphi(\omega) = diag(\exp(\zeta))^{-1}(\omega - \mu)$; hence the variational densities take the following form:

$$q(\lambda) = \mathcal{N}_M(\lambda; 0, I) = \prod_{m=1}^{M} \mathcal{N}(\lambda_m; 0, 1)$$

The elliptical standardization transforms the unknown expectation in Equation 4 to an expectation in terms of a standard Gaussian density. Since the ELBO in no longer depending on $\varphi$, the gradients are directly calculated by being inserted into the known expectation. The optimization problem is solved with the implementation of the stochastic gradient ascent algorithm.

# Chapter 3

# Implementation in Mixture of Gaussians

Mixture of distributions are considered to be one of the most well-known ways to capture the non-systematic behaviour of some data, due to the weighted information provided by multiple distributions. The mixture of densities is widely used and studied, especially the Gaussian mixture which suggests multimodal data densities allowing the population to be spilt into sub-populations.

On this project, the focus lies on these kind of models which include latent indicators responsible for allocating the data point to the sub-populations (components), in order to simplify the calculations by reducing the complexity of the model. On the other hand, the mixture model can be written in Stan language without involving any of those latent indicators; just leaving the model in the original form with the mixing proportions. The likelihood of two univariate Gaussians below is presented with no latent indicators:

$$P(y;\theta) = p\mathcal{N}(\mu_1, \sigma_1^2) + (1-p)\mathcal{N}(\mu_2, \sigma_2^2)$$

where $\theta = (\mu_1,\ \mu_2,\ \sigma_1^2,\ \sigma_2^2,\ p)$ denotes the set of the parameters, $\mu_i$ indicates the mean value of the $i^{th}$ component, $\sigma_i^2$ the variance of the values in the $i^{th}$ component and $p$ the mixing proportion.

Consequently, the computer intensive methods: Variational Inference and Hamiltonian Monte Carlo, are implemented in mixture models coded in Stan language for which the latent structure can be omitted. On the other hand, the latent structure is adopted in the case of Gibbs Sampling in BUGS, since it is the only way to structure the mixture model in that software.

## 3.1 Examples in Mixture of Gaussians

Stan language constitutes a new probabilistic language for Bayesian Inference and especially, MCMC and Variational Bayes; Variational Bayes is the same to Variational Inference when the Mean – Field approach is adopted (Ormerod and Wand, 2010). In more details, the variational density of the parameter set is assumed to be the product of the marginal variational densities of each element of the parameter set.

At this stage, it is of great importance to mention that Stan, as a new language, is under development, especially in the part of Variational Inference and corrections are made occasionally. Nevertheless, it was deemed interesting to use Stan as one of the main softwares, in order to test its capabilities or even suggest corrections/simplifications that might be helpful in the future.

In this chapter, the examples presented concern mixture of Gaussian distributions for the likelihood. The procedure in terms of model's structure in Stan is similar for any mixture model, as well as for any number of components; therefore, the implementation is made on mixture of two univariate Gaussians for two reasons: a) the Gaussian is the most applicable distribution because most things in the nature tend to follow a normal behaviour and b) it is quite often the output of a test to be binary, indicating two groups according to the result of a suitable test; for instance, a group of individuals positive to a tested virus and a group that includes individuals negative to it. At this point is important to mention that efforts had been made to apply those algorithms, especially the ones suggested by Stan, in a quadrivariate mixture of Gaussians facing difficulties that are discussed at next.

In the aforementioned two cases, the statistical efficiency in terms of Mean Squared Error (MSE) and Bias, as well as the speed of the algorithms (CPU time), are tested. The first case concerns a mixture of two univariate Gaussians with well separated components and the second when the components are not so well separated, in order to test the performance of the algorithms in non-mixed and mixed cases.

For both cases, multiple datasets have been simulated with the data size varying from 10 to $10^5$ in order to test each algorithm's performance in small, medium and big datasets. The tables containing the MSE, Bias and CPU time are shown below, but first the forms of the aforementioned measurements are given for each of the five parameters of the univariate Gaussian mixture model $\theta = (\mu_1,\ \mu_2,\ \sigma_1^2,\ \sigma_2^2,\ p)$ :

$$MSE_k = \frac{1}{N}\sum_{i=1}^{N}(\widehat{\theta_{k_i}} - \theta_k)^2, \quad k = 1, \dots, 5$$

with $N$: number of simulated datasets, $\theta_k$: the true value of the $k^{th}$ parameter and $\widehat{\theta_{k_i}}$: the corresponding estimated value derived by the implementation of the chosen algorithm.

$$Bias_k = \mathrm{E}\big(\widehat{\theta_k}\big) - \theta_k, \quad k = 1, \dots, 5$$

where $\mathrm{E}\big(\widehat{\theta_k}\big) = \sum_{i=1}^{N} \widehat{\theta_{k_i}}$ is the mean value of the $k^{th}$ parameter when the sample size is $N$.

As regards CPU time, it is the amount of time measured in seconds for which CPU was used for processing the algorithms.

Overall, lower values of those three quantities indicate statistical efficiency and low computational cost respectively. To be more precise, MSE and Bias values are ideally to be close to zero. Moreover, it must be mentioned that the number of datasets simulated for each case is chosen to be 100, as this number is decent in terms of size and may increase the possibilities of deriving significant results.

## 3.2 Testing the performance of the algorithms

As it is already mentioned, pursues the performance of Gibbs Sampling (Gibbs), Variational Bayes (VB) and Hamiltonian Monte Carlo (HMC) in different cases. It is important to mention that the number of iterations in the MCMC algorithms is chosen to be 2000 with burn-in equivalent to 1000.

First case

- $\mathbf{0.3}\boldsymbol{\mathcal{N}}(\mathbf{10},\mathbf{3^2})+\mathbf{0.7}\boldsymbol{\mathcal{N}}(\mathbf{-10},\mathbf{2^2})$ - well separated Gaussian distributions

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Data size: **10** | **MEAN SQUARE ERROR** | | | | | | | | |
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | [3]*Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**:<br>***0.094*** | Comp.**1**:<br>*0.103* | Comp.**1**:<br>*0.097* | Comp.**1**:<br>*0.080* | Comp.**1**:<br>*0.086* | Comp.**1**:<br>***0.075*** | *0.009* | *0.013* | ***0.006*** |
| | Comp.**2**:<br>***0.105*** | Comp.**2**:<br>*0.497* | Comp.**2**:<br>*0.398* | Comp.**2**:<br>*0.720* | Comp.**2**:<br>*0.509* | Comp.**2**:<br>***0.306*** | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1:**<br>***-0.007*** | Comp.**1**:<br>*-0.892* | Comp.**1**:<br>*0.031* | Comp.**1**:<br>*-0.047* | Comp.**1**:<br>*-0.097* | Comp.**1:**<br>***-0.036*** | ***0.004*** | *0.011* | *0.005* |
| | Comp.**2**:<br>***-0.054*** | Comp.**2**:<br>*0.110* | Comp.**2**:<br>*0.098* | Comp.**2**:<br>*0.107* | Comp.**2**:<br>*0.217* | Comp.**2**:<br>***0.083*** | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | ***VB*** | | | ***HMC*** | | |
| | 88.75 | | | **9.35** | | | 135.46 | | |

**Table 3.2.1:** MSE, Bias and CPU time of each algorithm when the two components are well-separated and the size of each of the 100 datasets is 10

In the Table 3.2.1, the performance of each algorithm seems to be respectable as the values of the tested quantities – MSE and Bias - are close to zero. Variational

---

[3] $p$ is the probability of the first component $\boldsymbol{\mathcal{N}}(\mathbf{10},\mathbf{3^2})$ being realized and $1-p$ the probability of the second component $\boldsymbol{\mathcal{N}}(\mathbf{-10},\mathbf{2^2})$

Inference might be less significant than the others; however the difference is not considerably high. Nevertheless, its speed is quite higher (CPU time = 9.35 seconds).

| Data size: $10^2$ | **MEAN SQUARE ERROR** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean: $\mu$* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**: *0.064* | Comp.**1**: *0.063* | Comp.**1**: ***0.057*** | Comp.**1**: *0.040* | Comp.**1**: *0.036* | Comp.**1**: ***0.025*** | ***0.002*** | *0.008* | ***0.002*** |
| | Comp.**2**: ***0.085*** | Comp.**2**: *0.297* | Comp.**2**: *0.298* | Comp.**2**: *0.510* | Comp.**2**: *0.209* | Comp.**2**: ***0.207*** | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: $\mu$* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1:** ***-0.003*** | Comp.**1**: *-0.592* | Comp.**1**: *0.021* | Comp.**1**: *-0.027* | Comp.**1**: *-0.067* | Comp.**1:** ***-0.015*** | ***0.002*** | *0.008* | *0.005* |
| | Comp.**2**: *-0.084* | Comp.**2**: *0.080* | Comp.**2**: ***0.078*** | Comp.**2**: *0.167* | Comp.**2**: *0.087* | Comp.**2**: ***0.053*** | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | ***VB*** | | | ***HMC*** | | |
| | 137.78 | | | **21.75** | | | 292.69 | | |

**Table 3.2.2:** MSE, Bias and CPU time of each algorithm when the two components are well-separated and the size of each of the 100 datasets is 100

In the Table 3.2.2, it is shown that all the algorithms produced quite accurate results for the whole parameter set, in this specific case, since the MSE, as well as the Bias are close to zero everywhere. As regards the time cost, Variational Bayes (or Inference) seems to be considerably faster than the rest of the algorithms (CPU time = 21.75 seconds).

| Data size: $10^3$ | **MEAN SQUARE ERROR** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.1:<br>***0.006*** | Comp.1:<br>*0.039* | Comp.1:<br>*0.027* | Comp.1:<br>***0.005*** | Comp.1:<br>*0.035* | Comp.1:<br>*0.021* | ***0.001*** | *0.008* | ***0.001*** |
| | Comp.2:<br>***0.024*** | Comp.2:<br>*0.209* | Comp.2:<br>*0.027* | Comp.2:<br>***0.016*** | Comp.2:<br>*0.112* | Comp.2:<br>*0.107* | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.1:<br>***-0.001*** | Comp.1:<br>*-0.027* | Comp.1:<br>*0.020* | Comp.1:<br>*-0.018* | Comp.1:<br>*-0.056* | Comp.1:<br>***-0.013*** | *-0.003* | *0.008* | ***-0.001*** |
| | Comp.2:<br>***0.005*** | Comp.2:<br>*0.012* | Comp.2:<br>*0.006* | Comp.2:<br>*0.057* | Comp.2:<br>*0.064* | Comp.2:<br>***0.045*** | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | ***VB*** | | | ***HMC*** | | |
| | 605.70 | | | **53.78** | | | 1154.20 | | |

**Table 3.2.3:** MSE, Bias and CPU time of each algorithm when the two components are well-separated and the size of each of the 100 datasets is 1,000

In the Table 3.2.3, the sample size increases to 1,000 and the MSE/Bias values approach faster the zero value in each case. It could be noticed that Gibbs Sampling, as well as Hamiltonian Monte Carlo may produce slightly more accurate results than Variational Bayes, since they appear to derive lower values, but the differences seem to be negligible. Once more, Variational Bayes is considerably faster.

| Data size: $\mathbf{10^4}$ | **MEAN SQUARE ERROR** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean: $\mu$* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**: - | Comp.**1**: *0.001* | Comp.**1**: - | Comp.**1**: - | Comp.**1**: *0.002* | Comp.**1**: - | - | *0.000* | - |
| | Comp.**2**: - | Comp.**2**: *0.004* | Comp.**2**: - | Comp.**2**: - | Comp.**2**: *0.011* | Comp.**2**: - | | | |
| | | | | | | | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: $\mu$* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**: - | Comp.**1**: *0.007* | Comp.**1**: - | Comp.**1**: - | Comp.**1**: *0.026* | Comp.**1**: - | - | *0.005* | - |
| | Comp.**2**: - | Comp.**2**: *-0.010* | Comp.**2**: - | Comp.**2**: - | Comp.**2**: *0.058* | Comp.**2**: - | | | |
| | | | | | | | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | ***VB*** | | | ***HMC*** | | |
| | $\gg$ 963.30 | | | **963.30** | | | $\gg$ 963.30 | | |

**Table 3.2.4:** MSE, Bias and CPU time of each algorithm when the two components are well-separated and the size of each of the 100 datasets is 10,000

In the Table 3.2.4, the sample size has reached 10,000 datapoints per dataset. The computational cost of this simulation is high, but Variational Bayes managed to complete the process within 963.30 seconds. On the other hand, the MCMC algorithms were considerably slow without deriving any results after a long hour. Hence, it was made the decision to stop the procedure since Variational Inference had already converged.

| Data size: $\mathbf{10^5}$ | **MEAN SQUARE ERROR** | | |
|---|---|---|---|
| | *Mean: $\mu$* | *Standard deviation: $\sigma^2$* | *Mixing Proportion: p* |
| | ***VB*** | *VB* | ***VB*** |
| | Comp.**1**: *0.001* | Comp.**1**: *0.002* | *0.000* |
| | Comp.**2**: *0.004* | Comp.**2**: *0.011* | |
| | **BIAS** | | |
| | *Mean: $\mu$* | *Standard deviation: $\sigma^2$* | *Mixing Proportion: p* |
| | ***VB*** | *VB* | ***VB*** |
| | Comp.**1**: *0.007* | Comp.**1**: *0.026* | *0.005* |
| | Comp.**2**: -*0.010* | Comp.**2**: *0.058* | |
| | **CPU TIME (in sec)** | | |
| | *VB* | | |
| | **9879.93** | | |

**Table 3.2.5:** MSE, Bias and CPU time of Variational Bayes when the two components are well-separated and the size of each of the 100 datasets is 100,000

In Table 3.2.5 are presented the results of Variational's Bayes performance in 100 datasets with 100,000 data points each. The algorithm is very fast in terms of CPU time. The MCMC algorithms weren't able to converge within a comparable time framework, due to the high computational cost in the calculations.

To sum up the aforementioned conclusions, all the algorithms in general seemed to be statistical efficient in the case of two well separated univariate Gaussians. Variational Inference in smaller datasets, might be less efficient in comparison to the MCMC algorithms, but the difference was not so noticeable. Regarding the time required for the algorithm to converge, Variational Inference (or Bayes) was considerably the fastest algorithm, rendering itself a quite useful tool in cases of massive datasets. The

MCMC algorithms didn't show the same capability in the bigger problems due to their difficulty in scaling to large datasets.

Second case

- $\mathbf{0.3\mathcal{N}(1.75, 3^2) + 0.7\mathcal{N}(-1.75, 2^2)}$ – not so well separated Gaussians[4]

| Data size: **10** | **MEAN SQUARE ERROR** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**: ***0.109*** | Comp.**1**: *0.189* | Comp.**1**: *0.112* | Comp.**1**: *0.098* | Comp.**1**: *0.180* | Comp.**1**: ***0.092*** | *0.013* | *0.025* | ***0.011*** |
| | Comp.**2**: ***0.167*** | Comp.**2**: *0.503* | Comp.**2**: *0.420* | Comp.**2**: *0.567* | Comp.**2**: *0.689* | Comp.**2**: ***0.367*** | | | |
| | | | | | | | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.**1**: ***-0.009*** | Comp.**1**: *-0.986* | Comp.**1**: *0.045* | Comp.**1**: *-0.053* | Comp.**1**: *-0.178* | Comp.**1**: ***-0.045*** | ***0.009*** | *0.020* | ***0.009*** |
| | Comp.**2**: ***-0.064*** | Comp.**2**: *0.189* | Comp.**2**: *0.100* | Comp.**2**: *0.178* | Comp.**2**: *0.294* | Comp.**2**: ***0.098*** | | | |
| | | | | | | | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | *VB* | | | ***HMC*** | | |
| | 92.25 | | | **11.15** | | | 146.12 | | |

**Table 3.2.6:** MSE, Bias and CPU time of each algorithm when the two components are *not* so well-separated and the size of each of the 100 datasets is 10

According to Table 3.2.6, the MSE and Bias values are systematically higher for Variational Bayes in comparison to the MCMC algorithms, for the case of mixed components. This situation may denote that the aforementioned algorithm tends to

---

[4] the tables for data size equal to 10 and 1,000 are provided, since the first case represents the performance of the algorithms in small data sets and the second, in big ones

produce less accurate estimates in mixed cases because it approximates the target density rather than sampling from the asymptotically exact posterior. At this point, it is important to mention that the sample size was quite small, consequently it was expected to have less significant estimates, especially for Variational Inference. Nevertheless, the values are not significantly higher than those of the MCMC algorithms, indicating than Variational Bayes still could be used as the main algorithm in the mixture of mixed Gaussians with not so well separated components, according to this example. Moreover, as expected, its speed outperforms the one of the others. As regards Hamiltonian Monte Carlo and Gibbs Sampling, their performance in that case cannot distinguish which one is more preferred; both are statistically efficient.

| Data size: $10^3$ | **MEAN SQUARE ERROR** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.1: **0.017** | Comp.1: 0.042 | Comp.1: 0.056 | Comp.1: **0.010** | Comp.1: 0.067 | Comp.1: 0.053 | **0.004** | 0.012 | 0.005 |
| | Comp.2: 0.078 | Comp.2: 0.387 | Comp.2: **0.076** | Comp.2: **0.023** | Comp.2: 0.178 | Comp.2: 0.114 | | | |
| | | | | | | | | | |
| | **BIAS** | | | | | | | | |
| | *Mean: μ* | | | *Standard deviation: $\sigma^2$* | | | *Mixing Proportion: p* | | |
| | *Gibbs* | ***VB*** | *HMC* | ***Gibbs*** | *VB* | ***HMC*** | *Gibbs* | ***VB*** | *HMC* |
| | Comp.1: **-0.005** | Comp.1: -0.035 | Comp.1: 0.032 | Comp.1: -0.026 | Comp.1: -0.065 | Comp.1: **-0.025** | -0.006 | 0.015 | **-0.004** |
| | Comp.2: **0.009** | Comp.2: 0.026 | Comp.2: 0.011 | Comp.2: 0.065 | Comp.2: 0.077 | Comp.2: **0.053** | | | |
| | | | | | | | | | |
| | **CPU TIME (in sec)** | | | | | | | | |
| | ***Gibbs*** | | | ***VB*** | | | ***HMC*** | | |
| | 611.63 | | | **58.74** | | | 1168.54 | | |

**Table 3.2.7:** MSE, Bias and CPU time of each algorithm when the two components are not so well-separated and the size of each of the 100 datasets is 1,000

In Table 3.2.7, the performance of the algorithms is proportional to the one discussed in Table 3.2.6, with the only difference lying on the bigger sample size which automatically increases the statistical efficiency of the algorithms.

To sum up the aforementioned, Variational Inference as expected was the fastest algorithm in terms of CPU time in seconds. On the other hand, its accuracy decreased due to the mixing components, however this decrease was not considerably high because the framework of Gaussian distributions may favored its performance, as well as the fact than constrain for label switching was inserted into the Stan model. The MCMC algorithms seem both to perform well, with Gibbs Sampling probably being more accurate in terms of MSE and Bias. Furthermore, both of them were slow, with Hamiltonian Monte Carlo being slower.

<u>Last case</u>

The algorithms provided in Stan were implemented for a quadrivariate mixture of two Gaussians. Despite the fact that similar procedure as the univariate case was pursued in structuring the code, both of the algorithms made unreasonably unstable estimation for the parameters in comparison to Gibbs Sampling in BUGS. At this point, it would be quite helpful to highlight that Stan is under development, especially for the Variational Inference method, since a warning message is deriving after implementation letting us know that the algorithm is on experimental level and may produce unstable results or even wrong. More reasons could be that coding in Stan, may be complex if the user starts building the model on her/his own without consulting similar examples.

Therefore, it is considered decent to recheck the results of Stan's output in the future by applying the algorithms manually in different softwares.

## 3.3 Some point estimates along with relevant values

It is interesting to present the point estimation of the mixture parameters derived by the three algorithms, along with their standard deviation, as well as the effective sample sizes for the MCMC algorithms. Two data sets of different size were chosen and the aforementioned values are illustrated below:

True values: $\boldsymbol{\mu_1 = 10,\ \mu_2 = -10,\ \sigma_1^2 = 3,\ \sigma_2^2 = 2,\ p = 30\%,\ 1 - p = 70\%}$

| | *Gibbs Sampling* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *Mean values (point estimates)* | | | *Standard deviation of the Mean values* | | | *Effective Sample size* | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ |
| Data size: $10^3$ | Comp.**1**: *9.9* | Comp.**1**: *3.16* | Comp.**1**: *30%* | Comp.**1**: *0.2* | Comp.**1**: *0.001* | *0.01* | Comp.**1**: *1000* | Comp.**1**: ***799*** | *1000* |
| | Comp.**2**: *-10* | Comp.**2**: *1.82* | Comp.**2**: *70%* | Comp.**2**: *0.1* | Comp.**2**: *0.001* | | Comp.**2**: *1000* | Comp.**2**: ***800*** | |
| | *Variational Inference* | | | | | | | | |
| | *Mean values (point estimates)* | | | *Standard deviation of the Mean values* | | | *Effective Sample size* | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | – | | |
| | Comp.**1**: *9.87* | Comp.**1**: *2.83* | Comp.**1**: *28%* | Comp.**1**: *0.17* | Comp.**1**: *0.12* | *0.02* | | | |
| | Comp.**2**: *-10.10* | Comp.**2**: *2.05* | Comp.**2**: *32%* | Comp.**2**: *0.08* | Comp.**2**: *0.05* | | | | |
| | *Hamiltonian Monte Carlo* | | | | | | | | |
| | *Mean values (point estimates)* | | | *Standard deviation of the Mean values* | | | *Effective Sample size* | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ |
| | Comp.**1**: *9.86* | Comp.**1**: *2.95* | Comp.**1**: *29%* | Comp.**1**: *0.01* | Comp.**1**: *0.001* | *0.01* | Comp.**1**: *1000* | Comp.**1**: *1000* | *1000* |
| | Comp.**2**: *-9.94* | Comp.**2**: *1.98* | Comp.**2**: *71%* | Comp.**2**: *0.001* | Comp.**2**: *0.001* | | Comp.**2**: *1000* | Comp.**2**: *1000* | |

**Table 3.3.1:** Point estimates along with their standard deviation and the effective sample size for each algorithm when the dataset of size 1,000 is derived from a mixture of two univariate Gaussians with well separated components

True values: $\boldsymbol{\mu_1 = 10,\ \mu_2 = -10,\ \sigma_1^2 = 3,\ \sigma_2^2 = 2,\ p = 0.3}$

| Data size: $\mathbf{10^5}$ | ***Gibbs Sampling*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | ***Mean values (point estimates)*** | | | ***Standard deviation of the Mean values*** | | | ***Effective Sample size*** | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ |
| | Comp.**1**: *10* | Comp.**1**: *3.16* | Comp.**1**: *30%* | Comp.**1**: *0.01* | Comp.**1**: *0.001* | *0.01* | Comp.**1**: ***800*** | Comp.**1**: *1000* | *1000* |
| | Comp.**2**: *-10* | Comp.**2**: *2* | Comp.**2**: *70%* | Comp.**2**: *0.01* | Comp.**2**: *0.001* | | Comp.**2**: *1000* | Comp.**2**: *1000* | |
| | ***Variational Inference*** | | | | | | | | |
| | ***Mean values (point estimates)*** | | | ***Standard deviation of the Mean values*** | | | ***Effective Sample size*** | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | – | | |
| | Comp.**1**: *10.07* | Comp.**1**: *3.06* | Comp.**1**: *28%* | Comp.**1**: *0.01* | Comp.**1**: *0.02* | *0.001* | | | |
| | Comp.**2**: *-10.08* | Comp.**2**: *1.98* | Comp.**2**: *32%* | Comp.**2**: *0.02* | Comp.**2**: *0.02* | | | | |
| | ***Hamiltonian Monte Carlo*** | | | | | | | | |
| | ***Mean values (point estimates)*** | | | ***Standard deviation of the Mean values*** | | | ***Effective Sample size*** | | |
| | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ | $\mu$ | $\sigma^2$ | $p$ |
| | Comp.**1**: *9.98* | Comp.**1**: *3.00* | Comp.**1**: *30%* | Comp.**1**: *0.001* | Comp.**1**: *0.001* | *0.01* | Comp.**1**: *1000* | Comp.**1**: *1000* | *1000* |
| | Comp.**2**: *-9.99* | Comp.**2**: *2.00* | Comp.**2**: *70%* | Comp.**2**: *0.001* | Comp.**2**: *0.001* | | Comp.**2**: *1000* | Comp.**2**: *1000* | |

**Table 3.3.2:** Point estimates along with their standard deviation and the effective sample size for each algorithm when the dataset of size 100,000 is derived from a mixture of two univariate Gaussians with well separated components

According to Table 3.3.1 and Table 3.3.2, all the algorithms derived quite accurate point estimates for the parameters of this mixture model, since the values seem to be pretty close to the true ones in each case. As regards the effective sample size of the MCMC algorithms, Hamiltonian Monte Carlo in this case, produced 1,000 uncorrelated samples out of 1,000 (number of samples is 2,000 but the first 1,000 are discarded). On the other hand, Gibbs Sampling had high effective samples sizes as well, for all the parameters, except from the variances of the components in Table 3.3.1 and the mean value of the first component in Table 3.3.2 where values lower than 1,000 were produced, indicating that out of the 1,000 posterior samples, at most 800 are uncorrelated.

# Chapter 4

# Conclusions

Variational Inference is considered to be the fastest algorithm in terms of the CPU time required for the algorithm to produce results, rendering this method pretty useful in massive data problems, where the need for fast inference is indisputable. Second comes Gibbs Sampling with considerable lower speed in terms of CPU time and last Hamiltonian Monte Carlo, due to the extra time required for the auxiliary variables to be computed at each iteration. At this stage, it must be mentioned once more that a part of the differences in speed may be due to the different softwares being used to implement the algorithms.

Hamiltonian Monte Carlo and Gibbs Sampling were also tested in their ability to derive uncorrelated data points by calculating the effective sample size for each parameter. Hamiltonian Monte Carlo performed better than Gibbs Sampling, since the effective sample sizes for each parameter of the mixture model were greater, indicating that the algorithm tends to produce uncorrelated data points. In terms of statistical efficiency, all the algorithms for both models- one Gaussian mixture with two well separated components and another with two not so well separated - derived quite accurate results. In the latter case, the statistical efficiency was lower, however the estimates were quite accurate. Gibbs Sampling and Hamiltonian Monte Carlo, as MCMC methods which sample asymptotically from the exact posterior, were expected to be efficient. Variational Inference, especially in the latter case where the two components mix, derived less accurate results than the MCMC algorithms, however the differences in terms of MSE and Bias seemed to be insignificant. It is important to note that Variational Inference was expected to perform even poorly in the estimation of the two variances, since according to the literature (Bishop, 2006) it tends to underestimate them; on the other hand, the way that it is implemented in Stan, as well as the framework of the normality, may favoured its performance. To sum up, according to the aforementioned results and the literature, MCMC methods despite of being popular for their ability to sample from the asymptotically exact posterior, they cannot be scaled in large datasets. On the other hand, Variational Inference is capable of being implemented without hesitation in massive datasets, but in general it is inclined to underestimate the variance in the Gaussian problems.

In the multivariate case, the results of Variational Inference and Hamiltonian Monte Carlo weren't accurate in spite of having coded correctly the multivariate mixture model in Stan terms. At this point, it is crucial to mention and highlight that a warning message appeared after each implementation of Variational's Inference command revealing that the algorithm is experimental, since the procedure has not been thoroughly tested and may be unstable or buggy; hence the interface is about to change. Regarding Hamiltonian Monte Carlo, the results in the simulation procedure weren't derived since the algorithm got stuck after the first iterations.

Consequently, it is wise to test further the results in the multivariate case, as well as in the univariate by implementing manually the algorithms in another software until Stan is tested thoroughly. Nevertheless, it was considered important as well as interesting to use Stan as one of the main softwares, since it suggests a black box procedure for Variational Inference and Hamiltonian Monte Carlo, rendering it a quite useful tool after its upgrade. Moreover, it was deemed reasonable to test its capabilities in practice and make suggestions as regards corrections or simplifications in coding, such as the need for increasing the flexibility of the user to structure on her/his own way the mixture model. So far, it might be tricky for the user to begin constructing a complex model by him/herself in Stan coding, without seeking for advice in similar examples in the literature.

## 4.1 Future Research

There are several avenues for further investigation that we were not able to pursue due to time constrains. As mentioned before, Variational Inference and Hamiltonian Monte Carlo could be implemented manually in an alternative language, such as R or Python, for univariate and multivariate mixture models in order to test the performance of the algorithms, as well as to compare the results with Stan. Furthermore, it would be interesting enough to work with mixture factorial experiments (Nobile and Green, 2000) by applying the aforementioned algorithms.

In addition, a quite challenging and simultaneously promising investigation could be the use of Variational Inference in combination with MCMC sampling algorithms to enable Bayesian inference to meet the demands of the new “big data” era.

# Bibliography

Betancourt, M. and Girolami, M. (2013). Hamiltonian Monte Carlo for hierarchical models. https://arxiv.org/abs/1312.0906

Betancourt, M. and Stein, L. C. (2011). The geometry of Hamiltonian Monte Carlo. https://arxiv.org/abs/1112.4118

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis*. Chapman &Hall/CRC Press, London, third edition. 95, 112, 114, 116, 146, 167, 258, 369, 380, 436, 438, 446, 469, 495, 509

Gelman, A., Roberts, G.O., Gilks, W.R. (1996). Efficient Metropolis jumping rules, *Bayesian Statistics V*, 599-608, Clarendon press, Oxford, 1996

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., Blei, D. (2016). Automatic Differentiation Variational Inference. https://arxiv.org/abs/1506.03431

Leimkuhler, B., Reich, S. (2004). *Simulating Hamiltonian Dynamics*. Cambridge University Press. 471

Mbala, P., Baguelin, M., Ngay, I., Rosello, A., Mulembakani, P., Demiris, N., Edmunds W.J., Muyembe J-J. (2017). Evaluating the frequency of asymptomatic Ebola virus infection. *Phil. Trans. R. Soc.* B 372. http://dx.doi.org/10.1098/rstb.2016.0303

Neal, R. M. (2011). MCMC using Hamiltonian Dynamics, *Handbook of Markov Chain Monte Carlo,* 2:113-162. Chapman & Hall / CRC Press

Nobile, A., Green, P.J. (2000). Bayesian analysis of factorial experiments by mixture modelling, *Biometrika, Volume 87, Issue 1, 1 March 2000, Pages 15–35.* https://doi.org/10.1093/biomet/87.1.15

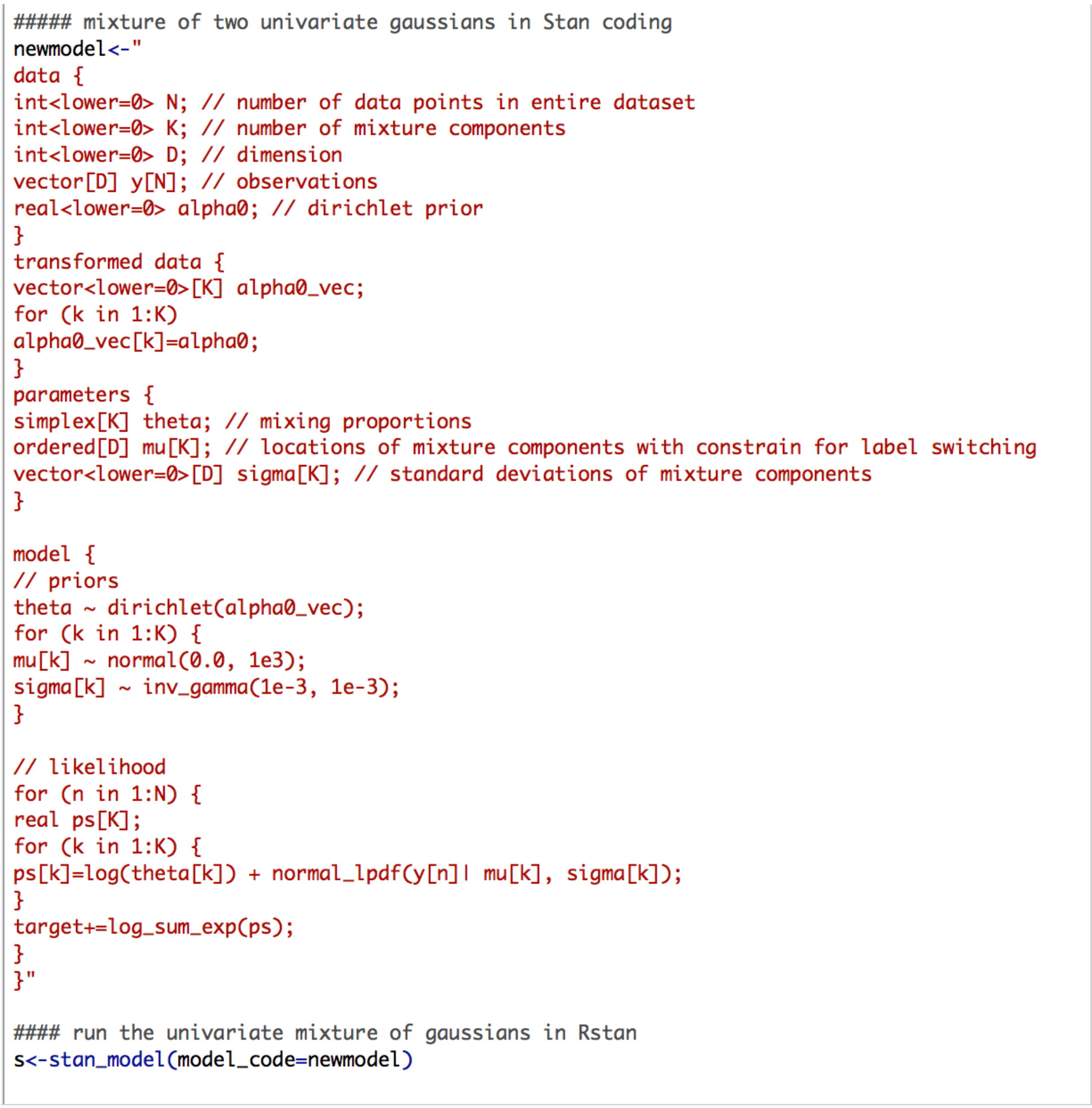
Stan Development Team (2015). Stan Modeling Language: User's Guide and Reference Manual. Version 2.8.0

Ormerod, J.T., Wand, M.P. (2010). Explaining Variational Approximations, *The American Statistician*, 64:2, 140-153

Yildirim, I. (2012). Gibbs sampling. http://www.mit.edu/~ilkery/papers/GibbsSampling.pdf

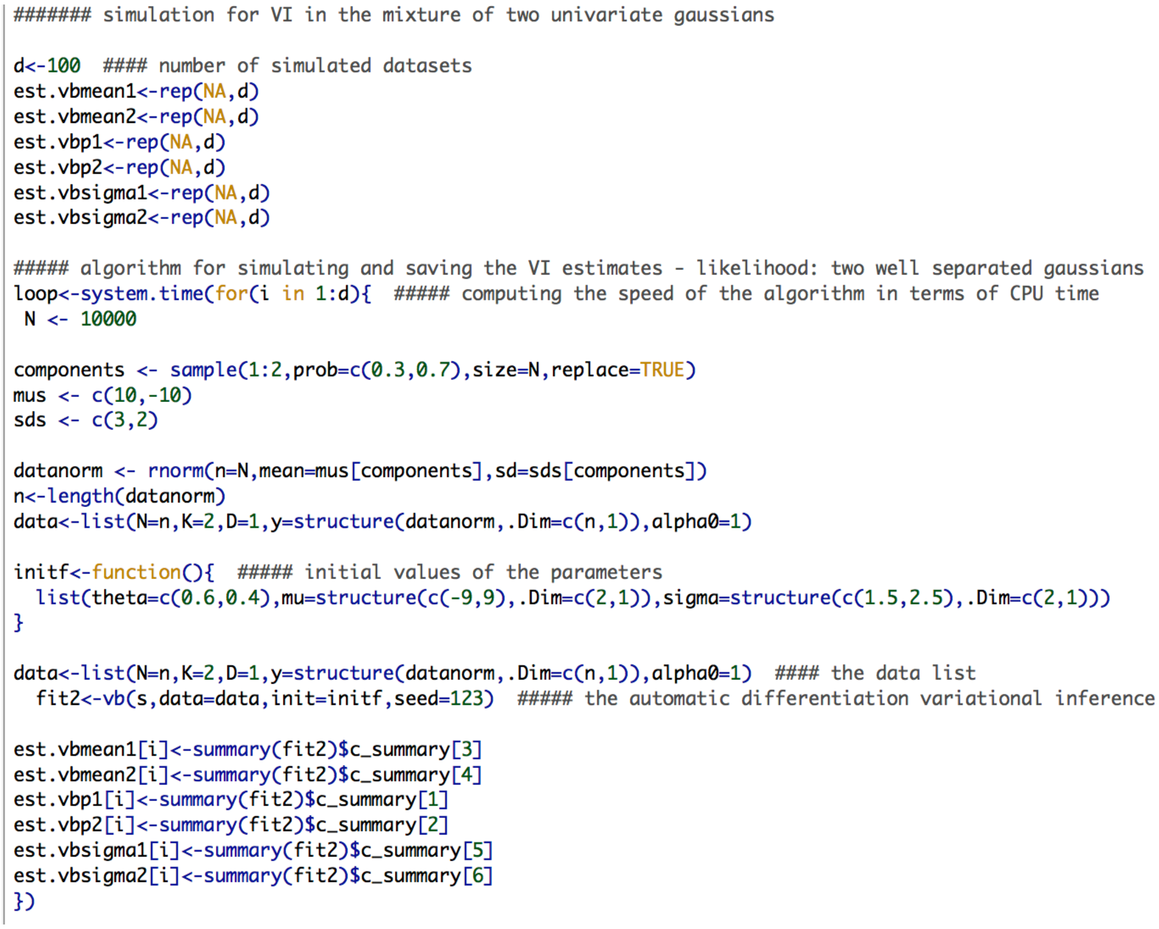
# Appendix

## R Code

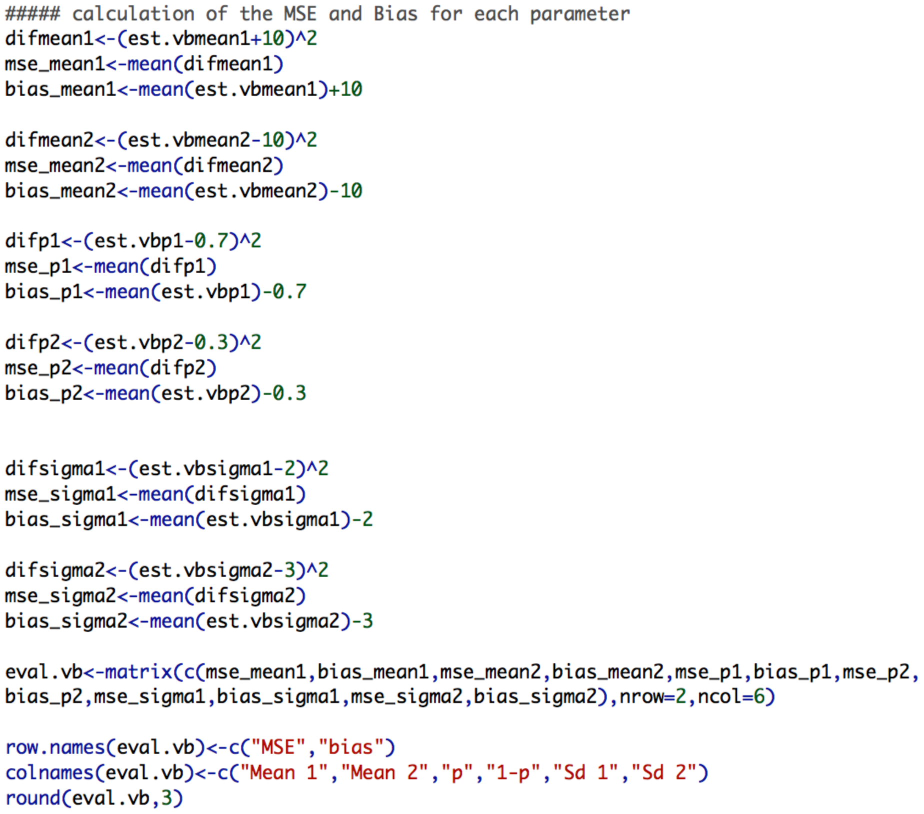


```
##### mixture of two univariate gaussians in Stan coding
newmodel<-"
data {
int<lower=0> N; // number of data points in entire dataset
int<lower=0> K; // number of mixture components
int<lower=0> D; // dimension
vector[D] y[N]; // observations
real<lower=0> alpha0; // dirichlet prior
}
transformed data {
vector<lower=0>[K] alpha0_vec;
for (k in 1:K)
alpha0_vec[k]=alpha0;
}
parameters {
simplex[K] theta; // mixing proportions
ordered[D] mu[K]; // locations of mixture components with constrain for label switching
vector<lower=0>[D] sigma[K]; // standard deviations of mixture components
}

model {
// priors
theta ~ dirichlet(alpha0_vec);
for (k in 1:K) {
mu[k] ~ normal(0.0, 1e3);
sigma[k] ~ inv_gamma(1e-3, 1e-3);
}

// likelihood
for (n in 1:N) {
real ps[K];
for (k in 1:K) {
ps[k]=log(theta[k]) + normal_lpdf(y[n]| mu[k], sigma[k]);
}
target+=log_sum_exp(ps);
}
}"

#### run the univariate mixture of gaussians in Rstan
s<-stan_model(model_code=newmodel)
```

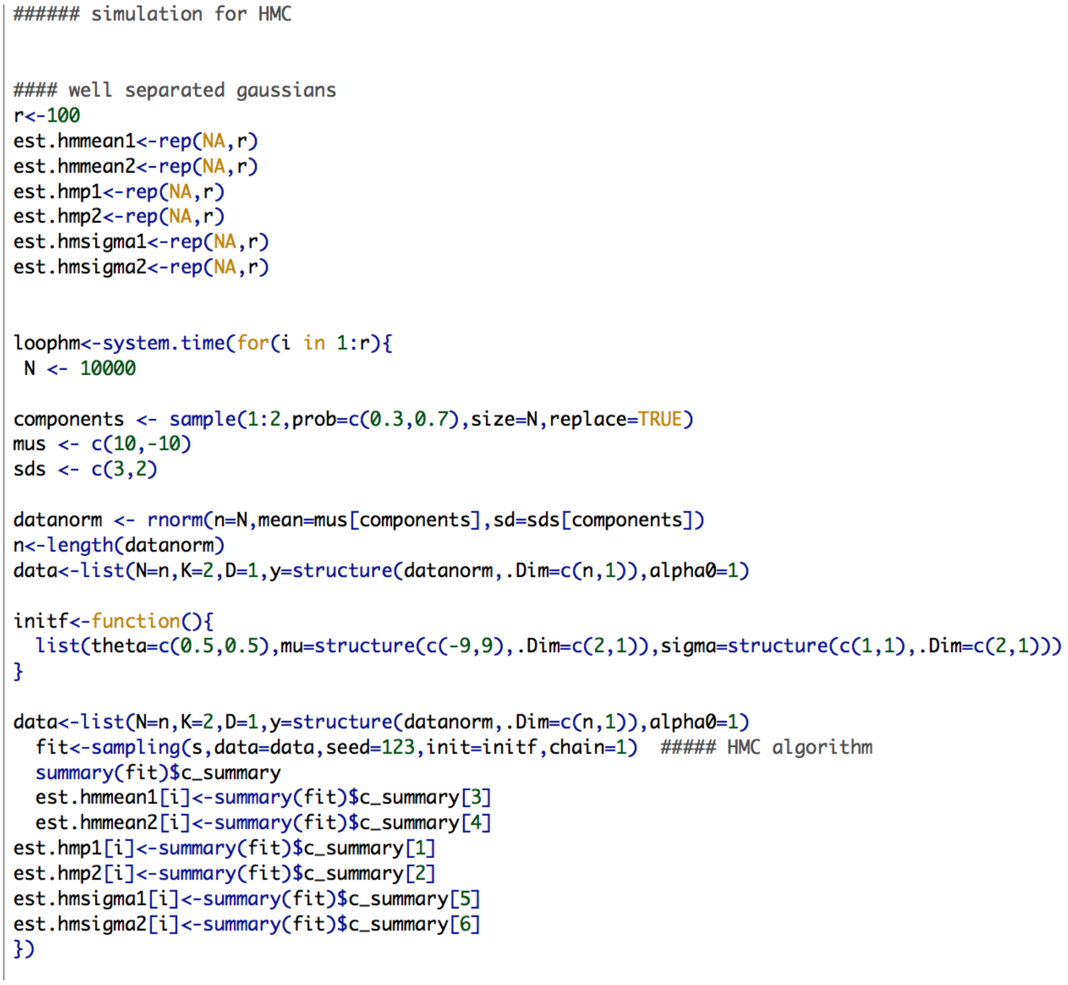
The simulation code for the two well separated Gaussians is provided for all the algorithms – VI, HMC & GS.



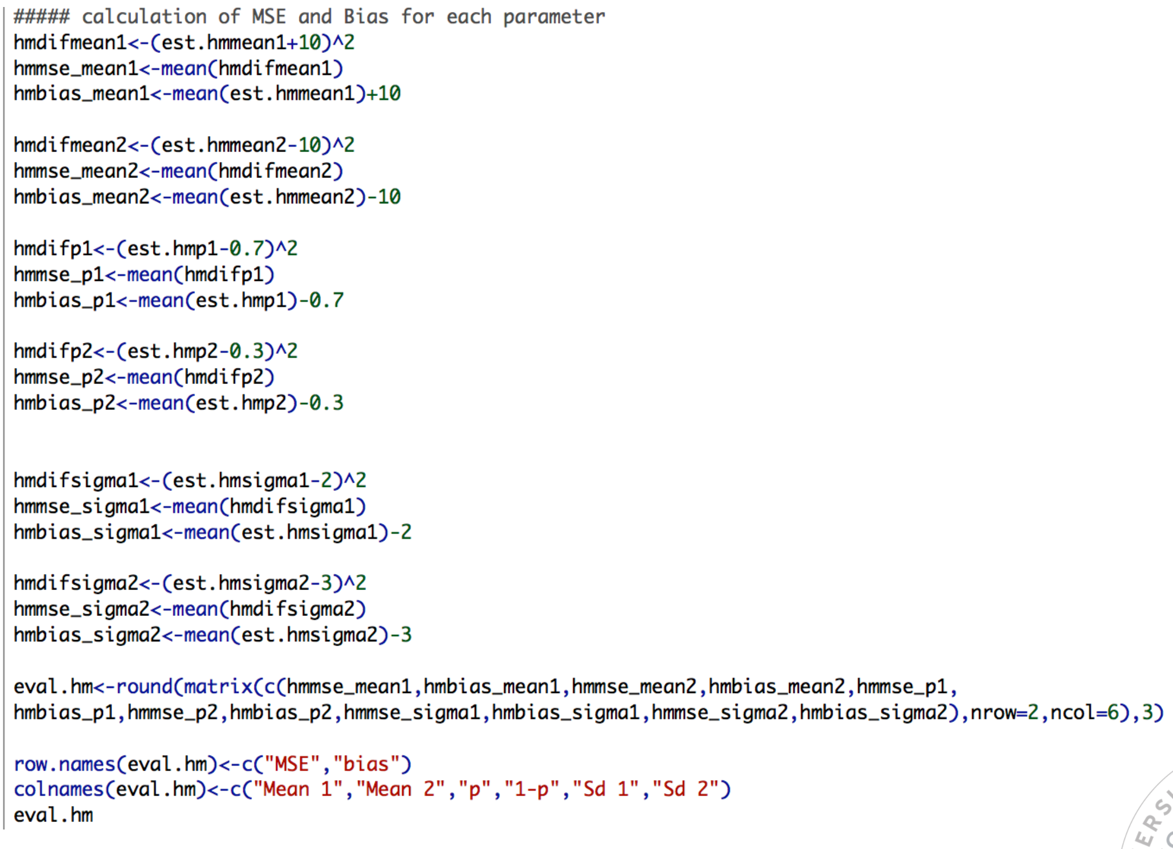
```
####### simulation for VI in the mixture of two univariate gaussians

d<-100  #### number of simulated datasets
est.vbmean1<-rep(NA,d)
est.vbmean2<-rep(NA,d)
est.vbp1<-rep(NA,d)
est.vbp2<-rep(NA,d)
est.vbsigma1<-rep(NA,d)
est.vbsigma2<-rep(NA,d)

##### algorithm for simulating and saving the VI estimates - likelihood: two well separated gaussians
loop<-system.time(for(i in 1:d){  ##### computing the speed of the algorithm in terms of CPU time
 N <- 10000

components <- sample(1:2,prob=c(0.3,0.7),size=N,replace=TRUE)
mus <- c(10,-10)
sds <- c(3,2)

datanorm <- rnorm(n=N,mean=mus[components],sd=sds[components])
n<-length(datanorm)
data<-list(N=n,K=2,D=1,y=structure(datanorm,.Dim=c(n,1)),alpha0=1)

initf<-function(){  ##### initial values of the parameters
  list(theta=c(0.6,0.4),mu=structure(c(-9,9),.Dim=c(2,1)),sigma=structure(c(1.5,2.5),.Dim=c(2,1)))
}

data<-list(N=n,K=2,D=1,y=structure(datanorm,.Dim=c(n,1)),alpha0=1)  #### the data list
  fit2<-vb(s,data=data,init=initf,seed=123)  ##### the automatic differentiation variational inference

est.vbmean1[i]<-summary(fit2)$c_summary[3]
est.vbmean2[i]<-summary(fit2)$c_summary[4]
est.vbp1[i]<-summary(fit2)$c_summary[1]
est.vbp2[i]<-summary(fit2)$c_summary[2]
est.vbsigma1[i]<-summary(fit2)$c_summary[5]
est.vbsigma2[i]<-summary(fit2)$c_summary[6]
})
```



```
##### calculation of the MSE and Bias for each parameter
difmean1<-(est.vbmean1+10)^2
mse_mean1<-mean(difmean1)
bias_mean1<-mean(est.vbmean1)+10

difmean2<-(est.vbmean2-10)^2
mse_mean2<-mean(difmean2)
bias_mean2<-mean(est.vbmean2)-10

difp1<-(est.vbp1-0.7)^2
mse_p1<-mean(difp1)
bias_p1<-mean(est.vbp1)-0.7

difp2<-(est.vbp2-0.3)^2
mse_p2<-mean(difp2)
bias_p2<-mean(est.vbp2)-0.3


difsigma1<-(est.vbsigma1-2)^2
mse_sigma1<-mean(difsigma1)
bias_sigma1<-mean(est.vbsigma1)-2

difsigma2<-(est.vbsigma2-3)^2
mse_sigma2<-mean(difsigma2)
bias_sigma2<-mean(est.vbsigma2)-3

eval.vb<-matrix(c(mse_mean1,bias_mean1,mse_mean2,bias_mean2,mse_p1,bias_p1,mse_p2,
bias_p2,mse_sigma1,bias_sigma1,mse_sigma2,bias_sigma2),nrow=2,ncol=6)

row.names(eval.vb)<-c("MSE","bias")
colnames(eval.vb)<-c("Mean 1","Mean 2","p","1-p","Sd 1","Sd 2")
round(eval.vb,3)
```

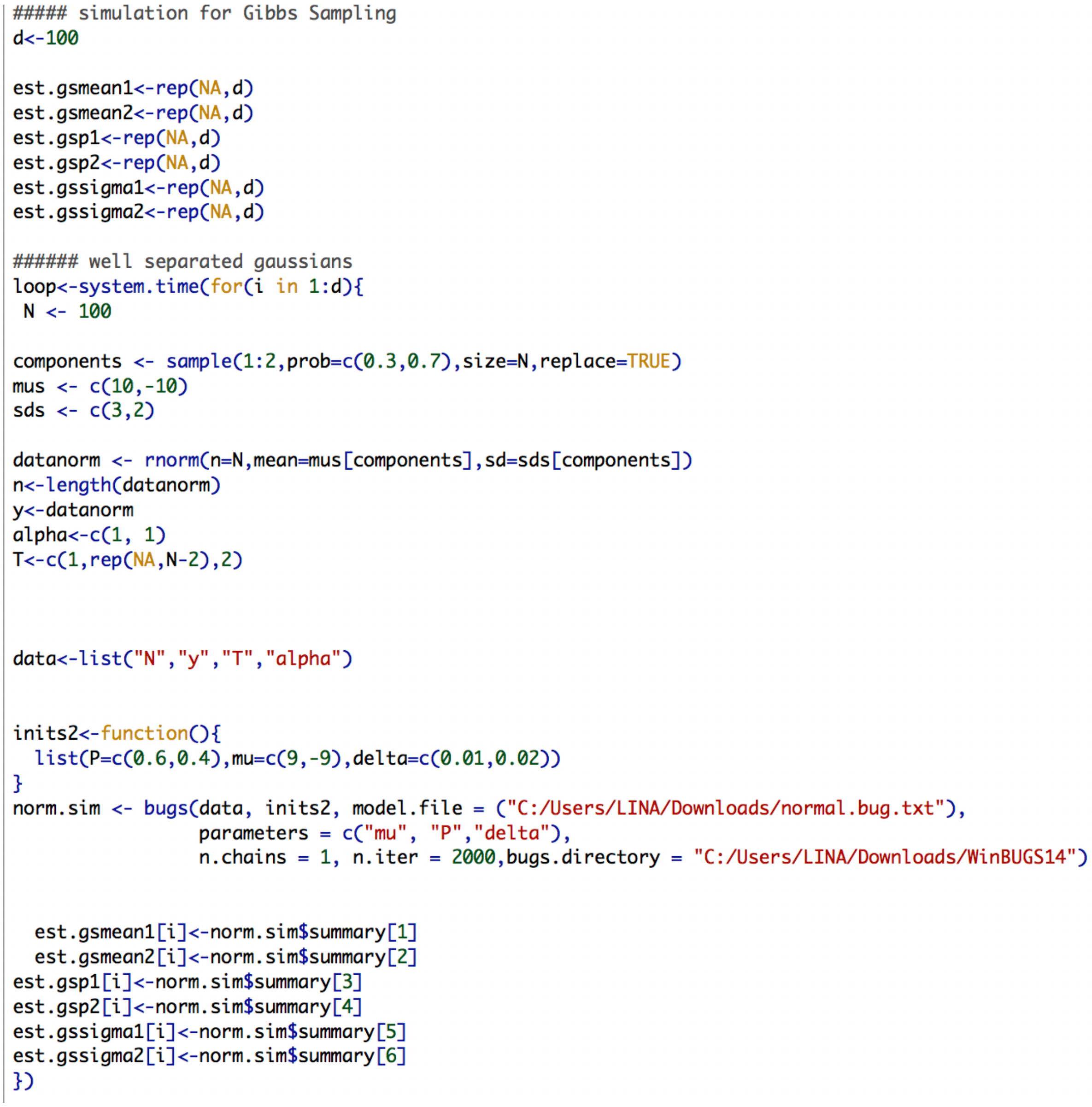


```
###### simulation for HMC


#### well separated gaussians
r<-100
est.hmmean1<-rep(NA,r)
est.hmmean2<-rep(NA,r)
est.hmp1<-rep(NA,r)
est.hmp2<-rep(NA,r)
est.hmsigma1<-rep(NA,r)
est.hmsigma2<-rep(NA,r)


loophm<-system.time(for(i in 1:r){
 N <- 10000

components <- sample(1:2,prob=c(0.3,0.7),size=N,replace=TRUE)
mus <- c(10,-10)
sds <- c(3,2)

datanorm <- rnorm(n=N,mean=mus[components],sd=sds[components])
n<-length(datanorm)
data<-list(N=n,K=2,D=1,y=structure(datanorm,.Dim=c(n,1)),alpha0=1)

initf<-function(){
  list(theta=c(0.5,0.5),mu=structure(c(-9,9),.Dim=c(2,1)),sigma=structure(c(1,1),.Dim=c(2,1)))
}

data<-list(N=n,K=2,D=1,y=structure(datanorm,.Dim=c(n,1)),alpha0=1)
  fit<-sampling(s,data=data,seed=123,init=initf,chain=1)  ##### HMC algorithm
  summary(fit)$c_summary
  est.hmmean1[i]<-summary(fit)$c_summary[3]
  est.hmmean2[i]<-summary(fit)$c_summary[4]
est.hmp1[i]<-summary(fit)$c_summary[1]
est.hmp2[i]<-summary(fit)$c_summary[2]
est.hmsigma1[i]<-summary(fit)$c_summary[5]
est.hmsigma2[i]<-summary(fit)$c_summary[6]
})
```

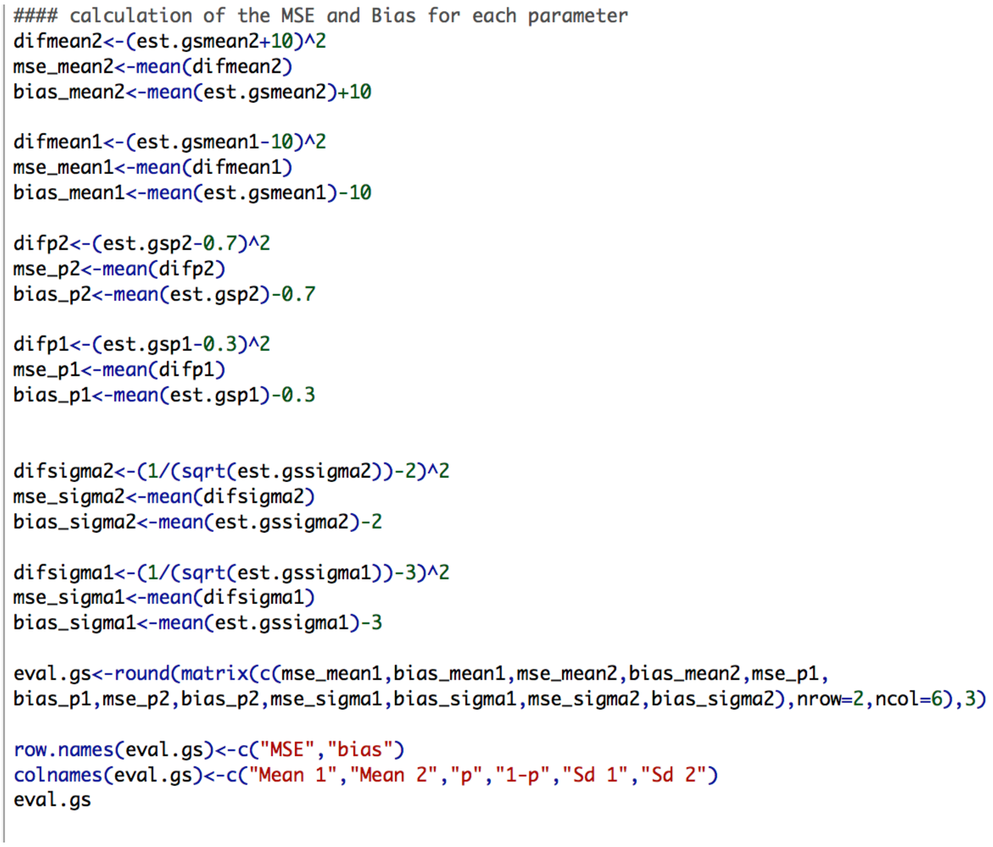


```
##### calculation of MSE and Bias for each parameter
hmdifmean1<-(est.hmmean1+10)^2
hmmse_mean1<-mean(hmdifmean1)
hmbias_mean1<-mean(est.hmmean1)+10

hmdifmean2<-(est.hmmean2-10)^2
hmmse_mean2<-mean(hmdifmean2)
hmbias_mean2<-mean(est.hmmean2)-10

hmdifp1<-(est.hmp1-0.7)^2
hmmse_p1<-mean(hmdifp1)
hmbias_p1<-mean(est.hmp1)-0.7

hmdifp2<-(est.hmp2-0.3)^2
hmmse_p2<-mean(hmdifp2)
hmbias_p2<-mean(est.hmp2)-0.3


hmdifsigma1<-(est.hmsigma1-2)^2
hmmse_sigma1<-mean(hmdifsigma1)
hmbias_sigma1<-mean(est.hmsigma1)-2

hmdifsigma2<-(est.hmsigma2-3)^2
hmmse_sigma2<-mean(hmdifsigma2)
hmbias_sigma2<-mean(est.hmsigma2)-3

eval.hm<-round(matrix(c(hmmse_mean1,hmbias_mean1,hmmse_mean2,hmbias_mean2,hmmse_p1,
hmbias_p1,hmmse_p2,hmbias_p2,hmmse_sigma1,hmbias_sigma1,hmmse_sigma2,hmbias_sigma2),nrow=2,ncol=6),3)

row.names(eval.hm)<-c("MSE","bias")
colnames(eval.hm)<-c("Mean 1","Mean 2","p","1-p","Sd 1","Sd 2")
eval.hm
```

```
###### Gaussian univariate mixture model in OpenBugs – saved as “normal.bug.txt”

model
   {
      for( i in 1 : N ) {
         y[i] ~ dnorm(mu[i], tau[i])
         mu[i] <- lambda[T[i]]
         T[i] ~ dcat(P[])
      }
      P[1:2] ~ ddirch(alpha[])
      theta ~ dnorm(0.0, 1.0E-6)I(0.0, )
      lambda[2] <- lambda[1] + theta
      lambda[1] ~ dnorm(0.0, 1.0E-6)
      tau[1] ~ dgamma(0.001, 0.001)
      tau[2] ~ dgamma(0.001, 0.001)
sigma[1] <- 1 / sqrt(tau[1])
sigma[2] <- 1 / sqrt(tau[2])
   }
```



```
##### simulation for Gibbs Sampling
d<-100

est.gsmean1<-rep(NA,d)
est.gsmean2<-rep(NA,d)
est.gsp1<-rep(NA,d)
est.gsp2<-rep(NA,d)
est.gssigma1<-rep(NA,d)
est.gssigma2<-rep(NA,d)

###### well separated gaussians
loop<-system.time(for(i in 1:d){
 N <- 100

components <- sample(1:2,prob=c(0.3,0.7),size=N,replace=TRUE)
mus <- c(10,-10)
sds <- c(3,2)

datanorm <- rnorm(n=N,mean=mus[components],sd=sds[components])
n<-length(datanorm)
y<-datanorm
alpha<-c(1, 1)
T<-c(1,rep(NA,N-2),2)



data<-list("N","y","T","alpha")


inits2<-function(){
  list(P=c(0.6,0.4),mu=c(9,-9),delta=c(0.01,0.02))
}
norm.sim <- bugs(data, inits2, model.file = ("C:/Users/LINA/Downloads/normal.bug.txt"),
                 parameters = c("mu", "P","delta"),
                 n.chains = 1, n.iter = 2000,bugs.directory = "C:/Users/LINA/Downloads/WinBUGS14")


  est.gsmean1[i]<-norm.sim$summary[1]
  est.gsmean2[i]<-norm.sim$summary[2]
est.gsp1[i]<-norm.sim$summary[3]
est.gsp2[i]<-norm.sim$summary[4]
est.gssigma1[i]<-norm.sim$summary[5]
est.gssigma2[i]<-norm.sim$summary[6]
})
```



```
#### calculation of the MSE and Bias for each parameter
difmean2<-(est.gsmean2+10)^2
mse_mean2<-mean(difmean2)
bias_mean2<-mean(est.gsmean2)+10

difmean1<-(est.gsmean1-10)^2
mse_mean1<-mean(difmean1)
bias_mean1<-mean(est.gsmean1)-10

difp2<-(est.gsp2-0.7)^2
mse_p2<-mean(difp2)
bias_p2<-mean(est.gsp2)-0.7

difp1<-(est.gsp1-0.3)^2
mse_p1<-mean(difp1)
bias_p1<-mean(est.gsp1)-0.3


difsigma2<-(1/(sqrt(est.gssigma2))-2)^2
mse_sigma2<-mean(difsigma2)
bias_sigma2<-mean(est.gssigma2)-2

difsigma1<-(1/(sqrt(est.gssigma1))-3)^2
mse_sigma1<-mean(difsigma1)
bias_sigma1<-mean(est.gssigma1)-3

eval.gs<-round(matrix(c(mse_mean1,bias_mean1,mse_mean2,bias_mean2,mse_p1,
bias_p1,mse_p2,bias_p2,mse_sigma1,bias_sigma1,mse_sigma2,bias_sigma2),nrow=2,ncol=6),3)

row.names(eval.gs)<-c("MSE","bias")
colnames(eval.gs)<-c("Mean 1","Mean 2","p","1-p","Sd 1","Sd 2")
eval.gs
```