DAVID MOLONEY

CANDIDATE 1012918

ST HUGH'S COLLEGE, UNIVERSITY OF OXFORD

SEPTEMBER 2017

CLUSTERING LEUKEMIA PATIENTS ACCORDING TO THEIR APPROXIMATE POSTERIOR PARAMETER DISTRIBUTION OF A DYNAMICAL SYSTEM

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science in Applied Statistics

SUMMARY

Chronic myeloid leukemia is a cancer of the white blood cells. Various mathematical models exist that model the dynamics of the disease progression, one of which was presented and analysed as part of this work.

Given data that described the progression of chronic myeloid leukemia over time for 69 newly diagnosed patients, this project solved an inverse problem that modeled the observed data. An initial value problem is presented that defines the inverse problem, and a Bayesian approach was used to find a solution. Unlike the previous literature, which used approximate Bayesian computation, this project implemented a Markov chain Monte Carlo (MCMC) algorithm through statistical programming language Stan.

In a standard Bayesian approach, the solution to the inverse problem for each patient was taken to be the posterior distribution of the multidimensional parameter that solved the initial value problem.

Patients were then compared by estimating the mean Euclidean distance between samples of the parameter of interest from their corresponding posterior distributions. The similarities between patients were visualised using a multidimensional scaling projection. Finally, the patients were separated into four clusters, using complete linkage hierarchical clustering, according to differences between their posterior distributions. The clusters largely separated the patients into groups in which disease progression was similar.

ACKNOWLEDGEMENTS

I would firstly like to thank my supervisor, Professor Geoff Nicholls, for his continued guidance throughout the project and for the countless suggestions he made over the past few months.

I would like to thank all staff and students in the department of statistics for their work and counsel over the past year.

Finally, I would like to express my special thanks to Julianna, my entire family, and all of my friends for the endless guidance and support over the years. I would not be where I am without you.

CONTENTS

1 Introduction **1**

1.1 Inverse Problems 1

1.2 Background Biology 1

1.3 Summary of Relevant Literature 3

1.4 Mathematical Model 4

1.5 Observation Model 6

2 Prior Elicitation **7**

2.1 Prior 8

2.2 Prior Suitability 10

3 Exploratory Data Analysis **12**

4 Model Fitting **13**

4.1 Hamiltonian Monte Carlo 13

4.2 Posterior MCMC Sampling 15

5 MCMC Diagnostics **16**

5.1 Convergence 16

5.2 Solution Suitability 19

5.3 Maximum a Posteriori Estimation 20

6 Patient Analysis **23**

6.1 Linear Discriminant Analysis 23

6.2 Patient Clustering 24

6.3 Dissimilarity Between Patients 25

6.4 Multidimensional Scaling 28

6.5 Hierarchical Clustering 30

7 Conclusions **36**

7.1 Solving the Inverse Problem 36

7.2 Patient Clustering 36

8 Appendix **40**

8.1 R Code 43

8.2 Stan Code 58

1 INTRODUCTION

1.1 Inverse Problems

Given a set of observations $\mathfrak{Y}$, parameters $\mathfrak{X}$, and a mapping

$$y = f(x) + \epsilon \qquad \text{for } x \in \mathfrak{X} \text{ and } y \in \mathfrak{Y}, \tag{1}$$

with unknown noise ϵ, the function f approximates observations y through model parameters x. The *forward problem* aims to predict observations y, given model parameters x.

The *inverse problem*, on the other hand, is considered the "inverse" to the forward problem. It begins with given data or observations y, and aims to estimate the model parameters x that best describe the data [1]. Conceptually, the inverse problem can be viewed as using data to infer model parameters.

$$\textit{Data} \longrightarrow \textit{Model Parameters} \tag{2}$$

By considering the pair (x, y) as random variables, the inverse problem can be "solved" using a Bayesian approach. The "solution" to the inverse problem can be defined as the posterior distribution of the model parameters x given the observations y, denoted $x|y$.

This project focused on solving an inverse problem. A system of ordinary differential equations will be defined in Section 1.4 that, given a set of model parameters, can be solved to estimate the BCR-ABL transcript level in the blood of patients undergoing treatment for chronic myeloid leukemia. Given BCR-ABL patient transcript data [2], the aim was to "solve" the inverse problem by estimating posterior distributions of the model parameters using Markov chain Monte Carlo techniques.

1.2 Background Biology

Haematopoiesis is the intricate process that allows for the formation of blood cellular components. Through this process, all blood cell components are derived from haematopoietic stem cells (HSCs) found in the bone marrow. Firstly, HSCs give rise to (myeloid and lymphoid) progenitor cells, which in turn, yield the wide range of blood cell types through further differentiation steps [3].

However, the healthy haematopoietic process can be disrupted to form deficient cells. There is strong evidence to suggest that this disease process will produce deficient cells in parallel to the healthy haematopoietic process. Therefore, a branch of the haematopoietic system will produce healthy blood cells normally while another branch will simultaneously produce deficient cells [4].

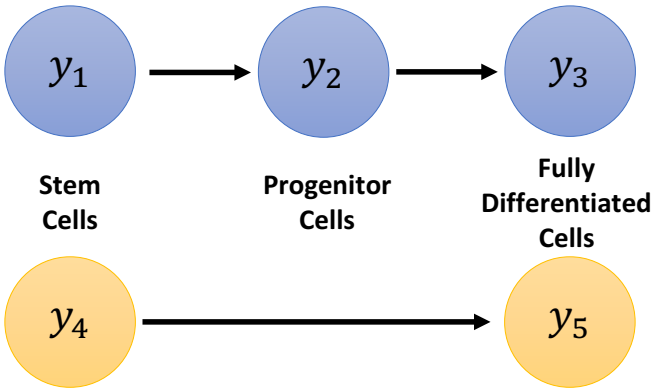


Figure 1: Haematopoietic process in the cell niche with the healthy and leukemic cells in blue and yellow respectively. The cell labels are defined in 3, and arrows represent transitions between cell types.

Interactions between healthy and diseased cell types can affect the disease progression. These interactions are a result of shared dependency on nutrients, cofactors, and physiological conditions that maintain stem cell properties. These environmental factors, which determine the fate of the stem cell, are known as the stem cell niche [5].

In the case of chronic myeloid leukemia (CML), a leukemic haematopoietic process produces unhealthy leukemic differentiated blood cells. This haematopoietic process begins with leukemic stem cells (LSCs). As mentioned, the leukemic haematopoietic process will produce differentiated leukemic blood cells in parallel to the healthy haematopoietic system deriving from HSCs. Both of these processes will interact due to the common influence of their niche, and therefore, the dynamics of both systems can be described as competitive interactions under common conditions due to the niche.

1.3 Summary of Relevant Literature

As was mentioned in Section 1.1, the primary focus of this project was to solve an inverse problem using a Bayesian approach. By treating the model parameters as random variables, the Bayesian "solution" to the inverse problem, as given by Bayes rule, is the posterior probability distribution of the model parameters given the observed data. This is a common method to solving the inverse problem with a more in-depth description found in [6]. More specifically, an overview of the Bayesian approach to inverse problems with applications in differential equations is found in [7], in which, algorithmic methods in Markov chain Monte Carlo (MCMC), filtering, and variational techniques are discussed.

A system of ordinary differential equations will be defined in Section 1.4. These differential equations, together with initial conditions, form an initial value problem which models the haematopoietic process as described in Section 1.2. Given noisy observation data, estimation of the parameters of the initial value problem defines the inverse problem. Therefore, literature that apply a Bayesian approach to solving inverse problems in differential equations are of particular relevance. There are many different approaches to such inverse problems. A family of collocation methods are outlined in [8] in which collocation is used to solve a system of differential equations while the collocation points are compared to the experimental data in the inverse problem. Another example of an approach to inverse problems in differential equations includes a combined data smoothing and estimation method developed in [9].

This report draws on previous literature by MacLean [10] which implemented an approximate Bayesian computation (ABC) scheme to perform parameter inference on the initial value problem modeling the haematopoietic process. ABC is a simple technique which estimates the model parameters that produce predicted data that is "similar" to the observed data in the inverse problem. A detailed description of the use of ABC in parameter estimation is given in [11].

This project built upon the work by MacLean in a number of ways. Instead of ABC, this project instead implements a MCMC technique to perform parameter inference on the initial value problem. MCMC was chosen as, unlike the aforementioned methods, it produces samples distributed exactly according to the posterior distribution. Although it can be a computationally expensive approach, it is a well optimised and researched method. It is a very common approach to inverse problems. Some examples of the use of MCMC to solve inverse problems involving differential equations can be found in [12, 13, 14].

This project also differed from the work by MacLean in the treatment of the observational data. As will be described in further sections of the report, observational data existed for 69 leukemia patients that either went into remission or relapse. In the literature by MacLean, the patients that went into remission were combined to form a single overlying remission patient. The inverse problem was then solved by estimating the posterior parameter distributions for both this overlying remission patient and for one relapse patient using ABC. This project extended on the previous work by considering each patient separately, and the posterior parameter distributions of each were estimated using MCMC. Unlike in the work by MacLean, this project then clustered the patients according to their posterior parameter distributions.

1.4 Mathematical Model

As previously mentioned in Section 1.2, the dynamics of the healthy and leukemic haematopoietic processes can be described as competitive interactions in their common niche. This hands itself to using population dynamic models defined by ordinary differential equations (ODEs). Many such models have been developed and examined in previous studies [15]. In this project, the following simple model using niche competition was implemented.

For this model, the following five competing populations of different blood cell components were considered.

$$\begin{aligned} y_1(t) &- \text{Healthy haematopoietic stem cells (HSCs)} \\ y_2(t) &- \text{Healthy progenitor cells} \\ y_3(t) &- \text{Fully differentiated healthy cells} \\ y_4(t) &- \text{Leukemic stem cells (LSCs)} \\ y_5(t) &- \text{Fully differentiated leukemic cells} \end{aligned} \tag{3}$$

Let $z(t) = y_1(t) + y_2(t) + y_3(t) + y_4(t) + y_5(t)$ be the total cell population at time t, and let $\dot{y}_i = dy_i/dt$ be the standard notation for the time derivative of y_i. The dynamics of the

populations defined above can then be given by the following system of ODEs.

$$\begin{aligned}
\dot{y_1} &= \theta_1 y_1\big(k-z\big) - \theta_2 y_1 \\
\dot{y_2} &= \theta_2 y_1 + \theta_3 y_2\big(k-z\big) - \theta_4 y_2 \\
\dot{y_3} &= \theta_4 y_2 - \theta_5 y_3 \\
\dot{y_4} &= \theta_6 y_4\big(k-z\big) - \theta_7 y_4 \\
\dot{y_5} &= \theta_7 y_4 - \theta_8 y_5
\end{aligned} \tag{4}$$

In the above system of ODEs, k is defined as the carrying capacity of the niche (i.e. the total number of cells that can exist within the niche). In this analysis, the carrying capacity was defined as $k = 1$. Therefore, each population, as defined in 3, represents the proportion of the niche taken by that population at time t. The dynamical model parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_8)$ are defined as follows:

$$\begin{array}{lcl}
\theta_1 & - & \text{Rate of self-renewal of} y_1 \\
\theta_2 & - & \text{Rate of transition from} y_1 \text{to} y_2 \\
\theta_3 & - & \text{Rate of self-renewal of} y_2 \\
\theta_4 & - & \text{Rate of transition from} y_2 \text{to} y_3 \\
\theta_5 & - & \text{Rate of disappearance of} y_3 \\
\theta_6 & - & \text{Rate of self-renewal of} y_4 \\
\theta_7 & - & \text{Rate of transition from} y_4 \text{to} y_5 \\
\theta_8 & - & \text{Rate of disappearance of} y_5
\end{array} \tag{5}$$

where each rate represents the proportional change per unit time as described above. The above defined rates will introduce boundary constraints on the dynamical model parameters. These will be discussed in Section 2.

By considering the sign of $\dot{y_1}, \dot{y_2}, \dot{y_3}, \dot{y_4}, \dot{y_5}$, and $\dot{z}$, it can be easily shown that the above system enforces the correct bounds on each cell population and on the total population within the niche. That is, each of $y_1(t), y_2(t), y_3(t), y_4(t), y_5(t)$, and $z(t)$ are restricted to the interval $[0,1]$ for all $t > 0$ if they are each in the interval at time $t = 0$.

The above system can be written as the following initial value problem (IVP)

$$\begin{array}{lcl}
\dot{\boldsymbol{y}}(t) & = & \boldsymbol{f}(t, \boldsymbol{y}(t), \boldsymbol{\theta}) \\
\boldsymbol{y}(0) & = & \boldsymbol{y^{(0)}}
\end{array} \tag{6}$$

with populations $\boldsymbol{y}(t) = \big(y_1(t), y_2(t), \ldots, y_5(t)\big)$, ODE parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_8)$, and initial values $\boldsymbol{y^{(0)}} = \big(y_1^{(0)}, y_2^{(0)}, \ldots, y_5^{(0)} \big)$. The IVP can easily be solved for populations $\boldsymbol{y}(t)$ with given values for the ODE parameters $\boldsymbol{\theta}$ and initial values $\boldsymbol{y^{(0)}}$ using standard numerical methods.

1.5 Observation Model

The first part of this project focused on using the mathematical model defined in Section 1.4 to model the "cancer levels" in leukemia patients in data provided by Prof. Ingo Roeder for use in [10]. This dataset consisted of information from 69 leukemia patients undergoing tyrosine kinase inhibitor (TKI) therapy who typically go into remission or relapse. For each patient, there existed a series measuring the "cancer levels" in the blood over time. The "cancer level" at time point j for a given Patient i is quantified by the observed level of BCR-ABL protein in the blood at that time and is labeled in this report as R_{ij}, where $i \in \{1, \ldots, 69\}$ and $j \in \{1, \ldots, n_i\}$. Decreasing BCR-ABL levels over time corresponds to remission, and a spike in BCR-ABL corresponds to relapse. In relation to the ODEs in 4, and given parameters and initial values $\boldsymbol{\theta}$ and $\boldsymbol{y^{(0)}}$ respectively, the predicted BCR-ABL level at time t, $R(\ t \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}\ )$, can be defined by the following relation [16]

$$R(\ t \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}\ ) \quad = \quad \frac{y_5(t)}{y_5(t) + 2y_3(t)} \tag{7}$$

where $y_5(t)$ and $y_3(t)$ are solutions to the IVP 6.

This gave rise to the following observation model for a given Patient $i \in \{1, \ldots, 69\}$ and each $j \in \{1, \ldots, n_i\}$

$$R_{ij} \quad = \quad R\big(\ t_{ij} \mid \boldsymbol{\theta^{(i)}}, \boldsymbol{y^{(0,i)}}\ \big) \ + \ \epsilon_{ij} \tag{8}$$

with noise $\epsilon_{ij} \sim N(0, v^{(i)})$, ODE parameters $\boldsymbol{\theta^{(i)}}$, and initial values $\boldsymbol{y^{(0,i)}}$.

Given the Observation Model 8 for Patient $i \in \{1, \ldots, 69\}$, Bayesian parameter inference was then carried out in order to determine the posterior distribution of the following 14-dimensional parameter

$$\begin{aligned} \boldsymbol{\lambda^{(i)}} \quad &= \quad \big(\theta_1^{(i)}, \ldots, \theta_8^{(i)}, y_1^{(0,i)}, \ldots, y_5^{(0,i)}, v^{(i)}\big) \\ &= \quad (\boldsymbol{\theta^{(i)}}, \boldsymbol{y^{(0,i)}}, v^{(i)}). \end{aligned} \tag{9}$$

For the remainder of this section, the i notation is dropped for convenience, and the posterior

is described for parameter $\boldsymbol{\lambda}$ for an arbitrary patient. Given observed BCR-ABL level data $D = \{R_1, \ldots, R_n\}$ the posterior is given by Bayes formula

$$p(\boldsymbol{\lambda}|D) \quad \propto \quad p(D|\boldsymbol{\lambda})\ \pi(\boldsymbol{\lambda}). \tag{10}$$

The data was independently normal with $R_j \sim N\big(\ R(t_j \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}), v\ \big)$ for $j \in \{1, \ldots, n\}$, which defined the likelihood

$$\begin{aligned} p(D|\boldsymbol{\lambda}) \quad &= \quad \prod_{j=1}^{n} N\big(\ R_j;\ R(t_j \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}), v\ \big) \\ &= \quad \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi v}} \exp\left(\frac{\big(R_j - R(t_j \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}})\big)^2}{2v} \right). \end{aligned} \tag{11}$$

The choice of a normal likelihood is a slight inaccuracy as it is defined on $(-\infty, \infty)$, whereas $R(t \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}})$ is bounded on $[0,1]$. A log-normal likelihood was considered to partially account for this, however, there were issues with convergence in the Markov chain Monte Carlo algorithm for values near the boundary at 0. Given that there was no real statistical advantage in using the log-normal likelihood, a normal distribution was used.

The prior can be written as the independent product of the priors for $\boldsymbol{\theta}$, $\boldsymbol{y^{(0)}}$, and v as follows

$$\pi(\boldsymbol{\lambda}) \quad = \quad \pi(\boldsymbol{\theta})\ \pi(\boldsymbol{y^{(0)}})\ \pi(v). \tag{12}$$

The individual priors in Equation 12 will be discussed further in Section 2.

2 PRIOR ELICITATION

As mentioned in Section 1.5, Bayesian parameter inference was carried out on the 14 dimensional parameter $\boldsymbol{\lambda} = \big(\theta_1, \ldots, \theta_8, y_1^{(0)}, \ldots, y_5^{(0)}, v\big) = (\boldsymbol{\theta}, \boldsymbol{y^{(0)}}, v)$. The first 13 dimensions were required to solve the Initial Value Problem 6, and the final dimension was the variance parameter v from the normal likelihood in Equation 11. The prior knowledge on these parameters was limited. However, the following considerations were made.

- The patients considered in the data were undergoing treatment for leukemia. Therefore, the initial BCR-ABL level for each patient, $R(\ 0 \mid \boldsymbol{y^{(0)}}\ )$, is expected to be large.
- Since the patients were undergoing treatment and typically go into remission or relapse,

it is expected that, for the majority of patients, the BCR-ABL level would initially decrease. That is, $R(\,\epsilon \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}\,) < R(\,0 \mid \boldsymbol{y^{(0)}}\,)$ for some small time $\epsilon > 0$.

- The ODE parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_8)$ should have a non-informative prior as they determine the dynamics of the model. Therefore, their distribution should allow for dynamics that could model the BCR-ABL level for patients that go into either relapse or remission.

  Since each ODE parameter θ_i for $i \in \{1, \ldots, 8\}$ are presented as rates in 5, each represents a proportion of that population to change state per unit time. Therefore, $\theta_i \geq 0$ to maintain the correct direction of transition as defined in 5, and $\theta_i \leq 1$ to prevent a larger proportion of the corresponding cell population to transition state than actually exists.

- The variance parameter υ should be reasonable given that the range of R is $[0,1]$. The support of the prior for υ should also be large enough to allow for uncertainty in the model.

2.1 Prior

In Section 1.5, the prior for $\boldsymbol{\lambda}$ was stated as the independent product of priors for $\boldsymbol{\theta}$, $\boldsymbol{y^{(0)}}$, and υ.

$$\pi(\boldsymbol{\lambda}) \quad = \quad \pi(\boldsymbol{\theta})\ \pi(\boldsymbol{y^{(0)}})\ \pi(\upsilon) \tag{13}$$

The three priors were considered separately, and they were carefully chosen to satisfy the considerations made above. The performance for each is discussed in Section 2.2.

- The first prior $\pi(\boldsymbol{\theta})$ considered the ODE parameters that define the dynamics of the model. Similar to [15], the prior for each θ_i with $i \in \{1, \ldots, 8\}$ was chosen to be uniform on [0,1]. This was simply chosen as it was necessary for the prior to be non-informative, and to have bounds on each parameter that satisfy the considerations made above. Therefore,

$$\begin{aligned} \pi(\boldsymbol{\theta}) \quad &= \quad \prod_{i=1}^{8}\ \pi(\theta_i) \\ &= \quad \prod_{i=1}^{8}\ U(\,\theta_i;\,0,1) \\ &= \quad 1. \end{aligned} \tag{14}$$

- For use as the variance in the normal likelihood, the prior for v was chosen to be a gamma distribution with shape and rate parameters $(\alpha, \beta) = (0.25, 5)$. The prior density can be seen in the appendix in Figure 19. This was simply chosen to produce reasonable values for variance v with a large enough support to allow for uncertainty. Therefore,

$$\begin{aligned} \pi(v) &= \Gamma(\ v;\ \alpha, \beta) \\ &= \frac{\beta^{\alpha}}{\Gamma(\alpha)} v^{\alpha-1} e^{-\beta v} \end{aligned} \tag{15}$$

  The median of v is approximately 0.0087 and is centered in the 95% confidence interval $C_{95} = [5 \times 10^{-8}, 3 \times 10^{-1}]$.

- The prior for the initial conditions $\boldsymbol{y^{(0)}}$ was taken to be a scaled Dirichlet distribution where

$$\sum_{i=1}^{5} y_i^{(0)} = z^{(0)},$$

  and $z^{(0)}$ is the total cell niche population at time $t = 0$. Therefore, by defining $\boldsymbol{y^{(0)}}$ in terms of the latent total cell population $z^{(0)}$ and scaled initial values $\tilde{\boldsymbol{y}}^{(0)}$

$$\boldsymbol{y^{(0)}} = z^{(0)}\ \tilde{\boldsymbol{y}}^{(0)}, \tag{16}$$

  the prior for $\boldsymbol{y^{(0)}}$ can be formed with the standard change of variables jacobian $\boldsymbol{J}$ as

$$\begin{aligned} \pi(\boldsymbol{y^{(0)}}) &= \pi(z^{(0)},\ \tilde{\boldsymbol{y}}^{(0)})\boldsymbol{J} \\ &= \pi(z^{(0)})\ \pi(\tilde{\boldsymbol{y}}^{(0)})\boldsymbol{J} \quad . \end{aligned} \tag{17}$$

  The total niche population proportion at the initial time $z^{(0)}$ was taken to have a uniform prior on [0,1]. A uniform distribution was chosen in order to be non-informative as there was no prior knowledge for the initial total proportion. The uniform prior was defined on [0,1] as the total niche proportion is clearly positive and bounded by the carrying capacity at 1. The prior for the scaled initial conditions $\tilde{\boldsymbol{y}}^{(0)}$ was taken to have a Dirichlet distribution with parameter vector $\boldsymbol{\alpha} = (0.9, 0.9, 0.4, 0.9, 1.5)$. This was chosen to satisfy the considerations made previously. The jacobian can be calculated in a standard way, but for simplicity, the joint prior on $z^{(0)}$ and $\tilde{\boldsymbol{y}}^{(0)}$ is given as

$$\begin{aligned} \pi(z^{(0)},\ \tilde{\boldsymbol{y}}^{(0)}) &= \pi(z^{(0)})\ \pi(\tilde{\boldsymbol{y}}^{(0)}) \\ &= \frac{\Gamma\left(\sum_{i=1}^{5} \alpha_i\right)}{\prod_{i=1}^{5} \Gamma(\alpha_i)} \prod_{i=1}^{5} \left(\tilde{y}_i^{(0)}\right)^{\alpha_i - 1} \end{aligned} \tag{18}$$

where $\tilde{y}_i^{(0)} = y_i^{(0)} / \sum_{i=1}^{5} y_i^{(0)}$.

The prior densities of these five initial condition parameters are shown in the appendix in Figure 19. The corresponding prior density for the initial BCR-ABL level $R(\ 0 \mid \boldsymbol{y^{(0)}}\ )$ is shown below in Figure 2.

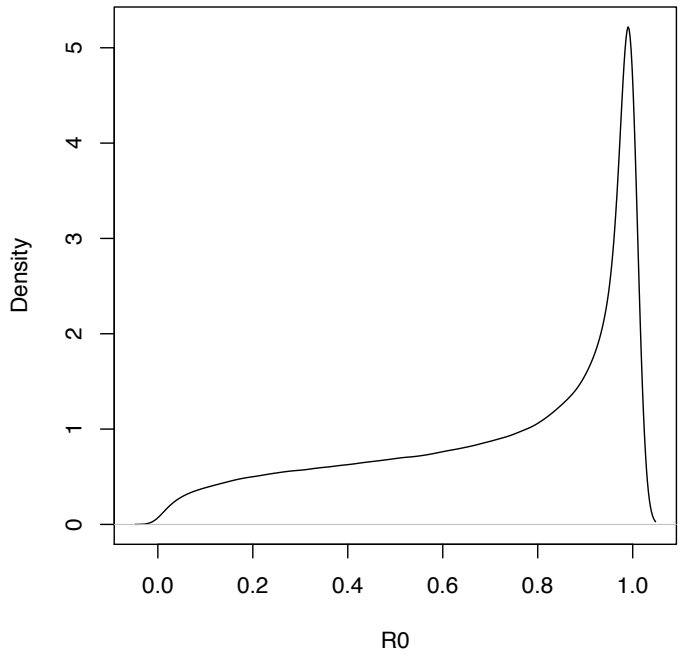


Figure 2: Prior density for the initial BCR-ABL level $R(\ 0 \mid \boldsymbol{y^{(0)}}\ )$.

2.2 Prior Suitability

As mentioned at the beginning of this section, there were a number of considerations made in this project when choosing a prior. These considerations were all satisfied individually as follows:

- As seen in Figure 2, the prior density for $R(\ 0 \mid \boldsymbol{y^{(0)}}\ )$ has a negative skew with mean approximately 0.71. This matched the prior knowledge that the initial BCR-ABL level $R(\ 0 \mid \boldsymbol{y^{(0)}}\ )$ is usually large.
- In order to test the prior knowledge that the value of $R(\ t \mid \boldsymbol{\theta}, \boldsymbol{y^{(0)}}\ )$ will initially decrease due to the patients undergoing treatment, 10000 values of $\boldsymbol{\lambda}$ were sampled from the prior.

In approximately 75.1% of cases, the corresponding BCR-ABL level initially decreased after time $t = 0$. This matched well with the assumed prior knowledge.

- Figure 3 below shows 50 sample BCR-ABL plots with parameters sampled from the prior. As previously mentioned, the BCR-ABL level quantifies the "cancer level" for the patient at any given time. As can clearly be seen, the individual samples result in trace solutions that could represent either remission (BCR-ABL tends to zero) or relapse (BCR-ABL decreases and then spikes). To test this, 10000 traces of BCR-ABL levels were produced by sampling values of $\boldsymbol{\lambda}$ from the prior. 59.1% of the sampled traces had a BCR-ABL level less than 0.05 after 65 months, indicating remission. This suggested that there was no large bias in the prior towards either clinical outcome.

- The gamma prior for the variance as defined in Equation 15 produced a median value of approximately 0.0087 centered in the 95% confidence interval $C_{95} = [5 \times 10^{-8}, 3 \times 10^{-1}]$. This suggested reasonable values for the variance v. This prior also has a large enough support to allow for uncertainty. The prior density for v can be seen in the appendix in Figure 19.

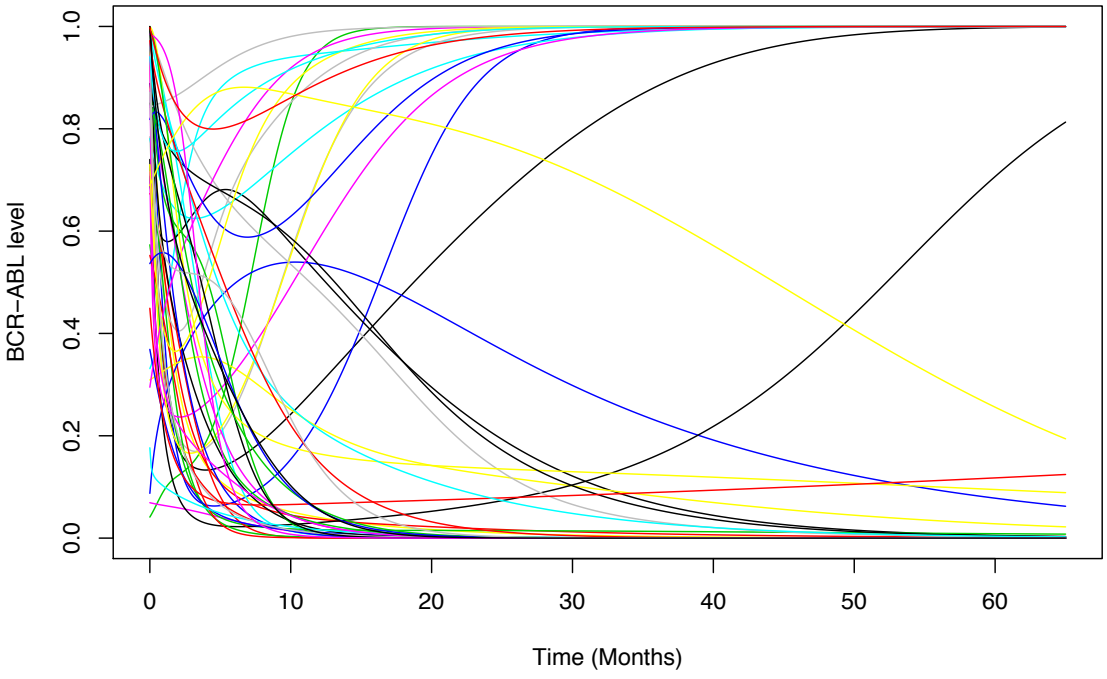


Figure 3: Plots of BCR-ABL level with time as solved by the system of ODEs with theta and ICs sampled from the respective priors.

3 EXPLORATORY DATA ANALYSIS

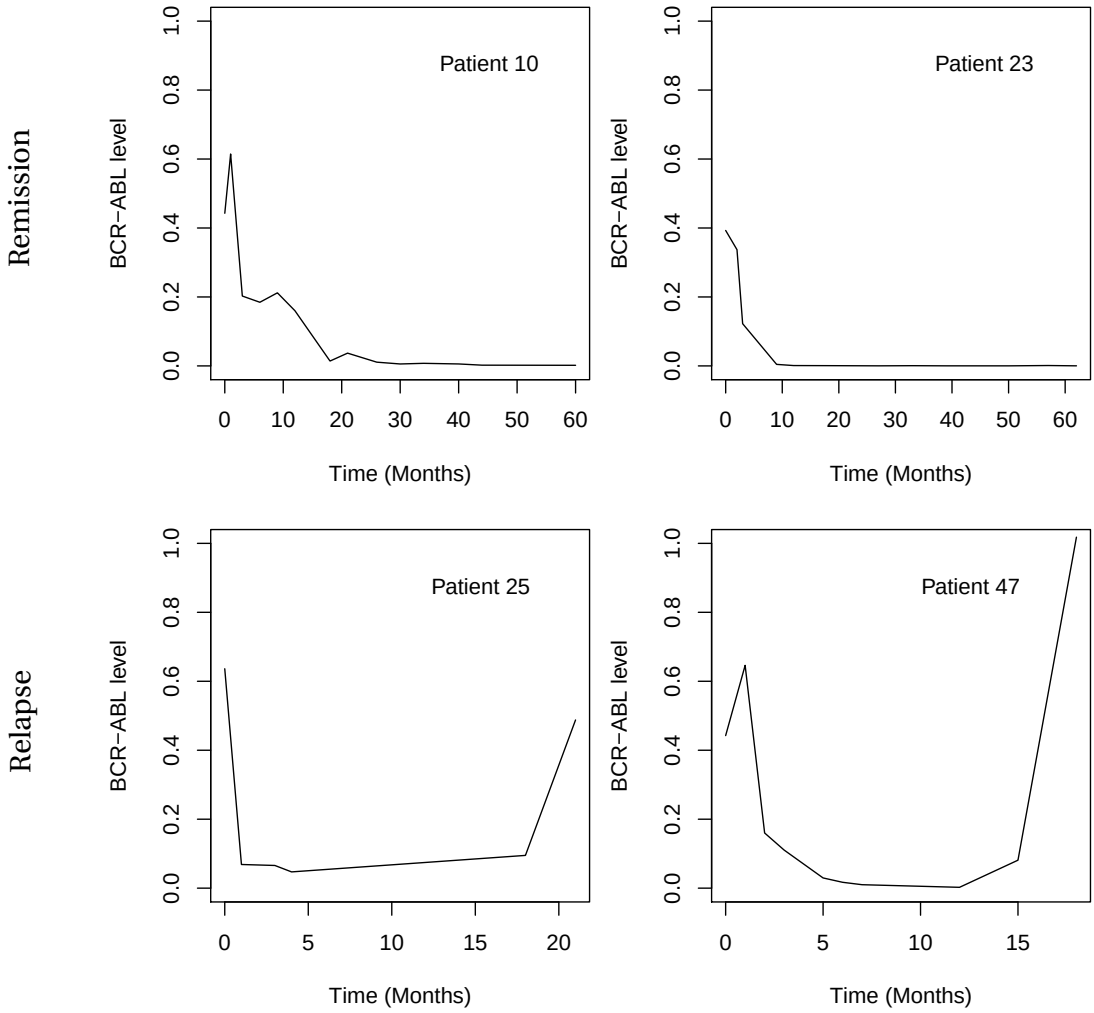


Figure 4: Plots of BCR-ABL level with time for remission and relapse cases in the data. Patients 10 and 23 respectively are plotted for remission. Below, Patients 25 and 47 respectively are plotted in the case of relapse.

The data used in this project was collected as part of a study to compare two different types of treatment for CML. The data consists of BCR-ABL transcript levels over time for 69 newly diagnosed CML patients receiving imatinib [2]. The vast majority of patients (57 out of 69) showed reducing BCR-ABL levels over time corresponding to remission.

Six patients had BCR-ABL transcript levels that failed to decrease over time. These patients were considered relapse patients. In the data, and in the remaining sections of this report,

these were the patients numbered 1, 7, 25, 26, 47, and 68.

The remaining six patients had varying behavior in the changing BCR-ABL transcript level. These patients were difficult to classify as either relapse or remission. In Figure 4, two examples of each remission and relapse are plotted.

In this project, Bayesian parameter inference was carried out for each patient individually with the exception of Patient 66. This particular case was removed as there was only one time measurement of the BCR-ABL level for this patient. Therefore define

$$I = \{1,2,\ldots,65,67,\ldots,69\} \tag{19}$$

to be the set of patients that were used for Bayesian parameter inference.

For Sections 4 and 5 in the report, Patients 10 and 25 are shown as examples for the remission and relapse cases respectively. All other cases were statistically similar.

4 MODEL FITTING

The posterior distribution of the 14 dimensional parameter $\boldsymbol{\lambda} = (\theta_1,\ldots,\theta_8, y_1^{(0)},\ldots,y_5^{(0)}, v)$ was estimated using Markov chain Monte Carlo (MCMC) implemented through statistical programming language Stan [17]. The MCMC algorithm and model fitting procedure used in this project will be discussed in Section 4.2, and the R and Stan code can be found in the appendix. Firstly, the Hamiltonian Monte Carlo algorithm, as used in Stan, will be described.

4.1 Hamiltonian Monte Carlo

Stan uses a Hamiltonian Monte Carlo (HMC) algorithm [18] implemented in R and compiled in C++. HMC is a gradient-based MCMC method that uses derivatives of the density function that is being sampled in order to generate efficient transitions between Markov chain samples. It is particularly useful for high dimensional sampling distributions where a random walk would require an extremely large number of warmup samples before convergence has occurred. Figure 5 shows a three dimensional surface of high posterior density. Posterior samples from a HMC algorithm are shown as green points in the plot. HMC provides a method for proposing samples close to the surface of high posterior density. As a result, this technique is far more

efficient than a simple random walk which would propose samples randomly in the high dimensional Euclidean space.

In this project, the goal is to draw 14 dimensional samples from posterior density $p(\boldsymbol{\lambda}|D)$. Since this is clearly a high dimensional example, HMC was a necessary choice. In this section, the HMC algorithm will be described with the aim of sampling parameter ϕ from density $p(\phi)$ for simplicity.

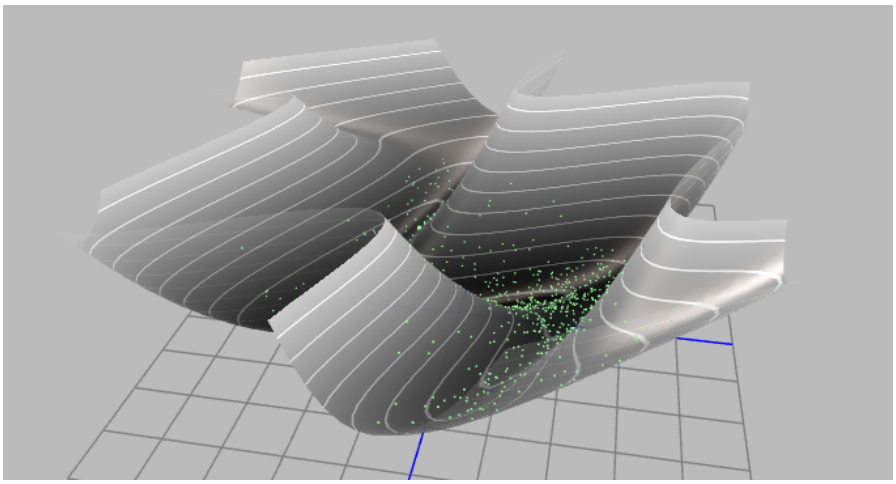


Figure 5: Visualisation of posterior samples (green) from a high dimensional surface solution as taken from *http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html*

HMC first introduces an auxiliary momentum variable ρ and draws from the joint density

$$p(\rho,\phi) \quad = \quad p(\rho|\phi)\ p(\phi) \quad .$$

In Stan, the momentum is drawn from a multivariate normal with 0 mean and covariance matrix Σ which is estimated during the warmup period.

The joint density $p(\rho,\phi)$ then defines a Hamiltonian

$$\begin{aligned} H(\rho,\phi) \quad &= \quad -\log p(\rho|\phi) \ - \ \log p(\phi) \\ &= \quad T(\rho|\phi) \ + \ V(\phi) \end{aligned}$$

where T and V are the kinetic and potential energy respectively.

Given a current parameter ϕ, a Markov chain transition is generated in two steps. First, a

value for ρ is sampled from the multivariate normal. Next, the joint system (ρ, ϕ) is evolved using the following simplified form of Hamilton's equations,

$$\begin{aligned} \frac{d\phi}{dt} &= +\frac{\partial T}{\partial \rho} \\ \frac{d\rho}{dt} &= -\frac{\partial V}{\partial \phi}. \end{aligned}$$

The joint system is evolved in Stan using a leapfrog integrator. The leapfrog algorithm takes discrete steps of a small time value ϵ and then alternates half-step updates in the momentum and full-step updates for the parameter as follows,

$$\begin{aligned} \rho &\leftarrow \rho - \frac{\epsilon}{2}\frac{\partial V}{\partial \phi} \\ \phi &\leftarrow \phi + \epsilon\, \Sigma\, \rho \\ \rho &\leftarrow \rho - \frac{\epsilon}{2}\frac{\partial V}{\partial \phi}. \end{aligned}$$

After applying L leapfrog steps, the resulting state (ρ^*, ϕ^*) is considered the next proposal. In order to account for numerical inaccuracies of the leapfrog integration step, this proposal will finally be applied to a Metropolis acceptance step, where the probability of accepting the proposal (ρ^*, ϕ^*) is

$$\alpha(\rho^*, \phi^* | \rho, \phi) = \min\Big(1,\ \exp\big(H(\rho,\phi) - H(\rho^*,\phi^*)\big)\Big).$$

If the proposed state is rejected, the previous parameter value ϕ is used in the next iteration.

Due to independence of the parameter ϕ and momentum ρ, the marginal $p(\phi)$ is taken by simply disregarding the momentum samples. This results in efficient sampling directly from $p(\phi)$.

4.2 Posterior MCMC Sampling

This section of the report describes the Bayesian model fitting process to data for any given patient. The Monte Carlo algorithm used in this project was identical for each patient. The procedure to generate m 14 dimensional posterior samples of $\boldsymbol{\lambda}$ from posterior density $p(\boldsymbol{\lambda}|D)$ for any given patient is outlined below.

- The initial parameter value $\boldsymbol{\lambda_1}$ was sampled randomly from the support of the priors as described in Section 2.1.

- Given the current parameter value $\boldsymbol{\lambda_i}$ with $i \in \{1,\ldots,m-1\}$, the next parameter in the Markov chain is chosen as follows.

    1. The IVP 6 was solved using a fouth and fifth order Runge-Kutta method [19], and the function $R(\ t \mid \boldsymbol{\theta_i}, \boldsymbol{y_i^{(0)}}\ )$ was calculated for $t \in \{t_j; j = 1,\ldots,n\}$ the set of n data time points for the given patient.
    2. The posterior probability was calculated using the likelihood for the given patient $p(D|\boldsymbol{\lambda_i})$ and the prior probability $\pi(\boldsymbol{\lambda_i})$ as defined in Equations 11 and 12 respectively.
    3. As described in Section 4.1, the auxiliary momentum variable ρ was sampled from a multivariate normal. The Hamiltonian was defined, and a proposal (ρ^*, ϕ^*) was generated after L leapfrog steps of the Hamiltonian equations. In Stan, the number of leapfrog steps L is dynamically adapted during sampling to optimize mixing.
    4. With probability

$$\alpha(\rho^*, \boldsymbol{\lambda}^* | \rho, \boldsymbol{\lambda_i}) \quad = \quad \min\Big(\ 1,\ \exp\big(H(\rho, \boldsymbol{\lambda_i}) - H(\rho^*, \boldsymbol{\lambda}^*)\big)\ \Big),$$

    set $\boldsymbol{\lambda_{i+1}} = \boldsymbol{\lambda}^*$. Otherwise set $\boldsymbol{\lambda_{i+1}} = \boldsymbol{\lambda_i}$.

5 MCMC DIAGNOSTICS

In this section, the MCMC chains for Patients 10 (Remission) and 25 (Relapse) are analysed to outline the procedure for all patients. For each of the two examples in this section, 4 MCMC chains of length 30000 with warmup length of 3000 were used. In this section, convergence of the chains and overall suitability of the results will be discussed.

5.1 Convergence

There are many different techniques to determine convergence of a MCMC chain [20]. In this report, the Gelman and Rubin test statistic will be considered along with visual inspections of trace plots and the auto-correlation function.

Patient 25 (Relapse)					Patient 10 (Remission)				
Par.	$\hat{R}$	n_{eff}	mean	sd	Par.	$\hat{R}$	n_{eff}	mean	sd
ν	1.00	23503	0.04	0.05	ν	1.00	34102	0.01	0.00
θ_1	1.00	42872	0.48	0.29	θ_1	1.00	63208	0.54	0.29
θ_2	1.00	44350	0.58	0.27	θ_2	1.00	51645	0.56	0.27
θ_3	1.00	49571	0.52	0.28	θ_3	1.00	51822	0.57	0.28
θ_4	1.00	45010	0.64	0.24	θ_4	1.00	45254	0.47	0.26
θ_5	1.00	34796	0.40	0.27	θ_5	1.00	47553	0.53	0.29
θ_6	1.00	40262	0.53	0.27	θ_6	1.00	53585	0.46	0.28
θ_7	1.00	32491	0.34	0.28	θ_7	1.00	45448	0.57	0.26
θ_8	1.00	43233	0.62	0.27	θ_8	1.00	25889	0.40	0.26
$y_1^{(0)}$	1.00	37102	0.14	0.14	$y_1^{(0)}$	1.00	50223	0.15	0.14
$y_2^{(0)}$	1.00	32750	0.18	0.16	$y_2^{(0)}$	1.00	50731	0.06	0.07
$y_3^{(0)}$	1.00	64139	0.04	0.05	$y_3^{(0)}$	1.00	41590	0.06	0.05
$y_4^{(0)}$	1.00	59547	0.06	0.07	$y_4^{(0)}$	1.00	37539	0.10	0.10
$y_5^{(0)}$	1.00	64177	0.10	0.09	$y_5^{(0)}$	1.00	39764	0.11	0.10

Table 1: Summary statistics for the posterior distributions for Patients 25 (Relapse) and 10 (Remission) respectively

Figure 6 shows the trace of the log-posterior for each of the four Markov chains for relapse Patient 25. As can clearly be seen, the log-posterior for each chain seems to have converged well in advance of the end of warm-up at iteration 3000. Each individual trace plot for the 14 parameters was also checked and showed similar convergence. The log-posterior trace plot for remission Patient 10 can be found in the appendix in Figure 20. It also indicated convergence of the log-posterior.

Figure 7 shows the average auto-correlation plot for the log-posterior across the four chains for relapse Patient 25. As can be seen, the auto-correlation rapidly decreased to zero which indicated good mixing in the log-posterior in the Markov chains. Again, the individual auto-correlation plots for the 14 parameters also indicated efficient mixing. For remission Patient 10, the auto-correlation function for the log-posterior and the individual parameters also indicated good mixing in the Markov chains. The auto-correlation plot for the log-posterior for Patient 10 can be seen in the appendix in Figure 21.

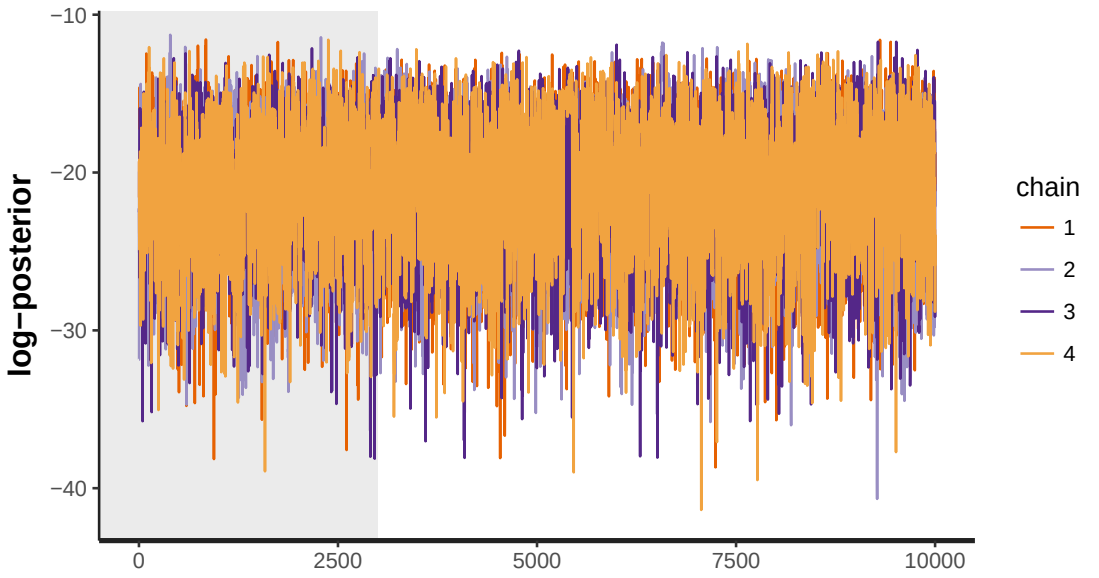


Figure 6: Trace of the log-posterior for Patient 25 (Relapse). Warm-up region for the first 3000 iterations is shaded.

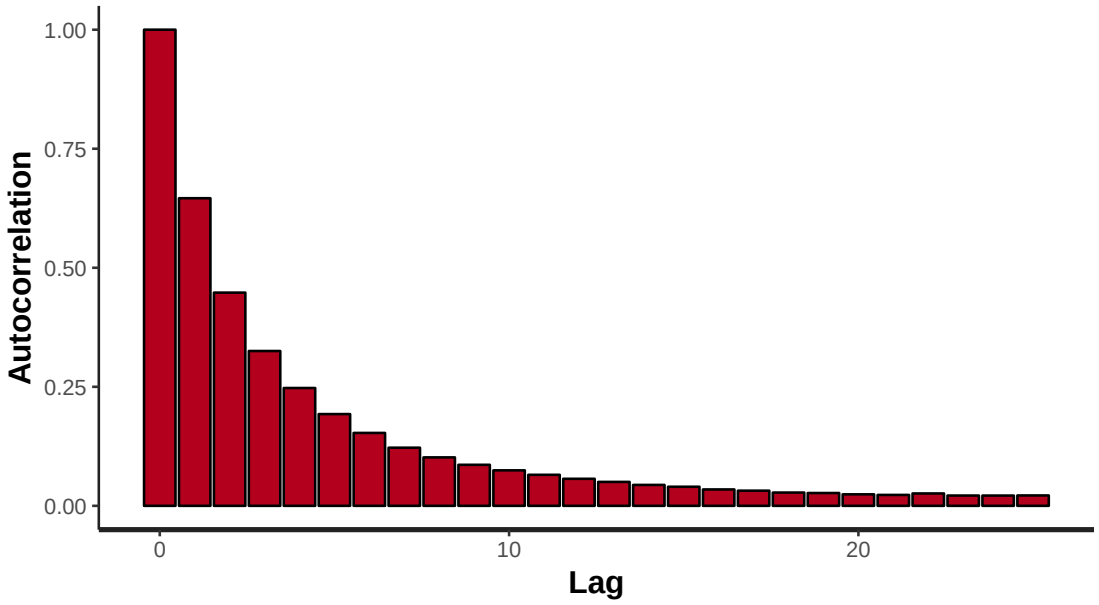


Figure 7: Average auto-correlation of the log-posterior across the four chains for Patient 25 (Relapse).

Related to the auto-correlation, the effective sample size n_{eff} is defined as

$$n_{\text{eff}} = \frac{n}{1+2\sum_{k=1}^{\infty}\rho(k)}, \tag{20}$$

where $\rho(k)$ is the average auto-correlation across the four chains at lag k, and n is the total sample size after warm-up iterations have been removed. In this case $n = 108,000$. As can be seen from Table 1, the effective sample size for each parameter for Patients 25 and 10 are all quite large relative to the total number of iterations n. This again is a good indication of good mixing.

Finally, the Gelman-Rubin convergence diagnostic, sometimes referred to as the potential scale reduction factor (PSRF), can be used to quantify convergence in the Markov chain for any uni-variate parameter, and is denoted $\hat{R}$. The PSRF is a function of the between-chain and within-chain variability for m Markov chains initialised randomly. Values of $\hat{R}$ close to 1 indicate convergence in the Markov chain for that parameter [21]. In this report, there were 4 chains initialised randomly for all patients. As can be seen in Table 1, $\hat{R}$ is within two decimal places of 1 for each of the 14 parameters for both Patients 10 and 25. This was again an indication of convergence in the Markov chains. All other patients showed similar levels of convergence to Patients 10 and 25.

5.2 Solution Suitability

Section 5.1 showed strong indications for convergence within the Markov chains. However, to determine if the posterior distribution is satisfactory, the final solutions were compared to the data. For each Patient 10 and 25, k 14-dimensional posterior samples were taken randomly from the sampled MCMC posterior chains. For each $i \in \{1,2,\ldots,k\}$, the corresponding predicted BCR-ABL transcript level $R(\ t \mid \boldsymbol{\theta_i}, \boldsymbol{y_i^{(0)}})$ was plotted for t in the range of the data time values for each patient, and where $\boldsymbol{\theta_i}$ and $\boldsymbol{y_i^{(0)}}$ were taken from the i^{th} randomly selected posterior sample. Below, Figure 8 shows plots of such sample solutions with $k = 30$ for each Patient 10 and 25 respectively.

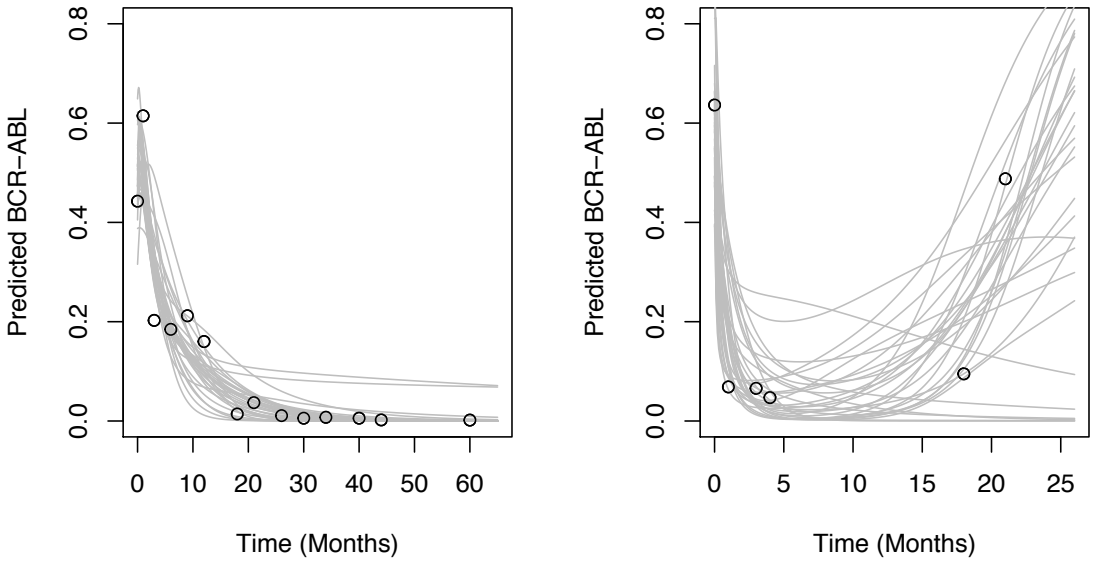


Figure 8: Predicted BCR-ABL levels given posterior samples for Patients 10 and 25 respectively.

In each case, the posterior distributions of the parameter sets produced solutions to the system of ODEs that matched the data well. However, as seen in Figure 8, there was large variability in the solutions. In particular, in the MCMC parameter space for the relapse example, there were a small number of solutions that could be categorised as remission. This was not the case for the remission example. The posterior distribution of the parameter sets for this example produced solutions that consistently described remission dynamics. All other patients produced very similar results.

5.3 Maximum a Posteriori Estimation

Given the posterior parameter distribution as sampled from the MCMC algorithm, it was decided to choose a single sample from each Markov chain to highlight convergence of that chain to the data. In this project the maximum a posteriori (MAP) estimate was chosen.

It should be noted that the posterior mean was not a suitable choice to represent the posterior distribution. This can be visualised by considering the lower dimensional example in Figure 9 of the subspace of $\mathbb{R}^3$ of high posterior density with HMC posterior samples plotted as green points. In this case, the posterior mean is not necessarily a point on this surface of high posterior density, and therefore, it would not necessarily match the data well. However,

the MAP estimate is simply the point on this surface that has the highest posterior probability. Therefore, it will closely fit to the data.

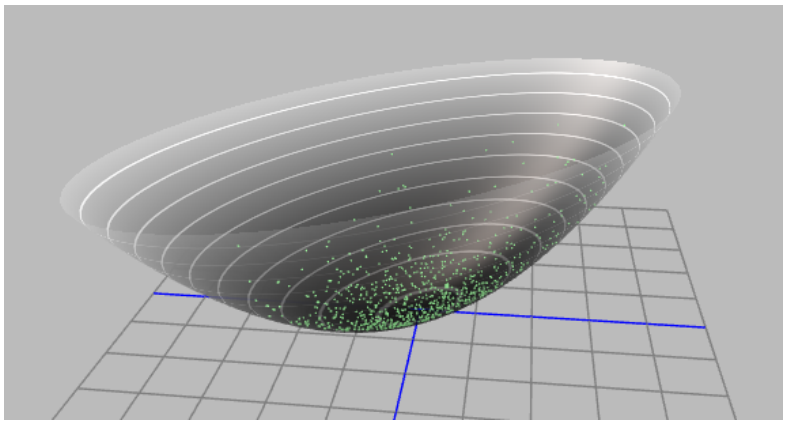


Figure 9: Visualisation of posterior samples (green) from a high dimensional surface solution as taken from *http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html*

In this project the MAP estimate $\boldsymbol{\lambda}^{\text{MAP}}$ was taken to be the 14-dimensional parameter with the highest posterior probability. It is defined as

$$\boldsymbol{\lambda}^{\text{MAP}} \quad = \quad \underset{\boldsymbol{\lambda}}{\arg\max}\, p(\boldsymbol{\lambda}|D) \tag{21}$$

where $\boldsymbol{\lambda} = \left(\theta_1,\ldots,\theta_8,y_1^{(0)},\ldots,y_5^{(0)},v\right)$. However, the MAP estimate was approximated by the MCMC sample in a given chain with the highest posterior probability, denoted $\tilde{\boldsymbol{\lambda}}$, and defined

$$\tilde{\boldsymbol{\lambda}} \quad = \quad \underset{\boldsymbol{\lambda_i};i=1,\ldots,m}{\arg\max}\; p(\boldsymbol{\lambda_i}|D) \tag{22}$$

where $\{\boldsymbol{\lambda_1},\ldots,\boldsymbol{\lambda_m}\}$ is the set of m 14-dimensional posterior samples from a given Markov chain once the warm-up has been removed. Note that in the 3-dimensional example in Figure 9 above, the approximate MAP estimate is simply the green MCMC posterior sample point with the highest posterior probability.

Figure 10 shows the plot of the predicted BCR-ABL levels given the approximate MAP estimate $R(\ t \mid \tilde{\boldsymbol{\lambda}})$ for Patient 25 (in red). The 95% confidence interval of the predicted BCR-ABL level is also given at each data point. This confidence interval $C_{95}(t)$ at time t is easily given due to the normal likelihood as

$$C_{95}(t) \quad = \quad \left[\ R(\ t \mid \tilde{\boldsymbol{\lambda}}) \ - \ z_{0.025}\sqrt{\tilde{v}}\ ,\ \ R(\ t \mid \tilde{\boldsymbol{\lambda}}) \ + \ z_{0.025}\sqrt{\tilde{v}}\ \right], \tag{23}$$

where $\tilde{v}$ is the 14th dimension of the approximate MAP estimate $\tilde{\boldsymbol{\lambda}}$, and $z_{0.025}$ is the significance level of a standard normal. Figure 10 also shows that the approximate MAP estimate matches the data well as each data point was within the corresponding 95% confidence interval around the MAP prediction at that time point. However, this highlights the innaccuracy with the normal likelihood as mentioned in Section 1.5 as the confidence intervals clearly leave the range of possible values, $[0,1]$, of the predicted BCR-ABL level. The corresponding plot for Patient 10 showed the same results and can be seen in the appendix in Figure 22.

In Figure 10, the blue dashed line represents the predicted BCR-ABL level given the posterior mean $R(\ t \mid \hat{\boldsymbol{\lambda}})$, where $\hat{\boldsymbol{\lambda}}$ is the posterior mean. As can be seen, the posterior mean does not match the data well which highlights the importance of the MAP estimate.

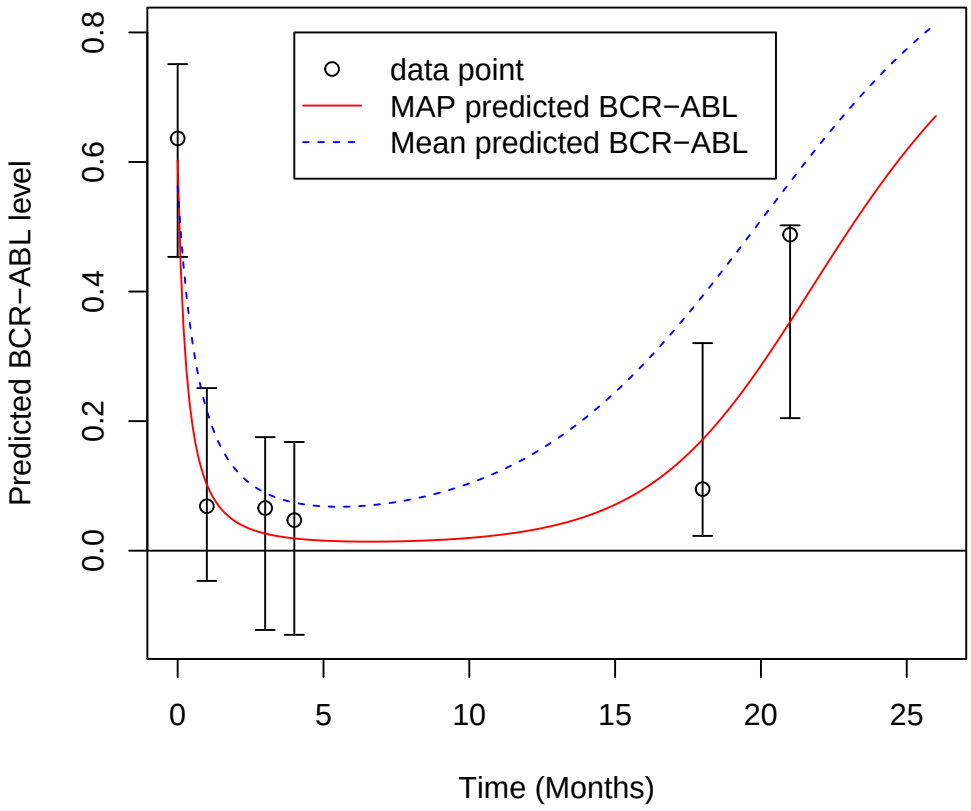


Figure 10: Approximate MAP BCR-ABL predictions for Patient 25 with confidence intervals for the predicted value at each data point.

6 PATIENT ANALYSIS

The next part of this project was to analyse the difference between individual patients. The MCMC algorithm, as described in Section 4, was performed on all 69 patients with the exception of Patient 66. As mentioned in Section 3, Patient 66 was removed as there was only one data measurement for this patient. As defined in Section 3, $I = \{1,2,\ldots,65,67,\ldots,69\}$ is therefore the set of patients to be considered.

For each patient, a chain of 14-dimensional parameters of length 10000, as sampled from the posterior distribution, was generated. A warm-up period of length 500 was removed from each MCMC chain. Note that the length of the chains was reduced to improve computation time. However, similar diagnostic tests to those in Section 5 were computed for each of these patients to ensure convergence of the Markov chains.

6.1 Linear Discriminant Analysis

The first question was to determine the impact of each model parameter on the clinical outcome. It was decided that only $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_8)$ should be considered to impact the clinical outcome. This was chosen since patients with varying initial BCR-ABL levels go into both remission and relapse. Therefore, it seemed reasonable to disregard the initial conditions for the Initial Value Problem 6 in this analysis.

Linear discriminant analysis (LDA) was first carried out on posterior samples of $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_8)$. For each Patient $i \in I$, 10000 posterior samples $\boldsymbol{\theta}^{(i,m)}$ for $m \in \{1,\ldots,10000\}$ were taken randomly with replacement from the MCMC chain of Patient i.

For each posterior sample $\boldsymbol{\theta}^{(i,m)}$ with $i \in I$ and $m \in \{1,\ldots,10000\}$, a corresponding classification label $l^{(i,m)}$ was defined as

$$l^{(i,m)} = \begin{cases} 1 & \text{if} \quad i \in \{1,7,25,26,47,68\} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

where 1 corresponds to samples from the posterior distribution of a relapse patient, and 0 corresponds to samples from the posterior distribution of a remission patient. Here $\{1,7,25,26,47,68\}$ was taken to be the set of relapse patients.

The classification rule $\hat{l}$, according to linear discriminant analysis, of posterior sample $\boldsymbol{\theta}$ with unknown label l is

$$\hat{l} = \begin{cases} 1 & \text{if } \boldsymbol{w}\cdot\boldsymbol{\theta} > c \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $\boldsymbol{w}$ and c were chosen to define the hyperplane that best separates the "cloud" of posterior samples $\boldsymbol{\theta}^{(i,m)}$ for $i \in I$ and $m \in \{1,\ldots,10000\}$ according to the corresponding classifier $l^{(i,m)}$ of each. The components of $\boldsymbol{w}$ are plotted in Figure 11.

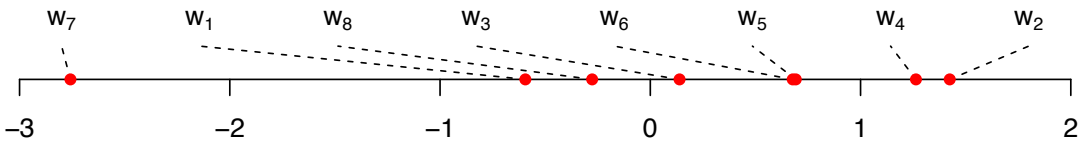


Figure 11: Parameter contributions to the LDA discriminant between 10000 posterior samples of relapse and remission patients with $\boldsymbol{w} = (-0.59, 1.42, 0.14, 1.26, 0.70, 0.68, -2.76, -0.28)$.

As can be seen in Figure 11, w_7 had the largest absolute value. Therefore, the θ_7 coefficient had the largest impact on the linear discriminant. This was an indication that θ_7 had the largest impact on clinical outcome. This made intuitive sense as θ_7 is the coefficient in the system of ODEs 4 that controls the rate of transition from leukemic stem cells y_4 to differentiated leukemic cells y_5.

6.2 Patient Clustering

The next part of the project focused on performing cluster analysis on the patients according to the posterior distributions of parameter $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_8)$. Unlike Section 6.1, no clinical outcome labels were assumed. There are many forms of such unsupervised clustering:

- *Hierarchical clustering* - Often referred to as connectivity based clustering. The objects are separated according to some predefined metric measuring the dissimilarity between them.
- *Centroid clustering* - A predefined number of clusters k can be easily found where each object is assigned to a cluster represented by a single value. The k-means algorithm is the most common form of centroid clustering.

- *Distribution based clustering* - Each group is assumed to have a defined probability distribution. A latent variable can be used to classify each object. Algorithms such as the expectation-maximization (EM) algorithm can then be used to estimate the most likely group for a given object.
- *Density based clustering* - Objects are divided into clusters according to areas of high density. Sparse areas of low density are required to separate clusters.

Given that each object in the clustering represents an approximation to a multivariate posterior distribution, it was decided that clustering with simple implementation and intuitive understanding was most important. For that reason, hierarchical clustering was chosen as it avoided any assumed distribution of patients per group. Instead, it simply required a predefined metric.

6.3 Dissimilarity Between Patients

A metric is any real-valued function $d : I \times I \rightarrow [0,\infty)$ such that the following hold for all $x, y, z \in I$:

- $d(x,y) \geq 0$.
- $d(x,y) = 0 \quad \iff \quad x = y$.
- $d(x,y) \;=\; d(y,x)$.
- $d(x,z) \;\leq\; d(x,y) \;+\; d(y,z)$.

In this project, I represents the set of patients as defined in Equation 19. The dissimilarity between a pair of patients was taken to be a measure of dissimilarity between their corresponding posterior parameter distributions.

There are many defined measures for dissimilarity between probability densities in literature including the Hellinger distance, Wasserstein metric, and Kolmogorov-Smirnov statistic [22]. Many of these measures can be easily calculated given the distributional form of the probability density, and most can easily be approximated for uni-variate sample distributions. However, as is the case in this project, calculating the dissimilarity between two multivariate sample distributions is a difficult task. Given time constraints, such dissimilarities between

patients were only considered for the univariate marginal distributions in each dimension. The Kolmogorov-Smirnov statistic was considered to measure the dissimilarity between the marginal posterior distribution in each dimension between a pair of patients. However, as this measure of dissimilarity did not capture the joint behavior in each posterior density, this method did not produce a satisfying clustering of the patients.

Therefore, to consider the joint behavior of each posterior distribution, a dissimilarity was used that considered the Euclidean distance between individual samples. Given two 8 dimensional samples $\boldsymbol{\theta_i}$ and $\boldsymbol{\theta_j}$ as sampled from the posterior distributions of Patients $i, j \in I$, the Euclidean distance between these two samples is

$$d_{\text{eucl}}(\boldsymbol{\theta_i}, \boldsymbol{\theta_j}) \quad = \quad \sqrt{\sum_{k=1}^{8} \left(\theta_{ik} - \theta_{jk}\right)^2} \tag{26}$$

where θ_{ik} is the k^{th} parameter for Patient i. Given Patients $i, j \in I$, the average Euclidean distance between posterior samples from their corresponding distributions was estimated using a Monte Carlo technique.

For an arbitrary pair of Patients $i, j \in I$, M Monte Carlo samples $\boldsymbol{\theta_i^{(m)}}$ and $\boldsymbol{\theta_j^{(m)}}$ for $m \in \{1, \ldots, M\}$ were independently sampled with replacement from the MCMC posterior chains of Patients i and j respectively. The Monte Carlo estimate $d_M(i, j)$ of the average Euclidean distance between sampled pairs between Patients $i, j \in I$ was calculated as

$$d_{\text{M}}(i, j) \quad = \quad \frac{1}{M} \sum_{m=1}^{M} d_{\text{eucl}}(\boldsymbol{\theta_i^{(m)}}, \boldsymbol{\theta_j^{(m)}}) \tag{27}$$

where $\boldsymbol{\theta_i^{(m)}}$ and $\boldsymbol{\theta_j^{(m)}}$ are the m^{th} randomly selected posterior samples from the MCMC chains of Patients i and j respectively.

Given Monte Carlo samples $\boldsymbol{\theta_{10}^{(m)}}$ and $\boldsymbol{\theta_{25}^{(m)}}$ for $m \in \{1, \ldots, 100000\}$ taken independently at random from the MCMC posterior chains of Patients 10 and 25 respectively, Figure 12 shows the density of the euclidean distances $d_{\text{eucl}}(\boldsymbol{\theta_{10}^{(m)}}, \boldsymbol{\theta_{25}^{(m)}})$ for $m = 1, \ldots, 100000$.

The true mean Euclidean distance $\mu_{10,25}$ between samples taken from the pair of Markov chains is unknown. However, as approximated through the Monte Carlo technique described previously, the approximated Monte Carlo sample mean $d_M(10, 25)$ is approximately 1.14. The

standard error $SE_{10,25}$ of the sample mean is

$$SE_{10,25} \quad = \quad \frac{s}{\sqrt{M}}\,, \tag{28}$$

where $M = 100000$ is the number of samples taken from each posterior distribution, and $s = 0.24$ is the standard deviation of the Monte Carlo sample of M Euclidean distances. Therefore, a 95% confidence interval $CI_{10,25}$ for the mean Euclidean distance $\mu_{10,25}$ between posterior samples from the Markov chains of Patients 10 and 25 is

$$CI_{10,25} \quad = \quad \left[\; d_M(10,25) \;-\; t_{M-1,0.025}\; SE_{10,25}\,,\;\; d_M(10,25) \;+\; t_{M-1,0.025}\; SE_{10,25} \;\right], \tag{29}$$

where $t_{M-1,0.025}$ is the corresponding significance level of a t-distribution with $M-1$ degrees of freedom. The length of this particular confidence interval was approximately 0.0029. This indicated strong accuracy of the estimate $d_M(10,25) = 1.14$.

A corresponding estimate $d_M(i,j)$ and confidence interval $CI_{i,j}$ was calculated for all pairs of Patients $i, j \in I$. The estimates for distinct patients ranges from 0.59 to 1.57 and the largest confidence interval length was found to be 0.0037. This strongly indicated the validity of approximating the mean Euclidean distance between posterior samples from the MCMC chains by a Monte Carlo technique.

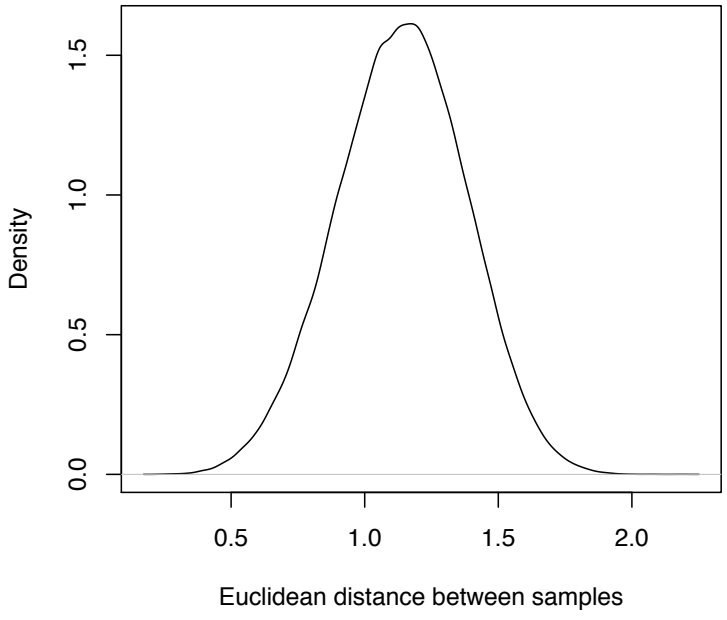


Figure 12: Probability density of the Euclidean distance between 100000 posterior samples randomly taken from each of the MCMC chains of Patients 10 and 25.

It should also be noted that the dissimilarity defined in Equation 27 is not a true metric. Given two sets of M independent Monte Carlo samples from the MCMC chain of Patient $i \in I$, note that the dissimilarity $d_M(i,i) > 0$. Therefore the second property in the definition of a metric does not hold. However, the remaining three properties hold asymptotically. Although d_M cannot be considered a true metric, it does measure dissimilarities between probability distributions in a natural way. For that reason, this dissimilarity was used for clustering in Section 6.5. The dissimilarities between patients will be visualised through a multidimensional scaling projection in the next section.

6.4 Multidimensional Scaling

Consider the simple problem of calculating the Euclidean distance between all pairs of n 2-dimensional points on the plain. Working in reverse to find the set of n 2-dimensional points is, however, more difficult. Multidimensional scaling (MDS), as developed by Torgerson [23], provides a method of approximating a solution to this inverse problem. Given a $n \times n$ symmetric distance matrix $\boldsymbol{D} = [d_{ij}]$, where d_{ij} is some distance between objects i and j, the classical MDS algorithm is outlined as follows [24].

1. Define the square distance matrix $\boldsymbol{D^{(2)}} = [d_{ij}^2]$.

2. Define the centering matrix $\boldsymbol{B} = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D^{(2)}}\boldsymbol{J}$ where $\boldsymbol{J} = \boldsymbol{I_n} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\boldsymbol{T}}$ with $n \times n$ identity $\boldsymbol{I_n}$ and n-dimensional vector $\mathbf{1}$ with each entry 1. This process is called doubly centering.

3. Calculate the m largest eigenvalues $\lambda_1, \ldots, \lambda_m$, and corresponding eigenvectors $\boldsymbol{v_1}, \ldots, \boldsymbol{v_m}$ of $\boldsymbol{B}$, where m is the dimension of the output points. In this case $m = 2$.

4. Finally, set $\boldsymbol{X} = \boldsymbol{V_m}\boldsymbol{\Lambda_m^{1/2}}$, where $\boldsymbol{V_m}$ is the matrix of m eigenvectors and $\boldsymbol{\Lambda_m}$ is the diagonal matrix of m eigenvalues.

The output $n \times m$ matrix $\boldsymbol{X}$ is a matrix of n m-dimensional points that maintain the distances between objects according to matrix $\boldsymbol{D}$ as best as possible.

Given the measure of dissimilarity $d_{\mathrm{M}}(i,j)$ between Patients $i, j \in I$, calculated with M Monte Carlo samples, the above technique provides a simple way of visualizing the distances between the set of patients I by producing a MDS plot in 2-dimensional space.

By using $M = 100000$ Monte Carlo samples, the distance matrix $\boldsymbol{D}$ is defined as

$$\boldsymbol{D} = [d_{\mathrm{M}}(i, j)] \tag{30}$$

with Patients $i, j \in I$. Note that for simplicity in implementation with standard clustering techniques, the distance matrix $\boldsymbol{D}$ was set to be symmetric with 0 along the diagonal. Although this provided a false representation of the distance $d_{\mathrm{M}}(i, i)$ between any Patient $i \in I$ and itself, the matrix was still able to capture dissimilarity between distinct patients. The MDS algorithm was then implemented for distance matrix $\boldsymbol{D}$.

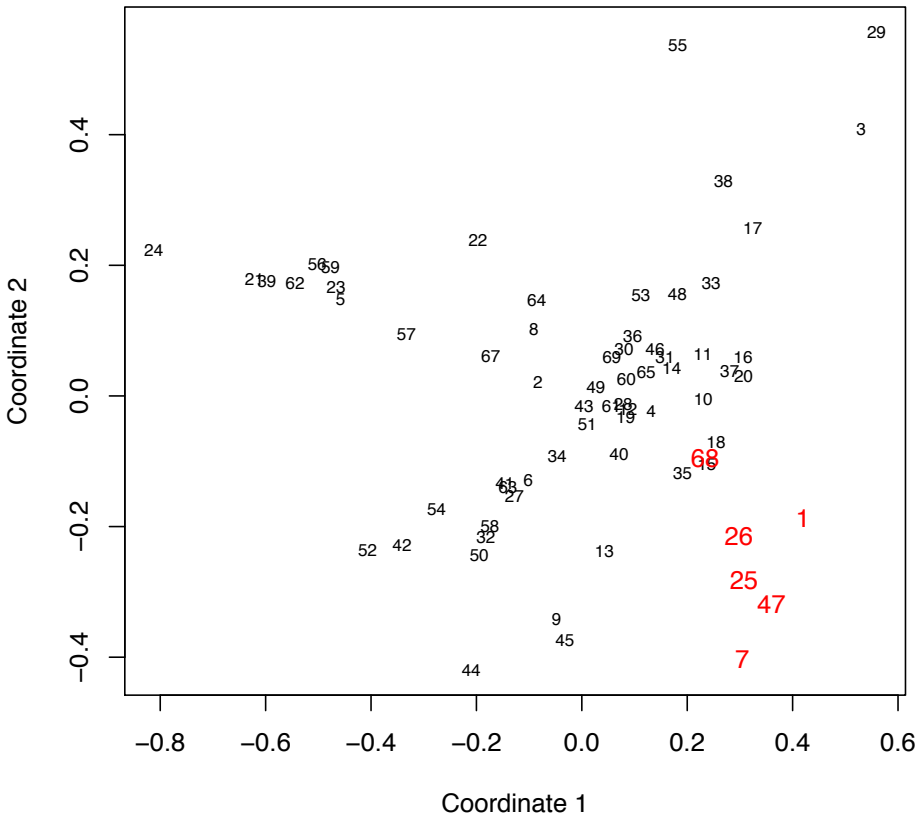


Figure 13: MDS plot for the average Euclidean distance between sampled pairs with relapse patients coloured in red.

As can be seen in Figure 13, the average Euclidean distance between samples from the joint posterior distribution of $\boldsymbol{\theta}$ separated the patients according to clinical outcome. Although relapse Patient 68 failed to separate completely from the remission patients, it should be noted that there were only three data points for this particular patient. Therefore, the posterior

parameter distribution for Patient 68 was likely heavily influenced by the prior. This likely explains its failure to completely separate from the remission patients. However, all other relapse patients were clearly separated into one cluster. Although the average Euclidean distance was estimated using a Monte Carlo technique (as described in Section 6.3), this definition of the distance between patients accounted for the joint behavior of the parameters $\theta_1, \ldots, \theta_8$ in the posterior distribution. This was likely the reason it outperformed the Kolmogorov-Smirnov distance metric on the marginal posterior distributions in each dimension.

As a result of the separation between remission and relapse patients in the MDS plot in Figure 13, the distance matrix $\boldsymbol{D}$, as defined by the dissimilarity $d_M(i,j)$ between Patients $i, j \in I$ with M Monte Carlo samples, was then used to perform hierarchical clustering on the set of patients I.

6.5 Hierarchical Clustering

Hierarchical clustering is a simple technique that forms a hierarchy of clusters of objects given a measure of dissimilarity between them. There are many different forms of hierarchical clustering which mainly differ according to a *linkage criterion*. The linkage criterion determines a measure of distance between two clusters. Single-linkage, average-linkage and complete-linkage [26] were used as part of this project. However, in this report, hierarchical clustering with complete-linkage is presented as it resulted in the most natural clustering of the patients. The distance between two clusters of objects, A and B, is defined under complete-linkage as

$$d_c(A,B) = \max\{d(a,b) | a \in A, b \in B\}, \tag{31}$$

where $d(a,b)$ is the predefined distance between objects a and b. Normally, hierarchical clustering of n objects is performed in a "bottom up" approach as follows.

1. Define each of the n objects to be their own cluster.
2. Join the pair of clusters A and B with the smallest distance $d_c(A,B)$ between them to form a single cluster $\{A,B\}$. This strictly reduces the number of clusters.
3. Repeat step 2 until only one cluster of all n objects remains.

The previous algorithm can be viewed as forming the clusters in reverse. Starting from one cluster, the clusters are iteratively divided until a desired number of clusters remain. Note that

in step 2 of the above algorithm, a problem with non-uniqueness may occur if multiple pairs of clusters have the same smallest distance between them. This was unlikely to occur in this project due to the real-valued nature of the distance metric. However, this potential problem is discussed in [27].

Figure 14 shows the dendrogram for the complete-linkage clustering of the patients using the distance matrix $\boldsymbol{D}$ defined in Equation 30. By visual examination of both the dendrogram in Figure 14 and MDS plot in Figure 13, it was decided to cluster the patients into 4 groups. The corresponding clustered MDS plot can be seen in Figure 15. This decision to cluster into four groups allowed for decent separation between the groups in the corresponding MDS plot. However, it should be noted that there is some overlap between the black and red groups as coloured in Figure 15.

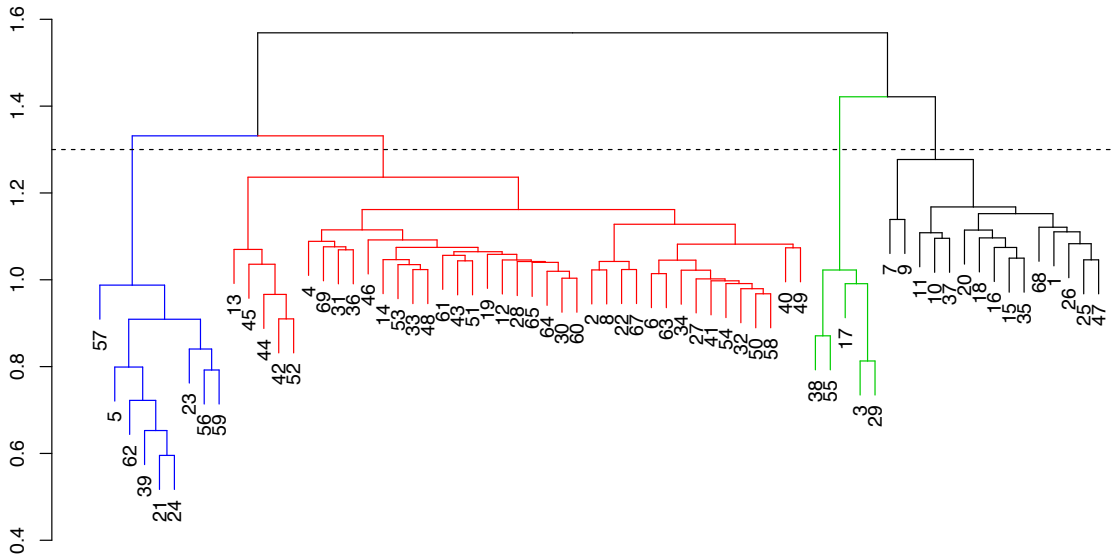


Figure 14: Dendrogram for hierarchical clustering with complete linkage. The dashed line represents a cut in order to produce 4 clusters.

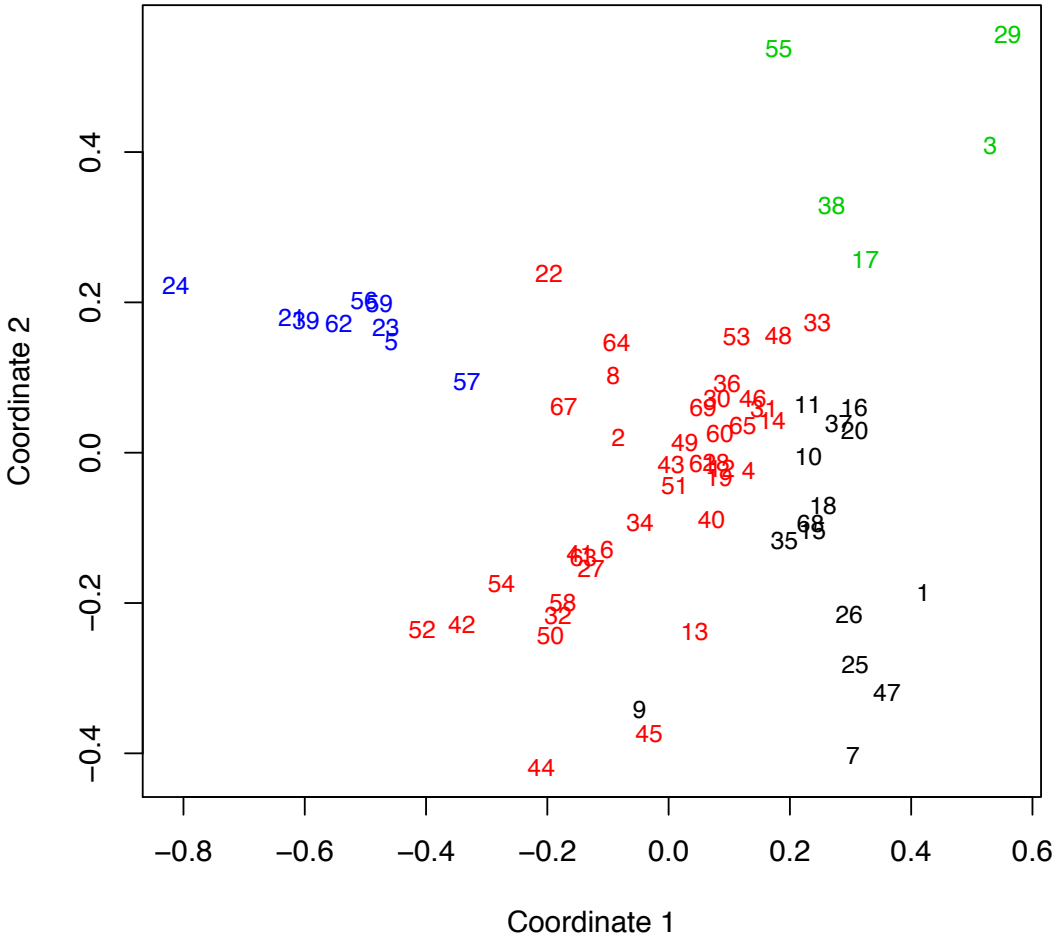
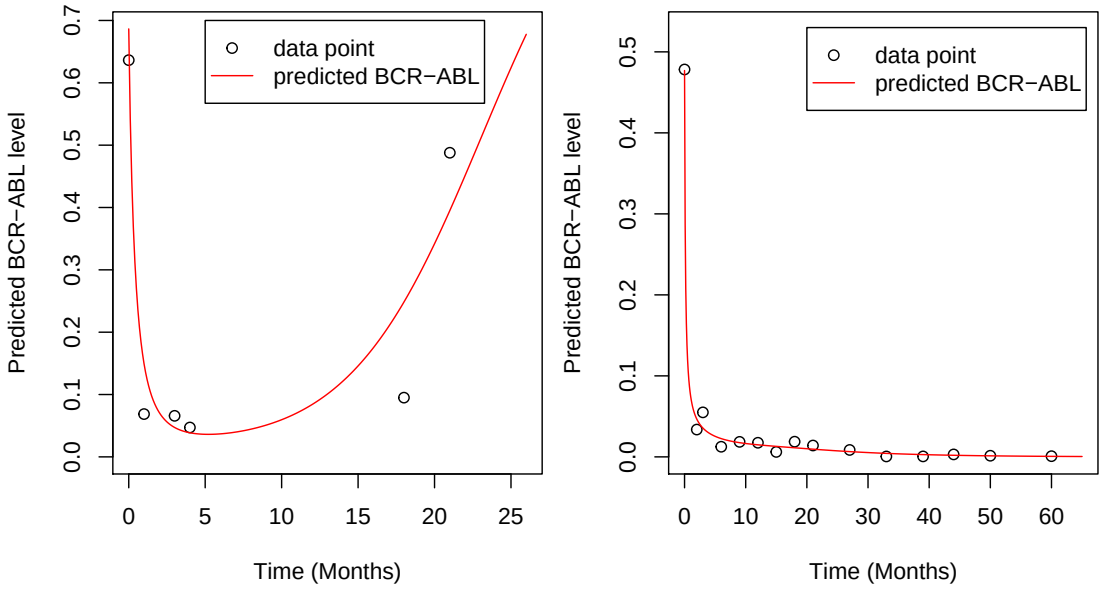
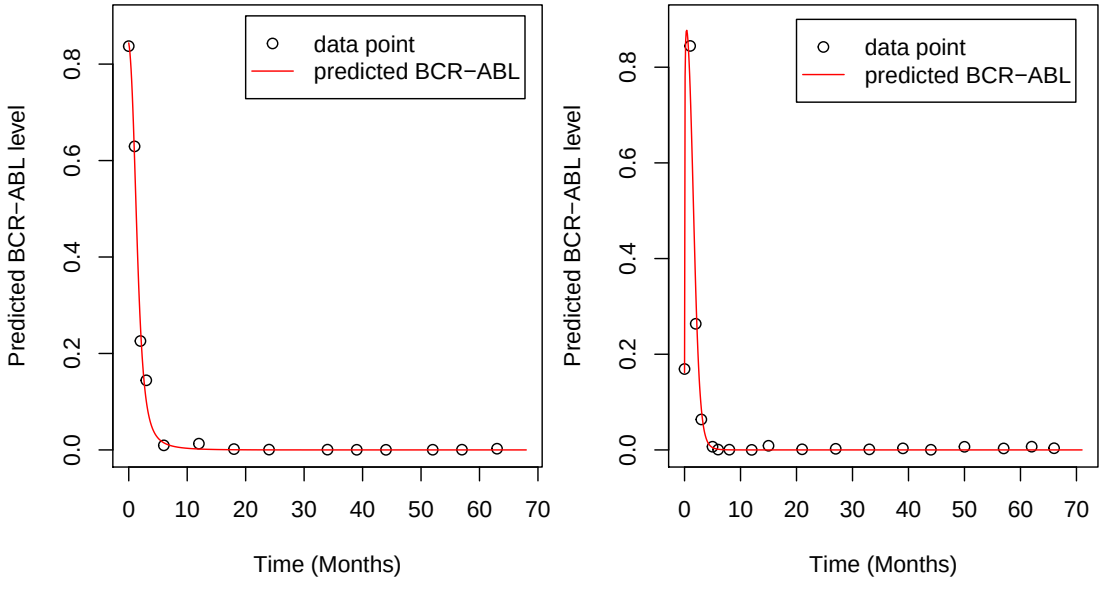


Figure 15: MDS plot using distance matrix ***D*** clustered by colour according to complete linkage hierarchical clustering in Figure 14.



(a) Patient 25. **(b)** Patient 15.

Figure 16: Plots of predicted BCR-ABL levels according to the posterior mode solution for Patients 25 and 15 respectively. Both patients clustered into the black group as in Figure 15.



(a) Patient 6 (Red Group). **(b)** Patient 62 (Blue Group).

Figure 17: Plots of predicted BCR-ABL levels according to the posterior mode solution for Patients 6 and 62 respectively. The patients were clustered into the red and blue groups respectively as in Figure 15.

It should be noted that in Figures 16, 17, and 18 the confidence intervals around predicted values, as shown in Figure 10, were removed. This was done for simplicity in interpreting the plots as the intention was to highlight the leukemia dynamics in the different groups.

Figure 16 shows the predicted BCR-ABL level plots given the MAP estimate for each of Patients 25 and 15 respectively. Although both patients are clustered into the black coloured group as in Figure 15, Patient 25 was considered to have gone into relapse, and Patient 15 clearly went into remission.

As can be seen in Figure 15, all remission Patients 1,7,25,26,47 and 68 were clustered together into the black group. However, numerous remission patients, including Patient 15 as in Figure 16b, were also clustered into this group. However, by examining Figure 15, it can be seen that the MDS plot did separate the relapse patients (with the exception of Patient 68) from the remission patients within the same cluster.

A different clustering technique may reduce this misclassification error according to the clinical outcome by making better use of the distance matrix $\boldsymbol{D}$ as used to produce the MDS plot in Figure 15.

Figure 17a shows the predicted BCR-ABL level plot given the MAP estimate for Patient 6. This patient was classified into the red group as in Figure 15. This patient can clearly be seen to have gone into remission. As can be seen in Figure 17a, the predicted BCR-ABL level for this patient continued to decrease over time and featured no turning points in the solution. This was the case for almost all patients clustered into the red group.

Figure 17b shows the predicted BCR-ABL level plot given the MAP estimate for Patient 62. Although still clearly in remission, this patient was clustered into the blue group according to the hierarchical clustering. However, there is a clear difference in the progression of the disease to Patient 6 as shown in Figure 17a. Patient 62, first had an increasing BCR-ABL level before eventually going into remission. This was the case for all patients as classified into the blue group.

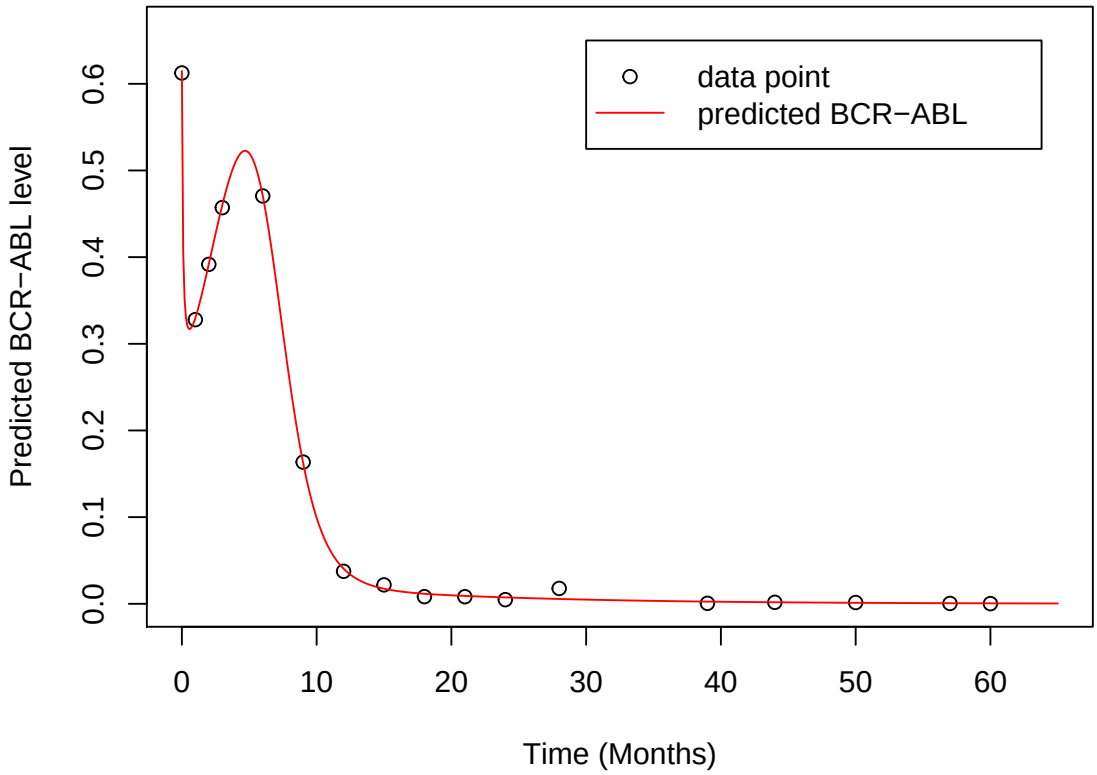


Figure 18: Plot of predicted BCR-ABL levels according to the posterior mode solution for Patient 29, which was clustered into the green group as in Figure 15.

Figure 18 shows the predicted BCR-ABL level plot given the MAP estimate for Patient 29. This patient again went into remission, but unlike patients clustered into the red and blue groups, the BCR-ABL trace for Patient 29 underwent two turning points. The predicted BCR-ABL transcript level initially dropped, before another spike at approximately month 5. Finally, the patient then went into remission.

This behavior was repeated for all but one patient as clustered into the green group. Although Patient 55 was clustered in the green group, the BCR-ABL level decreased over time similar to the majority of patients in the red group.

7 CONCLUSIONS

There were two sides to this project that have been outlined in this report. The first part, as outlined in Sections 1 to 5, was to define and solve the inverse problem as defined in Section 1.1. Parameter inference was carried out, through Markov chain Monte Carlo, on the defining parameters of the Initial Value Problem 6, and the posterior distributions were analysed.

The second part of this project was to use the posterior distribution of the 8-dimensional $\boldsymbol{\theta}$ parameter for each of 68 patients to compare and cluster the patients according to the progression of their chronic myeloid leukemia.

7.1 Solving the Inverse Problem

For each individual patient, the inverse problem was solved by approximating the posterior distribution of the parameter $\boldsymbol{\lambda} = \left(\theta_1,\ldots,\theta_8,y_1^{(0)},\ldots,y_5^{(0)},v\right)$ using a Markov chain Monte Carlo technique. The prior was defined in Section 2, and the chain was simulated through Stan as outlined in Section 4.1.

This methodolgy differed greatly from the previous work by MacLean [10]. Firstly, the authors used an approximate Bayesian computation technique. Although it is a simple and an easily implemented technique, it does not exactly sample according to the posterior. Secondly, the authors aggregated all remission patients before estimating the posterior parameter distribution. By doing so, any differences between the remission patients were not captured.

In this project, the MCMC chain converged excellently for each patient as shown in Section 5, and the posterior samples solved the Initial Value Problem 6 with solutions predicting the data well. For the vast majority of remission patients, the posterior samples exclusively predicted remission dynamics. However, the posterior distributions for relapse patients corresponded to remission predictions on a small number of occasions.

7.2 Patient Clustering

In the previous literature by MacLean [10], all remission patients were assumed to behave similarly, and their data was aggregated to form one overlying remission patient. However, in this project, the posterior parameter distribution was approximated for each patient separately.

This allowed for patients to be compared and clustered according to the posterior distribution of their corresponding MCMC chains.

In Section 6.3, a distance metric between patients based entirely on the posterior samples of $\boldsymbol{\lambda}$ was defined. This allowed the similarities between patients to be visualised by use of a multidimensional scaling plot. Using hierarchical clustering, the patients were then clustered into four groups.

One group contained all relapse patients along with a number of poorly clustered remission patients. The second group consisted almost entirely of patients that showed a continued reduction in the BCR-ABL level over time. With one exception, the third group contained remission patients that showed an initial increase in the BCR-ABL level before immediately going into remission. Finally, the fourth group, again with one exception, consisted of remission patients that went through two turning points in the BCR-ABL level over time.

In general, it was shown that the patients can successfully be clustered into different groups due to differences in the posterior parameter distribution of the Initial Value Problem 6. Each group primarily highlighted different dynamics in the time evolution of chronic myeloid leukemia.

REFERENCES

[1] Tarantola A (1987), Inverse Problem Theory and Methods for Model Parameter Estimation, *Elsevier Scientific Publishing Company*

[2] Müller MC, et al. (2003) Dynamics of BCR-ABL mRNA expression in first-line therapy of chronic myelogenous leukemia patients with imatinib or interferon α/ara-C. *Leukemia* 17:2392-2400

[3] Wang LD, Wagers AJ (2011) Dynamic niches in the origination and differentiation of haematopoietic stem cells. *Nat Rev Mol Cell Bio* 12(10):643-655.

[4] Wang JCY, Dick JE (2005) Cancer stem cells: Lessons from leukemia. *Trends Cell Biol* 15(9):494-501.

[5] Zhang J, et al. (2003) Identification of the haematopoietic stem cell niche and control of the niche size. *Nature* 425(6960):836-841.

[6] Sullivan T.J. (2015) Bayesian Inverse Problems. In: Introduction to Uncertainty Quantification. *Texts in Applied Mathematics, vol 63. Springer*:91-112.

[7] Stuart AM (2010) Inverse problems: A Bayesian perspective. *Acta Numerica*19:451-559.

[8] Baden N, Villadsen J (1982) A family of collocation based methods for parameter estimation in differential equations. *The Chemical Engineering Journal* 23:1-13.

[9] Ramsay JO, et al. (2007) Parameter estimation for differential equations: a generalized smoothing approach. *Series B, Statistical Methodology* 69(5):741-796.

[10] MacLean AL, Filippi S, Stumpf MPH (2014) The ecology in the hematopoietic stem cell niche determines the clinical outcome in chronic myeloid leukemia. *PNAS* 111(10):3883-3888.

[11] Toni T, et al. (2009) Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *J. R. Soc. Interface* 6(31):187-202.

[12] Golightly A, Wilkinson DJ (2011) Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus* 1(6):807-820.

[13] Calderhead B, et al. (2008) Accelerating Bayesian Inference over Nonlinear Differential Equations with Gaussian Processes. *NIPS* 21.

[14] Girolami M (2008) Bayesian inference for differential equations. *Theoretical Computer Science* 408:4-16.

[15] MacLean AL, Lo Celso C, Stumpf MPH (2013) Population dynamics of normal and leukemia stem cells in the haematopoietic stem cell niche show distinct regimes where leukemia will be controlled. *J R Soc Interface* 10(81):20120968.

[16] Roeder I, et al. (2006) Dynamic modeling of imatinib-treated chronic myeloid leukemia: Functional insights and clinical implementations. *Nat Med* 12(10):1181-1184.

[17] Stan Development Team. (2017) Stan Modeling Language - User's Guide and Reference Manual (Version 2.15). Available at *mc-stan.org/users/documentation/index.html*.

[18] Duane S. (1987) Hybrid Monte Carlo. *Physics Letters B*, 195(2): 216-222.

[19] Dormand JR, Prince PJ (1980) A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics* 6:19-26.

[20] Cowles MK, Carlin BP (1996) Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review, *Journal of the American Statistical Association*, 91(434): 883-904.

[21] Gelman A, Rubin DB. (1992) Inference from iterative simulation using multiple sequences, *Statistical Science*, 7:457-472.

[22] Cha SH. (2007) Comprehensive survey on distance/similarity measures between probability density functions, *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4): 300-307.

[23] Torgerson WS. (1952) Multidimensional scaling: I. Theory and method, *Psychometrika*, 17: 401.

[24] Borg I, Groenen P. (2005) Modern Multidimensional Scaling: theory and applications (2nd ed.), *Springer-Verlag*:207-212.

[25] Ghanem R, et al. (2016) Handbook of Uncertainty Quantification, *Springer*.

[26] Rokach L, Maimon O. (2005) Clustering Methods, *Springer US*:321-352.

[27] Fernández A, Gómez S. (2008) Solving non-uniqueness in agglomerative hierarchical clustering using multidendrograms, *Journal of Classification*, 25(1):43-65.

8 APPENDIX

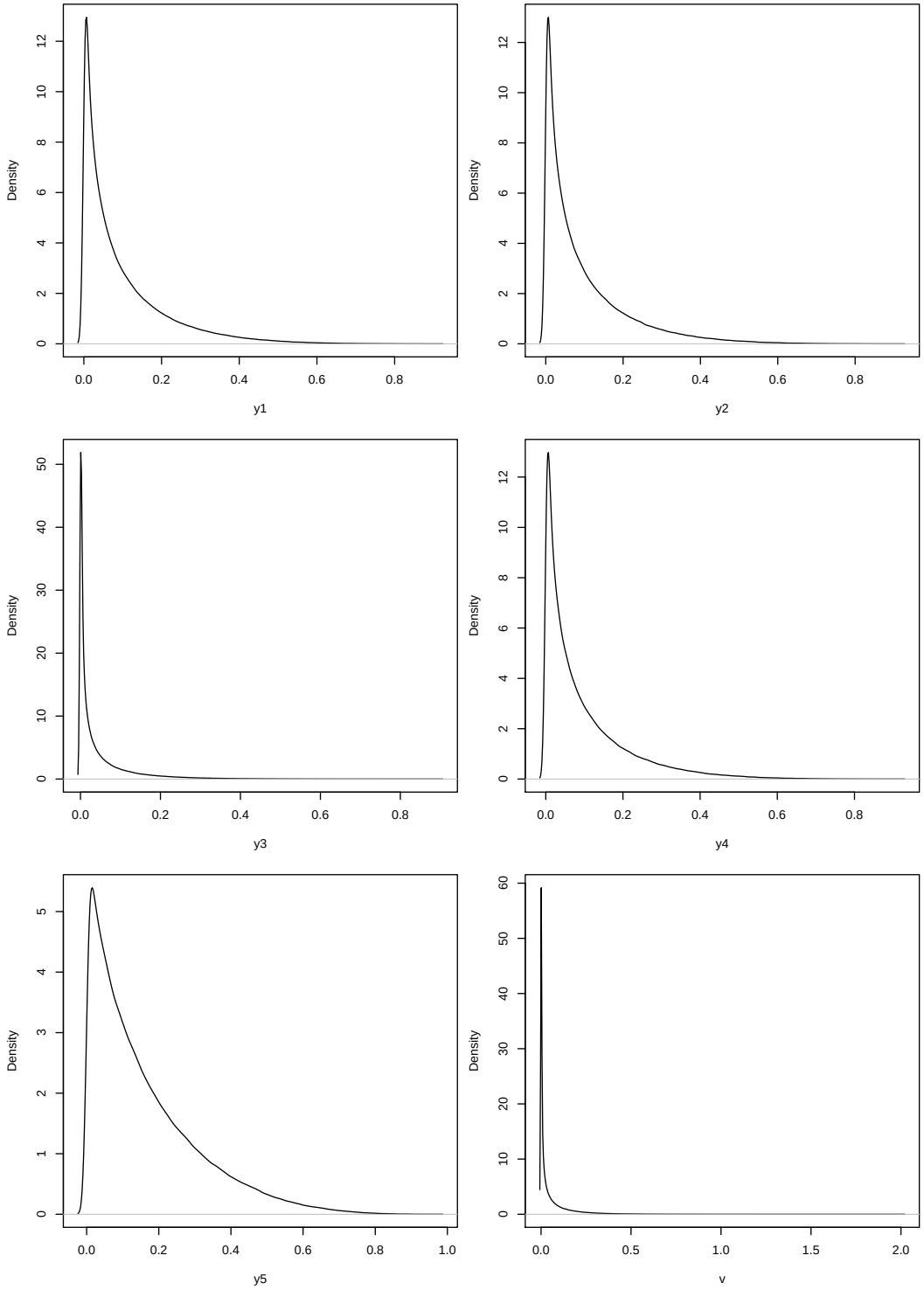


Figure 19: Prior densities for initial conditions $y_1^{(0)}, y_2^{(0)}, \ldots, y_5^{(0)}$ and variance parameter v.

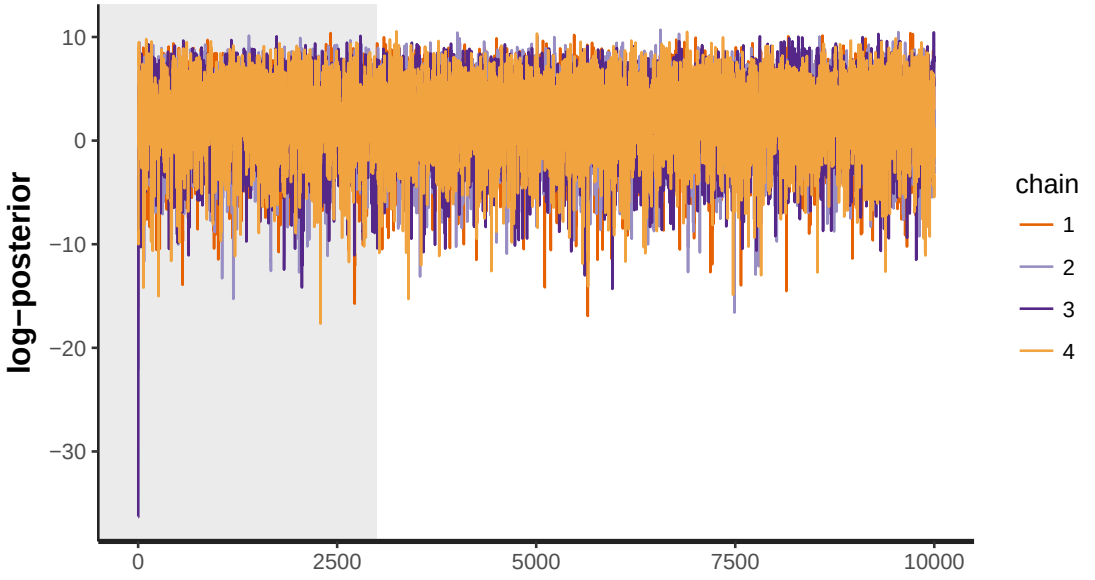


Figure 20: Trace of the log-posterior for Patient 10 (Remission). Warm-up region for the first 3000 iterations is shaded.

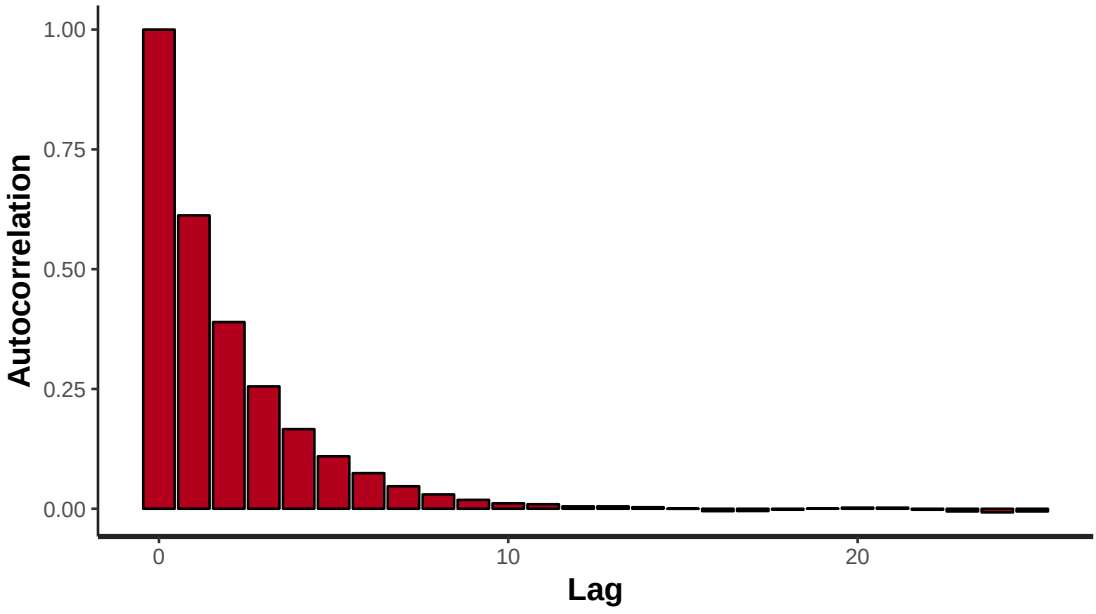


Figure 21: Average auto-correlation of the log-posterior across the four chains for Patient 10 (Remission).

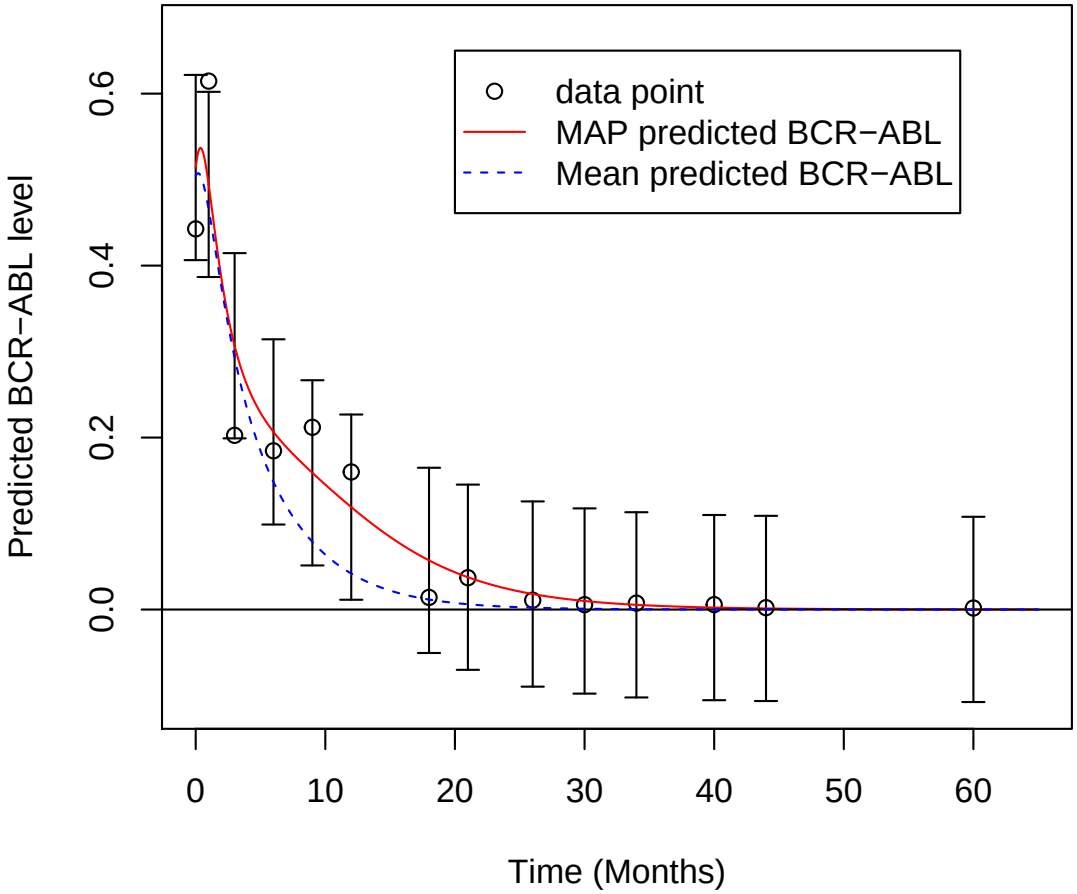


Figure 22: Approximate MAP BCR-ABL predictions for Patient 10 with confidence intervals for the predicted value at each data point.

8.1 R Code

```
1
2
3  #Various pakages to be installed and options to be set for stan
4  library(rstan)
5  rstan_options(auto_write = TRUE)
6  options(mc.cores = parallel::detectCores())
7  library(deSolve)
8  library(plot3D)
9  library(parallel)
10 library(MASS)
11 library(pdist)
12 library(bmk)
13 library(ggplot2)
14 library(ggdendro)
15 library(dendextend)
16 library(plotrix)
17
18 #Read in the data
19 data = read.csv("datatest.csv", header=T, stringsAsFactors=F)
20
21 #Set the data to be used
22 inputdata = setRemissionData(data, showPlot = T)
23 inputdata = setPatientData(data, 25, showPlot = F)
24
25 #set data for stan to use
26 standata = list(T = length(inputdata[,1]),
27                 R = inputdata$BCR.ABL1.ABL1....,
28                 t0 = 0,
29                 ts = inputdata$Month[-1],
30                 alpha = c(0.9,0.9,0.4,0.9,1.5))
31
32 #number of mcmc chains
33 n = 4
34
35 #Runs mcmc and saves in 'fit'
36 fit <- stan(file = 'MScDissertation/odemcmc.stan', data = standata,
37             iter = 30000, chains = n, warmup = 3000)
38
39 #Save the fit to file
40 #saveRDS(fit, file = "10Remission30000Normalfit.RData")
41
42
```

```
43 ######################################################
44 ##################   Functions   ####################
45 ######################################################
46
47 #Ode system for solving ode in r
48 desystem = function(t, state, parameters) {
49   with(as.list(c(state, parameters)), {
50     dx0 <- ax * x0 * (1 - x0 - x1 - x2 - y0 - y1) - bx * x0
51     dx1 <- bx * x0 + cx * x1 * (1 - x0 - x1 - x2 - y0 - y1) - dx * x1
52     dx2 <- dx * x1 - ex * x2
53     dy0 <- ay * y0 * (1 - x0 - x1 - x2 - y0 - y1) - by * y0
54     dy1 <- by * y0 - ey * y1
55     list(c(dx0, dx1, dx2, dy0, dy1))
56   })
57 }
58
59 #Takes a stan fit object and returns a list of the data
60 generateListData = function(stanfit = NULL, mcmcData = NULL, nruns = 1)
61 {
62   tempList = vector("list", nruns)
63
64   if(is.null(mcmcData))
65   {
66     la = as.array(stanfit)
67
68     for(i in seq_len(nruns))
69       tempList[[i]] = la[,i,]
70   }
71   else
72   {
73     tempList[[1]] = mcmcData
74   }
75   return(tempList)
76 }
77
78 #Generates various diagnostic plots for stan fit object 'mcmcfit'. It can plot a
       trace/density of theta/y0/v/logpost
79 generatediagnostics = function(mcmcfit = NULL, plotTrace = F, plotACF = F, plotDensity
       = F, plotTheta = F, ploty0 = F, plotv = F, plotlogpost = F)
80 {
81   print(mcmcfit, pars = c("theta", "y0", "v", "lp__"), probs=NULL)
82
83   thetaLabel = NULL
84   y0Label = NULL
```

```
85    vLabel = NULL
86    logpostLabel = NULL
87
88    if(plotTheta == T)
89      thetaLabel = "theta"
90    if(ploty0 == T)
91      y0Label = "y0"
92    if(plotv == T)
93      vLabel = "v"
94    if(plotlogpost == T)
95      logpostLabel = "lp__"
96
97    if(plotTrace == T)
98      print(stan_trace(mcmcfit, pars = c(thetaLabel, y0Label, vLabel, logpostLabel),
            inc_warmup = TRUE, nrow = 2) + xlim(0, 10000))
99
100   if(plotACF == T)
101     print(stan_ac(mcmcfit, pars = c(thetaLabel, y0Label, vLabel, logpostLabel))+
            ylab("Autocorrelation"))
102
103   if(plotDensity == T)
104     print(stan_dens(mcmcfit, pars = c(thetaLabel, y0Label, vLabel, logpostLabel),
            separate_chains = F, nrow=4,ncol=2))
105 }
106
107 #Calls the function 'generatediagnostics' for patient 'PatientID
108 #If plotsol = T then the predicted BCR-ABL is plotted also
109 PatientDiagnostics = function(PatientID = 1, plotTrace = F, plotACF = F, plotDensity =
      F, plotTheta = F, ploty0 = F, plotv = F, plotlogpost = F, plotsol = F)
110 {
111   samplefit = readRDS(paste("PatientData/",PatientID, "_10000fit.RData", sep=""))
112
113   if(plotsol == T)
114   {
115     data = read.csv("datatest.csv", header=T, stringsAsFactors=F)
116     inputdata = setPatientData(data, PatientID, showPlot = F)
117     sampleMcmcList = generateListData(samplefit, nruns = 4)
118     finalsolutionplot(mcmcList = sampleMcmcList, xdata = inputdata$Month, ydata =
          inputdata$BCR.ABL1.ABL1....)
119   }
120   generatediagnostics(mcmcfit = samplefit, plotTrace = plotTrace, plotACF = plotACF,
        plotDensity = plotDensity, plotTheta = plotTheta, ploty0 = ploty0, plotv =
        plotv, plotlogpost = plotlogpost)
121 }
```

```
122
123 #Calculates the total KS distance summed over theta parameter 'theta_ind'
124 Total_CDF_Distance = function(Patients = c(1,2), n = 4, theta_ind = 1:8)
125 {
126   samplefit1 = readRDS(paste("PatientData/",Patients[1], "_10000fit.RData", sep=""))
127   samplefit2 = readRDS(paste("PatientData/",Patients[2], "_10000fit.RData", sep=""))
128
129   data1 = extract(samplefit1)
130   data2 = extract(samplefit2)
131
132   mat1 = data1$theta
133   mat2 = data2$theta
134
135   totalDiff = 0
136   ind = seq(0,1,0.001)
137   for(k in theta_ind)
138   {
139     cdf1 = ecdf(mat1[,k])
140     cdf2 = ecdf(mat2[,k])
141     totalDiff = totalDiff + max(abs(cdf1(ind) - cdf2(ind)))
142   }
143   return(totalDiff)
144 }
145
146 #Calculates the approximate mean euclidean distance between samples from patients
      'Patients'
147 #If CI is TRUE, it instead calculates the width of the confidence interval
148 Average_Distance = function(Patients = c(1,2), num_samples = 100000, type = "mean", CI
      = F)
149 {
150   samplefit1 = readRDS(paste("PatientData/",Patients[1], "_10000fit.RData", sep=""))
151   samplefit2 = readRDS(paste("PatientData/",Patients[2], "_10000fit.RData", sep=""))
152
153   data1 = extract(samplefit1)
154   data2 = extract(samplefit2)
155
156   mat1 = data1$theta
157   mat2 = data2$theta
158
159   ind1 = sample(seq_len(dim(mat1)[1]), num_samples, replace = T)
160   ind2 = sample(seq_len(dim(mat2)[1]), num_samples, replace = T)
161
162   eucl = numeric(num_samples)
163   for(i in seq_len(num_samples))
```

```
164     eucl[i] = dist(rbind(mat1[ind1[i], ], mat2[ind2[i], ]))
165
166   if(CI == T)
167   {
168     sig = sd(eucl)
169     tsig = qt(1 - 0.05/2, df = num_samples - 1)
170     return(2 * tsig * sig / sqrt(num_samples))
171   }
172
173   else
174   {
175     if(type == "mean")
176       return(mean(eucl))
177
178     if(type == "max")
179       return(max(eucl))
180   }
181 }
182
183 #Calculates the distance matrix for one of the 3 defined distances
184 #avg_dist = T   => use mean distance between samples
185 #avg_dist = F   => use KS distance with theta parameter 'theta_ind'
186 DifferenceMatrix = function(PatientNumbers = 1:69, n = 4, theta_ind = 1:8, avg_dist =
      F, ci = F)
187 {
188   n = length(PatientNumbers)
189   grid = combn(PatientNumbers, 2)
190
191   no_cores = detectCores() - 1
192   cl = makeCluster(no_cores)
193   clusterExport(cl, "extract")
194   clusterExport(cl, "mdist")
195   clusterExport(cl, "best.col.perm")
196   clusterExport(cl, "pdist")
197   clusterExport(cl, "graph_from_adjacency_matrix")
198   clusterExport(cl, "set_vertex_attr")
199   clusterExport(cl, "maxmatching")
200
201   if(ci == T && avg_dist == F)
202     stop("Confidence interval length only for avg distance")
203   {
204     if(avg_dist == T)
205       temp = parApply(cl, grid, 2, Average_Distance, num_samples = 100000, type =
            "mean", CI = ci)
```

```
206
207       else
208         temp = parApply(cl, grid, 2, Total_CDF_Distance, n = 4, theta_ind = theta_ind)
209     }
210     stopCluster(cl)
211
212     mat = matrix(0, nrow = n, ncol = n)
213     mat[lower.tri(mat)] = temp
214     mat = mat + t(mat)
215     colnames(mat) = PatientNumbers
216     return(mat)
217   }
218
219   #Returns LDA fit object with 'num_samples' samples taken from each patient in
        'PatientIDs'
220   #Patients 'relapse_IDs' are categorised as relapse
221   PatientLDA = function(PatientIDs = 1:69, theta_ind = 1:8, y0_ind = 1:5, v_ind = 1,
        num_samples = 1000, relapse_IDs = c(7,25,47))
222   {
223     group = rep(as.numeric(PatientIDs %in% relapse_IDs) + 1, each = num_samples)
224     samples = SampleFromPost(PatientIDs = PatientIDs, theta_ind = theta_ind, y0_ind =
          y0_ind, v_ind = v_ind, num_samples = num_samples)
225
226     ldafit = lda(x = samples, grouping = group)
227     return(ldafit)
228   }
229
230   #Plots the mode predicted BCR-ABL. If ci=T then confience intervals are plotted
231   #meanSol = T   =>   the mean BCR-ABL at each time is plotted.  !!!! Long time to run
        !!!!
232   #indSol = T  =>  the mode solution for each chain is plotted
233   #input is a stan ouput list as returned by 'generateListData'
234   #xdata and ydata also need to be set and inputed as returned by 'setPatientData'
235   finalsolutionplot = function(mcmcList = NULL, xdata, ydata, meanSol = F, indSol = F, ci
        = F)
236   {
237     nruns = length(mcmcList)
238     mcmcData = NULL
239     if(indSol == T)
240     {
241       for(i in seq_len(nruns))
242       {
243         if(i == 1)
244           plot(xdata, ydata, ylim=c(0,0.8), xlim=c(0, max(xdata)+5), xlab = "Time
```

```
        (Months)", ylab = "Predicted BCR-ABL level")

  #Set parameters
  mcmcData = as.matrix(mcmcList[[i]])
  colnames(mcmcData) = NULL
  thetamcmc = mcmcData[,8:15]
  y0mcmc = mcmcData[,16:20]
  vmcmc = mcmcData[,1]
  zmcmc = mcmcData[,2]
  logpostmcmc = mcmcData[,21]

  if(meanSol == T)
  {
    mcmcLen = dim(mcmcData)[1]

    times = seq(0, max(xdata) + 5, by = 0.1)
    R = matrix(nrow = mcmcLen, ncol = length(times))
    for(j in seq_len(mcmcLen))
    {
      theta = thetamcmc[j,]
      y0 = y0mcmc[j,]

      parameters = c(ax = theta[1], bx = theta[2], cx = theta[3],
                     dx = theta[4], ex = theta[5], ay = theta[6],
                     by = theta[7], ey = theta[8])

      state = c(x0 = y0[1], x1 = y0[2], x2 = y0[3], y0 = y0[4], y1 = y0[5])

      out = ode(y = state, times = times, func = desystem, parms = parameters)

      R[j,] = out[,6]/(out[,6] + 2*out[,4])
    }
    meanR = apply(R,2,mean)
    lines(times, meanR, col=i+1)
  }

  else
  {
    max_index = which.max(logpostmcmc)

    theta = thetamcmc[max_index,]
    y0 = y0mcmc[max_index,]

    parameters = c(ax = theta[1], bx = theta[2], cx = theta[3],
```

```
288                         dx = theta[4], ex = theta[5], ay = theta[6],
289                         by = theta[7], ey = theta[8])
290
291         state = c(x0 = y0[1], x1 = y0[2], x2 = y0[3], y0 = y0[4], y1 = y0[5])
292
293         times = seq(0, max(xdata) + 5, by = 0.1)
294         out = ode(y = state, times = times, func = desystem, parms = parameters)
295
296         R = out[,6]/(out[,6] + 2*out[,4])
297         lines(out[,1], R, col=i+1)
298       }
299     }
300   }
301   else
302   {
303     for(i in seq_len(nruns))
304       mcmcData = rbind(mcmcData, as.matrix(mcmcList[[i]]))
305
306     colnames(mcmcData) = NULL
307     thetamcmc = mcmcData[,8:15]
308     y0mcmc = mcmcData[,16:20]
309     vmcmc = mcmcData[,1]
310     zmcmc = mcmcData[,2]
311     logpostmcmc = mcmcData[,21]
312
313     max_index = which.max(logpostmcmc)
314
315     theta = thetamcmc[max_index,]
316     y0 = y0mcmc[max_index,]
317     v = vmcmc[max_index]
318     sig = sqrt(v)
319
320
321     parameters = c(ax = theta[1], bx = theta[2], cx = theta[3],
322                    dx = theta[4], ex = theta[5], ay = theta[6],
323                    by = theta[7], ey = theta[8])
324
325     state = c(x0 = y0[1], x1 = y0[2], x2 = y0[3], y0 = y0[4], y1 = y0[5])
326
327     times = seq(0, max(xdata) + 5, by = 0.1)
328     out = ode(y = state, times = times, func = desystem, parms = parameters)
329
330     ind = match(xdata, times)
331
```

```
332     R = out[,6]/(out[,6] + 2*out[,4])
333
334     L = R[ind] - 1.96 * sig
335     U = R[ind] + 1.96 * sig
336
337     if(ci == T)
338     {
339       plotCI(xdata, R[ind], ui=U, li=L, xlim=c(0, max(xdata)+5), ylim=c(min(0,L),
              max(U)+0.05) , xlab = "Time (Months)", ylab = "Predicted BCR-ABL level",
              pch=NA)
340       points(xdata, ydata)
341
342       lines(out[,1], R, col=2)
343       abline(h=0)
344       legend(25,0.65, c("data point","predicted BCR-ABL"), pch = c(1,NA), lty = c(NA,
              1), col = c(1,2))
345     }
346     else
347     {
348       plot(xdata, ydata, xlim=c(0, max(xdata)+5), ylim=c(0, max(ydata)+0.05) , xlab =
              "Time (Months)", ylab = "Predicted BCR-ABL level")
349       lines(out[,1], R, col=2)
350     }
351   }
352 }
353
354 #Generates samples from the posterior for a given patient and return them in a matrix.
355 SampleFromPost = function(PatientIDs = 1:69, theta_ind = 1:8, y0_ind = 1:5, v_ind = 1,
      num_samples = 1000)
356 {
357   sampleMat = NULL
358   for(i in PatientIDs)
359   {
360     samplefit = readRDS(paste("PatientData/",i, "_10000fit.RData", sep=""))
361     data = extract(samplefit)
362     theta = data$theta
363     y0 = data$y0
364     v = data$v
365
366     samp_ind = sample(dim(theta)[1], num_samples)
367
368     if(is.null(v_ind))
369       PatientSample = cbind(theta[samp_ind, theta_ind], y0[samp_ind, y0_ind])
370     else
```

```
371         PatientSample = cbind(theta[samp_ind, thetaInd], y0[samp_ind, y0_ind],
                v[samp_ind])
372
373       sampleMat = rbind(sampleMat, PatientSample)
374     }
375     return(as.matrix(sampleMat))
376   }
377
378   #Function to plot BCR-ABL level plots of posterior samples. Similar input to
        'finalsolutionplot'
379   PlotSampleSolutions = function(mcmcList = NULL, xdata, ydata, numSamples = 50,
        plotIndSamples = F)
380   {
381     nruns = length(mcmcList)
382     mcmcData = NULL
383     for(i in seq_len(nruns))
384       mcmcData = rbind(mcmcData, as.matrix(mcmcList[[i]]))
385
386     colnames(mcmcData) = NULL
387     thetamcmc = mcmcData[,8:15]
388     y0mcmc = mcmcData[,16:20]
389     vmcmc = mcmcData[,1]
390     zmcmc = mcmcData[,2]
391     logpostmcmc = mcmcData[,21]
392
393     sampleInd = sample(seq_len(dim(mcmcData)[1]), numSamples)
394
395     tempList = vector("list", numSamples)
396
397     plot(xdata, ydata, ylim=c(0,0.8), xlim=c(0, max(xdata)+5), xlab = "Time (Months)",
          ylab = "Predicted BCR-ABL")
398
399     times = seq(0, max(xdata) + 5, by = 0.1)
400
401     for(i in seq_len(numSamples))
402     {
403       theta = thetamcmc[sampleInd[i],]
404       y0 = y0mcmc[sampleInd[i],]
405
406       parameters = c(ax = theta[1], bx = theta[2], cx = theta[3],
407                      dx = theta[4], ex = theta[5], ay = theta[6],
408                      by = theta[7], ey = theta[8])
409
410       state = c(x0 = y0[1], x1 = y0[2], x2 = y0[3], y0 = y0[4], y1 = y0[5])
```

```
411
412       out = ode(y = state, times = times, func = desystem, parms = parameters)
413
414       tempList[[i]] = out
415
416       R = out[,6]/(out[,6] + 2*out[,4])
417       lines(times, R, col=8, lwd = 1)
418     }
419     points(xdata, ydata)
420
421     if(plotIndSamples == T)
422     {
423       for(j in 1:5)
424       {
425         for(i in seq_len(numSamples))
426         {
427           tempout = tempList[[i]]
428           if(i == 1)
429             plot(tempout[,j+1], main = paste("Proportion of Variable",j), ylab =
                      paste("Variable",j), xlab = "Time (Months)",  ylim = c(0,1), type = 'l',
                      col=8, lwd=1)
430           else
431             lines(tempout[,j+1], col=8, lwd=1)
432         }
433       }
434     }
435   }
436
437   #Plots the individual solutions y1,...y5 to the system of ODEs
438   IndividualSolutionPlot = function(mcmcList = NULL, xdata, ydata)
439   {
440     nruns = length(mcmcList)
441
442     for(j in 1:5)
443     {
444       for(i in seq_len(nruns))
445       {
446         #Set parameters
447         mcmcData = as.matrix(mcmcList[[i]])
448         colnames(mcmcData) = NULL
449         thetamcmc = mcmcData[,8:15]
450         y0mcmc = mcmcData[,16:20]
451         vmcmc = mcmcData[,1]
452         zmcmc = mcmcData[,2]
```

```
453         logpostmcmc = mcmcData[,21]
454
455         max_index = which.max(logpostmcmc)
456
457         theta = thetamcmc[max_index,]
458         y0 = y0mcmc[max_index,]
459
460         parameters = c(ax = theta[1], bx = theta[2], cx = theta[3],
461                        dx = theta[4], ex = theta[5], ay = theta[6],
462                        by = theta[7], ey = theta[8])
463
464         state = c(x0 = y0[1], x1 = y0[2], x2 = y0[3], y0 = y0[4], y1 = y0[5])
465
466         times = seq(0, max(xdata) + 5, by = 0.1)
467         out = ode(y = state, times = times, func = desystem, parms = parameters)
468
469         if(i == 1)
470           plot(times, out[,j+1], ylim=c(0,1), xlim=c(0, max(xdata)+5),
                  ylab=paste("Proportion of Variable",j), main = paste("Trace of
                  Variable",j), type='l')
471         else
472           lines(times, out[,j+1], col=i+1)
473       }
474     }
475   }
476
477   #Sets the patient data for a given patient
478   setPatientData = function(fulldata, PatientNumber = 25, showPlot = F)
479   {
480     tempdata = fulldata
481     tempdata[,3] = as.numeric(tempdata[,3])
482     tempdata = na.omit(tempdata)
483
484     tempdata[,3] = tempdata[,3]/100
485     ind = which(tempdata[,1] == PatientNumber)
486
487     PatientData = tempdata[ind,]
488
489     if(showPlot == T)
490       plot(PatientData[,2], PatientData[,3], ylim=c(0,1), xlab = "Time from beginning of
              treatment (Months)", ylab = "BCR-ABL level")
491
492     return(PatientData)
493   }
```

```
494
495 #Plots the data for given patients 'PatientInd'
496 #If overlay = T, then all patient in PatientInd are plotted in one plot
497 dataPlot = function(fulldata, PatientInd = 25, overlay = F, type = 'l')
498 {
499   tempdata = fulldata
500   tempdata[,3] = as.numeric(tempdata[,3])
501   tempdata = na.omit(tempdata)
502
503   tempdata[,3] = tempdata[,3]/100
504   count = 1
505   for(PatientNumber in PatientInd)
506   {
507     ind = which(tempdata[,1] == PatientNumber)
508
509     if(type != 'l')
510       type = 'p'
511
512     PatientData = tempdata[ind,]
513     if(overlay == T)
514     {
515       if(count == 1)
516         plot(PatientData[,2], PatientData[,3], ylim=c(0,1), xlim = c(0,60), xlab =
              "Time from beginning of treatment (Months)", ylab = "BCR-ABL level", type
              = type, col = count)
517       else
518         lines(PatientData[,2], PatientData[,3], col = count)
519     }
520     else
521       plot(PatientData[,2], PatientData[,3], ylim=c(0,1), xlab = "Time (Months)", ylab
            = "BCR-ABL level", type = type)
522     count = count + 1
523   }
524 }
525
526 #Samples from the prior.
527 #plotDensities = T   =>   Density plots of the priors are returned
528 #plotTrace = T   =>   prior BCR-ABL solution plots are returned
529 #alpha is the coefficient for the unscaled Dirichlet in the prior
530 priorSampling = function(n = 10, plotDensities = T, plotTraces = F, alpha =
      c(1,1,1,1,1))
531 {
532   z = runif(n)
533   ytemp = rdirichlet(n, alpha)
```

```
534
535   theta = matrix(runif(8*n), nrow = n)
536
537   y = z * ytemp
538
539   R0 = y[,5]/(y[,5] + 2*y[,3])
540
541   print(mean(R0))
542   finalVals = numeric(n)
543
544   if(plotDensities == T)
545   {
546     par(mfrow = c(4,2))
547     plot(density(theta[,1]), main="Prior density of theta 1")
548     plot(density(theta[,2]), main="Prior density of theta 2")
549     plot(density(theta[,3]), main="Prior density of theta 3")
550     plot(density(theta[,4]), main="Prior density of theta 4")
551     plot(density(theta[,5]), main="Prior density of theta 5")
552     plot(density(theta[,6]), main="Prior density of theta 6")
553     plot(density(theta[,7]), main="Prior density of theta 7")
554     plot(density(theta[,8]), main="Prior density of theta 8")
555     par(mfrow = c(1,1))
556     plot(density(R0), xlab = "R0")
557     plot(density(z), main="Prior density of z")
558     par(mfrow = c(3,2))
559     plot(density(y[,1]), main = "", xlab = "y1")
560     plot(density(y[,2]), main = "", xlab = "y2")
561     plot(density(y[,3]), main = "", xlab = "y3")
562     plot(density(y[,4]), main = "", xlab = "y4")
563     plot(density(y[,5]), main = "", xlab = "y5")
564   }
565
566   if(plotTraces == T)
567   {
568     times = seq(0, 65, by = 0.1)
569     L = length(times)
570     R = matrix(nrow=n,ncol=length(times))
571     tempList = vector("list", n)
572     for(i in seq_len(n))
573     {
574
575       parameters = c(ax = theta[i,1], bx = theta[i,2], cx = theta[i,3],
576                      dx = theta[i,4], ex = theta[i,5], ay = theta[i,6],
577                      by = theta[i,7], ey = theta[i,8])
```

```
578
579       state = c(x0 = y[i,1], x1 = y[i,2], x2 = y[i,3], y0 = y[i,4], y1 = y[i,5])
580
581       out = ode(y = state, times = times, func = desystem, parms = parameters)
582       tempList[[i]] = out
583     }
584     count = 0
585     for(i in seq_len(n))
586     {
587       out = tempList[[i]]
588       R = out[,6]/(out[,6] + 2*out[,4])
589       finalVals[i] = R[L]
590       if(R[2] < R[1])
591         count = count + 1
592       if(i == 1)
593         plot(times, R, xlab="Time (Months)", ylab="BCR-ABL level", ylim = c(0,1),
              type='l', col = i)
594       else
595         lines(times, R, col = i)
596     }
597     print(count/n)
598     for(i in seq_len(n))
599     {
600       out = tempList[[i]]
601       if(i == 1)
602         plot(times, out[,2], xlab="time", ylab="x0", ylim = c(0,1), col=i, type='l')
603       else
604         lines(times, out[,2], col = i)
605     }
606     for(i in seq_len(n))
607     {
608       out = tempList[[i]]
609       if(i == 1)
610         plot(times, out[,3], xlab="time", ylab="x1", ylim = c(0,1), type = 'l', col=i)
611       else
612         lines(times, out[,3], col = i)
613     }
614     for(i in seq_len(n))
615     {
616       out = tempList[[i]]
617       if(i == 1)
618         plot(times, out[,4], xlab="time", ylab="x2", ylim = c(0,1), type='l', col=i)
619       else
620         lines(times, out[,4], col = i)
```

```
621     }
622     for(i in seq_len(n))
623     {
624       out = tempList[[i]]
625       if(i == 1)
626         plot(times, out[,5], xlab="time", ylab="y0", ylim = c(0,1), type='l', col=i)
627       else
628         lines(times, out[,5], col = i)
629     }
630     for(i in seq_len(n))
631     {
632       out = tempList[[i]]
633       if(i == 1)
634         plot(times, out[,6], xlab="time", ylab="y1", ylim = c(0,1), type='l', col=i)
635       else
636         lines(times, out[,6], col = i)
637     }
638
639     print("Percent less than 0.05:")
640     print(sum(finalVals < 0.05)*100/n)
641   }
642 }
```

8.2 Stan Code

```
1  functions {
2       real[] mlode(real t, real[] y, real[] theta, real[] x_r, int[] x_i) {
3           real dydt[5];
4           dydt[1] = theta[1] * y[1] * (1-y[1]-y[2]-y[3]-y[4]-y[5]) - theta[2] * y[1];
5           dydt[2] = theta[2] * y[1] + theta[3] * y[2] * (1-y[1]-y[2]-y[3]-y[4]-y[5]) -
                theta[4] * y[2];
6       dydt[3] = theta[4] * y[2] - theta[5] * y[3];
7       dydt[4] = theta[6] * y[4] * (1-y[1]-y[2]-y[3]-y[4]-y[5]) - theta[7] * y[4];
8       dydt[5] = theta[7] * y[4] - theta[8] * y[5];
9           return dydt;
10    }
11 }
12 data {
13     int<lower=1> T;
14         real R[T];
15         real t0;
16         real ts[T-1];
17     vector<lower = 0>[5] alpha;
18 }
```

```
19
20 transformed data {
21         real x_r[0];
22         int x_i[0];
23 }
24
25 parameters {
26         real<lower=0> v;
27     real<lower=0,upper=1> z;
28     simplex[5] y0temp;
29         real<lower=0,upper=1> theta[8];
30 }
31
32 transformed parameters {
33     vector[5] y0;
34     y0 = z * y0temp;
35 }
36
37 model {
38         real y_hat[T-1,5];
39         real R_hat[T];
40
41         v ~ gamma(0.25,5);
42         theta ~ uniform(0,1);
43     z ~ uniform(0,1);
44     y0temp ~ dirichlet(alpha);
45     R_hat[1] = y0[5]/(y0[5] + 2 * y0[3]);
46     R[1] ~ normal(R_hat[1], sqrt(v));
47         y_hat = integrate_ode_rk45(mlode, to_array_1d(y0), t0, ts, theta, x_r, x_i);
48
49         for (t in 2:T) {
50     R_hat[t] = y_hat[t-1,5]/(y_hat[t-1,5] + 2 * y_hat[t-1,3]);
51     R[t] ~ normal(R_hat[t], sqrt(v));
52         }
53 }
```