Bayesian Markov Chain Monte Carlo Methods in General Insurance

Fernanda Chaves Pereira

Thesis submitted to the degree of
Doctor of Philosophy in Actuarial Science

City University
School of Mathematics
Department of Actuarial Science and Statistics

London/England

October/2000

to Zilva,

"(...)
Qual foi a nossa vitória?
Um livro,
um livro repleto
de contatos humanos,
de camisas,
um livro
sem solidão, com homens
e ferramentas,
um livro
é a vitória.
(...)

Pablo Neruda"

Table of contents

List of tables and illustrations

Figures

Tables

Acknowledgement

This thesis is a work of four years and many persons helped me throughout this process. I would like to thank them all (Stefano, Helena, Marcia, Zoë, Bela, Luciana, Victoria, Mimia) for their support.

Specially, this work would never be possible without the help of three main "sets":

My supervisor, Professor Richard Verrall, who so wisely guided me to acquire the knowledge to produce this thesis. My sincere gratitude for his prompt availability;

My sponsors, Fundação Escola Nacional de Seguros (Funenseg), Brazilian institution that granted me with the financial support, making it viable for me to finalise my studies. More specifically I would like to thank Ms Silvia Klein, who related to me in their behalf and always believed on my project. I do believe in angels;

My mom, dad and sister, wonderful people that are my basis for everything I do. Thank you so much for your support and love.

Declaration

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Abstract

Using a modern Bayesian implementation technique, this thesis shows two applications of individual modelling in real data sets. The simulation approach is adopted, with a Markov chain Monte Carlo (MCMC) method – Reversible Jump MCMC – as the core of the thesis. This technique allows the definition of a model with few underlying assumptions and based on a changing-dimension parameter set.

Its first application is in automobile insurance, where the model estimates at the same time the number of groups and their respective risk parameters in order to have a better description of the analysed data. Since all this process is based on a continuous piecewise distribution, no obvious analytical solution for this type of problem is available. RJMCMC is the only stochastic simulation that allows this change of dimensionality.

The flexibility of this model is explored in the second application presented in this thesis. In this new case, the aim is not to define the number of groups, but to use a limited number assumptions to model the claim reserves in a dental insurance coverage.

Both applications model frequency and severity separately and apply the grouping technique to both discrete and continuous variables.

Chapter 1

Introduction

There are many models in actuarial science for estimating reserves, premium rating and many other risk assessment problems. In general insurance most of these models are based on data aggregated in some way, as is also the case in reinsurance, life assurance and pensions. Beside aggregating, most of these models also use quite limiting assumptions. There are few models that use a process based approach, looking at individual information underlying behaviour.

As statistical techniques develop, they facilitate the application of refined models and this thesis explores these new possibilities. In the area of Bayesian statistics, a new stochastic simulation technique has been recently developed – Reversible jump Markov chain Monte Carlo (RJMCMC). This technique has the ability of implementing models where the length of the parameter vector is not fixed and it will be fully defined in section 2.2.3.

This implementation technique does not have yet any published application in general insurance that the author is aware of. Thus, the core of this thesis is the application of Bayesian statistics modelling and RJMCMC to two risk assessment problems, where the data are not aggregated, but kept at an individual level. Also, besides the individual data structure, the models do not have a fixed number of parameters.

The first application on this thesis, in Chapter 5, is on age grouping, when age is measured in days and considered continuous. The data are on automobile insurance and were also analysed in Verrall and Yakoubov (1999), where the continuity assumption was dropped and only full years were considered. The second application is to dental claims data, in Chapter 6,

where the main objective is the definition of IBNR (Incurred But Not Reported) and RBNS (Reported But Not Settled) reserves. The data are on dental insurance and were also analysed in Arjas and Haastrup (1996), but implemented via Arjas and Gasbarra (1994) instead of RJMCMC.

1.1 Bibliographical review

Bayes theorem was first introduced in 1763, but it was in the last 20 years that a wider range of its applications has occurred. The growth in applications in this area is due to the development of stochastic simulation techniques of the Markov chain Monte Carlo (MCMC) class, which allow the applications of most of the Bayesian statistical models.

Metropolis-Hastings (Metropolis et all (1953); Hastings (1970)) were the pioneers on MCMC, and the methodology was further developed to Gibbs sampling (Geman and Geman (1984); Gelfand and Smith (1990)), which is now the widely used technique of this class of simulation. A good textbook on these implementation techniques is Gamerman (1997). Reversible jump Markov chain Monte Carlo (RJMCMC) (Green (1995)) is a newly developed technique and it is applied to some real data in Green and Richardson (1997), Dellaportas, Karlis and Xekalaki(1997) and Denison, Mallick and Smith (1998).

Actuaries were among the first practitioners to use the Bayesian philosophy, which is the basis of credibility theory published by Mowbray (1914). Although this theory is fully based on Bayes theorem, it was not until 1950 that a more theoretical approach exploring its origin was used (Bailey (1950)). Credibility theory is the main actuarial technique based on Bayesian statistics and many papers were published throughout the years (Bühlmann (1967), Bühlmann and Straub (1970), Bühlmann and Jewell (1987), Jewell (1974, 1975, 1976), Kremer(1982), Sundt (1982, 1983, 1987), Pereira(1998) are some examples).

Including not only credibility theory but also some other applications, Klugman (1992) and Liu, Makov and Smith (1996) give a review of Bayesian models in actuarial science. With the development of Bayesian statistics some new applications have appeared in actuarial science in claims reserving (Verrall (1990), Charissi (1997), Dellaportas and Ntzoufras (1997)) and graduation (Carlin (1992), Kouyoumoutzis (1998)). These models included more information about the value and structure of the parameters via the

prior distribution, and many of them use stochastic simulation methods. Boskov and Verrall (1994) use a two dimensional smoothing model to estimate a premium by postcode area, bringing a more elaborate model to the ratemaking process.

In parallel to these cited applications, some new ideas about the processes involved in claim development have appeared. The works of Jewell (1989, 1990) review the claim process, with the earlier paper considering continuous time, but not including the values of payment. This idea is reformulated in Norberg (1992, 1993, 1999), where the full process of occurrence, delay of reporting and partial payment of a claim are considered in the model. The process proposed is these papers is applied in Arjas and Haastrup (1996).

1.2 Thesis outline

The first chapter is the introduction. Chapter 2 includes a brief review of Bayesian statistics and the implementation techniques used in this work. Also in Chapter 2, Markov chain Monte Carlo (MCMC) simulation techniques are explained, including Metropolis-Hastings, Gibbs Sampling and RJMCMC.

In the third chapter a review of Bayesian modelling in actuarial science is presented. This chapter is a revised version of Pereira (1999) which was awarded the Highly Commended award on the Brian Hey Prize competition 1999, held by the English Institute of Actuaries. Sections on credibility, claims reserving and graduation are presented, showing the development of models in this area with the advent of stochastic simulation (mainly applications using Gibbs sampling). Finally in Chapter 3 a section about modern Bayesian theory is given.

Chapter 4 contains the theoretical framework for the thesis, with the models used in this work. This chapter also includes the explanation of the basic model, which is the basis of the applications presented in this thesis. This model is built straight from the Bayesian philosophy. Since the model looks at the development of the claim it can be used to monitor different aspects in this process. In this work the same model will be applied in two different types of risk assessment and its applications will be seen in Chapters 5 and 6.

The implementation technique presented in Chapter 2 and the basic model considered in Chapter 4 are put together and applied to two real data

sets in different types of risk assessment. The first application is in aggregation (or transformation into factors) of a continuous variable and is fully explained in Chapter 5. Chapter 6 is the application to claims reserving, an extension of the work from Arjas and Haastrup (1996) (also explored in Haastrup (1997)), where the claims reserves are obtained via data using individual information. Chapter 7 is the thesis conclusion, presenting some ideas of future research as well.

Chapter 2

Bayesian framework

Bayesian theory has been one of the most discussed and developed branches of statistics over recent years. There have been an enormous number of papers published by a large number of statistical researchers and practitioners (see Gilks *et al* (1996) for some references and applications of Metropolis-Hastings and Gibbs sampling). These recent developments are mainly due to, firstly, the failure of classical statistical methods to give solutions to many problems and, secondly, to the computer developments that have made it easier to perform calculations by simulation.

The models used in this thesis are fully based on Bayesian theory, and for this reason this second chapter presents an overview from basic concepts to new techniques. Most of the following sections can be found in DeGroot(1986) and Gamerman(1997). In this chapter no application to actuarial science is presented, but only the theoretical statistical background.

The first section of this chapter reviews some of the foundations and basic results of Bayesian statistics. Section 2.2 explain the Markov chain Monte Carlo class of simulation, including more specific sections for Metropolis-Hastings (2.2.1), Gibbs Sampling (2.2.2) and Reversible jump MCMC (2.2.3). In each of these sections an algorithm is presented and in the section referred to RJMCMC a bibliographical review is also included. In the last section a chapter conclusion is presented.

2.1 Basic concepts in Bayesian theory

As is well known, probability theory is the foundation for statistics, with the differences in the interpretation of the term probability also defining the respective differences in statistical theories. The subjective interpretation of probability, the basis of Bayesian theory, states that the probability that an analyst assigns to a possible outcome of a certain experiment represents his own judgement of the likelihood that a specific outcome will be obtained. This judgement is based on the analyst's beliefs and information about the experiment. As a contrast, frequency statistics, for example, do not include formally this judgement but only the information received from the observation set itself.

Bringing these interpretations to the inference problem of estimating a specific parameter, Bayesian statistics differs clearly from the others. In classical and frequency statistics the analyst is searching for a best estimator of a parameter that has a true fixed value, which is unknown at the moment of modelling. In Bayesian statistics the analyst does not believe in this true fixed value, but in a range represented by the previous information that he has and includes via the prior distribution.

The recognition of the subjective interpretation of probability has the effect of emphasising some of the subjective aspects of science. It also defines a formal way of including judgement in the model. This subjective information is included in the model by defining a prior distribution for the unknown parameters.

Bayes theorem is the formal mechanism of incorporating prior information into the modelling. The theorem mixes the prior subjective information with that observed in the experiment, producing a posterior distribution. This last distribution is considered as an update of the previous judgement (prior) through the data observed (likelihood).

More formally, Bayes theorem is defined as follows. Consider a process in which observations (Y is the vector of observations) are to be taken from a distribution for which the probability density function is $p(Y \mid \theta)$, where θ is a set of unknown parameters. Before any observation is made, the analyst would include all his previous information and judgements of θ in a prior distribution $p(\theta)$, that would be combined with the observations to give a posterior distribution $p(\theta \mid Y)$ in the following way:

$$p(\theta \mid Y) \propto p(Y \mid \theta) p(\theta) \qquad (2.1)$$

The complete definition of Bayes theorem is given by the normalisation constant, which is equal to the predictive distribution defined by the following formula (integrals are of the Riemann type throughout the thesis):

$$p(Y^*) = \int p(Y|\theta)\, p(\theta)\, d\theta \qquad (2.2)$$

This distribution is used to perform predictions (explaining its nomenclature) about future samples of Y (denoted by Y^*), while the posterior distribution is used to make inference about the parameter θ.

It can be difficult to define a prior distribution and even harder to justify the choice. In fact, it is one of the most controversial elements in Bayesian statistics. Many ways of defining this distribution have been proposed, but ideally the prior distribution and the values of its hyper-parameters should be chosen independently of the data and, together with the observational distribution, complete the model. Then, in a perfect world the posterior and predictive distributions are fully defined as well. This would be a Pure Bayes approach, and any estimation of θ is defined only after having all information to hand. If a point estimate is the analyst's objective, for instance, then a loss function is chosen and the respective value (mode, mean, as examples) is calculated from the posterior distribution. A fully defined posterior distribution contains all the information about the parameter, and given this completeness of information, it is argued that Bayesian theory gives a better description of the parameter.

Unfortunately it is usual that in Bayesian statistics much more calculation is needed to achieve an estimator of θ, and in many cases it is not even possible to derive the analytical form of the posterior distribution. In order to avoid such difficulties many analysts would use the concept of the conjugate prior distribution. This type of prior distribution relates to the structure of the observational distribution and reapplies this to the prior distribution. In this way it facilitates the calculations.

If the analyst does not want to include prior information, but does want to use a Bayesian approach, it is possible to use a non-informative prior. There are many ways of defining a non-informative prior where the main objective is to give as little prior information about the parameters as possible (Gamerman and Migon (1993), DeGroot (1986)). Some ways to use non-informative prior are the following:

- Jeffrey's information technique;
- a uniform prior distribution;
- a prior distribution with a large value for the coefficient of variation;

- parameters of the prior distribution based on a set of data.

The use of observed data in the prior is called empirical Bayes estimation. This gives greater weight to the data than the pure Bayesian approach, and the prior distribution has a frequency interpretation instead. The term empirical Bayes is sometimes also used for some of the non-informative prior procedures, a term defined by Klugman (1992) as "any method that attempts to shortcut one of the Bayesian steps."

Theoretically, a prior distribution could be included for all the parameters that are unknown in a model, so that any model could be represented in a Bayesian way. However, this often leads to intractable problems (mainly integrals without solutions). So the main limitation of Bayesian theory is the difficulty, and in many cases the impossibility, of analytically solving the required equations.

In such cases an approximation for the posterior of θ is necessary. To overcome those problems many approximations techniques can be used, such as Gauss Hermite, or Gaussian quadrature. But of greater importance and application are Markov chain Monte Carlo (MCMC) methods. In the last few decades this stochastic simulation technique has been developed in order to solve this problem and to obtain estimates of the posterior distribution. These techniques were turning points for Bayesian theory, making it possible to apply many models that otherwise could not have been applied. The essence of MCMC methods is that by sampling from specific simple distributions (derived from the combination of the likelihood and prior distributions), a sample from the posterior distribution is obtained in an asymptotic way. The next section explains MCMC methods in more details.

2.2 Markov chain Monte Carlo methods

Markov chain Monte Carlo (MCMC) is a class of stochastic simulation techniques for the derivation of the posterior distribution of a specific vector of parameters θ. Stochastic simulation means that instead of deriving the posterior distribution analytically a large sample of the set of parameters is obtained, where the sample is random and based on the characteristics of the model. With this large sample of θ many summary statistics can be calculated in order to draw conclusions about the posterior distribution, which it may not be possible to obtain in analytical form. The MCMC technique is particularly

useful when the parameter vector has a high dimension or when the analytical solution for the posterior distribution is not available.

The essence of this method is that, in an asymptotic way, it is possible to sample from the posterior distribution, and if a large sample is taken, a good approximation to the required posterior distribution is achieved. Thus, the analytical form is replaced by a large sample from the posterior distribution. Many summarising values can be calculated from this sampled posterior distribution. For instance, the mean could be approximated by the arithmetic average of the sample, and variance by the square of the sample standard deviation. Credibility intervals, the mode or a histogram are also easily evaluated.

This class of simulation has been used in a large number and wide range of applications, and has been found to be very powerful. In order to understand the method, suppose it is desired to know the posterior distribution $p(\theta^{(k)} \mid Y)$ related to the set of parameters $\theta^{(k)}=(\theta_1,\ldots,\theta_{n_k})$ with a fixed dimension n_k dependent on a constant k. In order to simplify notation, the superscript is dropped ($\theta=\theta^{(k)}$) and the posterior distributions, which are conditional on Y, are indicated by π. So $\pi(\theta) = p(\theta^{(k)} \mid Y)$ from now on.

MCMC is a recursive method, and in this case an initial point has to be defined; say that $\theta^0 = (\theta_1^0,\ldots,\theta_{n_k}^0)$. It is possible to define a recursive simulation for the set of parameters, where the new value θ^1 only depends on the previous value θ^0. The next value θ^1 is obtained randomly from the conditional distribution $\pi^*(\theta^1 \mid \theta^0)$. Given θ^1, once more the same distribution is used and θ^2 is sampled from $\pi^*(\theta^2 \mid \theta^1)$.

The distribution $\pi^*(\theta^i \mid \theta^{i-1})$ is the same for all $i=1,2,\ldots,N$, which guarantees a homogeneous chain in the procedure. The maintained structure of this transition distribution is the characteristic that guarantees that the distribution of θ^N given θ^0 converges as $N \to \infty$ to a limiting distribution $\pi^*(\theta)$, independent of θ^0 and only dependent on the transition density π^*. In addition, the Markovian property is observed because whenever it is necessary to define the distribution of θ^N given all the previous values $\theta^0,\theta^1,\ldots,\theta^{N-1}$, it is dependent only on θ^{N-1} and defined as $\pi^*(\theta^N \mid \theta^{N-1})$. Hence, this sequence forms a homogeneous Markov chain.

This is a powerful result and there are many rules in order to define a suitable transition density π^*. Independently of the chosen format, this

distribution should be easy to sample from. In this work, such a distribution is defined by generating each component of the set of parameters $\theta^{(k)}$ separately. This means that an updating of single components is used, which facilitates calculation since only one-dimensional sampling distributions are used.

When performing the MCMC simulation, there are some points that should be carefully taken into consideration, because despite the theoretical results ensuring the convergence, there is no absolute rule that assures when convergence is achieved. Usually it is easier to state that a convergence has not been achieved than the opposite. For instance, it must be checked whether the simulation is mixing well or, in other words, if the simulation procedure is visiting points over the whole range of possible values for θ. It should also be considered how large the sample should be, and whether the initial point, from which the simulation starts has a large influence. Among many other issues, the moment when convergence to the true distribution of θ is achieved should also be monitored. These rules are given in practice when analysing the results of the applications in Chapters 5 and 6.

All these features can make the technique difficult to apply, and, even worse, perhaps dangerous to use. This happens because once all the necessary procedures to start the simulation are ready, a sample of θ can always be obtained. This, however, does not mean that it is representative of the posterior distribution. In order to be sure that the sample does not have any deviation from the posterior distribution, the tests listed above have to be performed.

Meanwhile the analyst has to define two quantities. The first one is the starting value θ^{0}. This value should not influence the results, but could influence the number of samples taken before achieving convergence. In each chain the updating order of the parameters $\theta^{(k)}=(\theta_{1},...,\theta_{n_k})$ should also be defined.

The next quantity to determine is the value *M*, called the number of burn-in steps, such that subsequent samples can be assumed to be taken from the limiting distribution $\pi(\theta)$. There is no exact answer for this question, and the solution adopted here was to observe the trace of the sample (the plot of the sample value over time) until stability was observed. It was also chosen to use only one large sample starting from a specific value, similar to the one proposed by Geyer (1992), and to get as many samples as desired from the results θ^{m} for $m > M$.

Simulation deals with missing values in a very straightforward way. These values are treated as variables, in the same way as the parameters. So, in each iteration, a value for the missing value is also calculated and inference is carried out as usual. This is data augmentation, originally proposed by Tanner and Wong (1987). Suppose the data is $Y=(Y^m, Y^o)$, where m stands for missing data and o for the observed part. The method is simply to notice that it may be possible to sample by MCMC methods much more simply and efficiently from a distribution $p(\theta^{(k)}, Y^m \mid Y^o)$ than from $p(\theta^{(k)} \mid Y^o)$. Data augmentation is further explored later in the Gibbs sampling subsection 2.2.1.

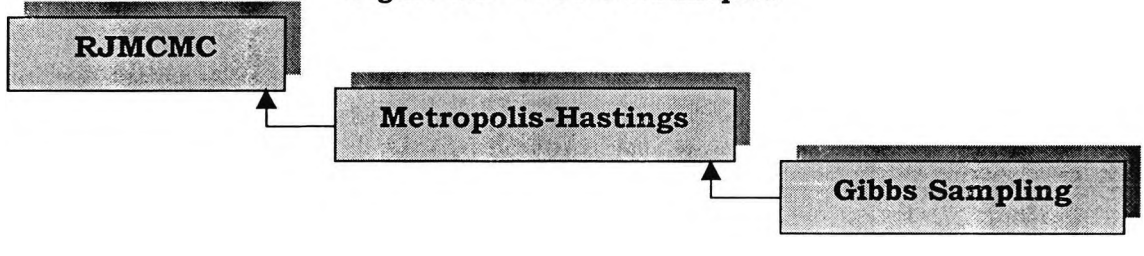
When n_k is fixed there are many types of MCMC techniques but the main ones are Metropolis-Hastings (subsection 2.2.2) and its special case, Gibbs sampling (subsection 2.2.1). They differ in the way the transition density π^* is defined, with Metropolis-Hastings being the more general one. In this thesis the applications generate each component of the set of parameters $\theta^{(k)}$ separately. In this way all the definitions are based on a single component.

When n_k is not fixed another technique called Reversible Jump Markov chain Monte Carlo (RJMCMC) is used. This is an extension of Metropolis-Hastings, defining a general framework for the change of parameter dimensionality problems, also interpreted as moving among models with different numbers of parameters.

Reversible jump is one among a few algorithms using MCMC that allow such changes. Two other works in the same direction could be cited. The first one is the work by Arjas and Gasbarra (1994) where an algorithm was built for the problem of specifying the hazard rate in a survival data problem. Their work is the basis for a part of the algorithm in this thesis, explained later in Chapter 5. The second work is the one developed by Grenander and Miller (1991, 1994), using jump-diffusions. Such a technique is not used in this thesis but further details can be found in Phillips and Smith (1995) which contains some examples, including image restoration.

Gibbs sampling, Metropolis-Hastings and RJMCMC will be fully explained in the following subsections. The simple scheme on figure 2.1 shows their connection.

Figure 2.1 – MCMC techniques



2.2.1 Gibbs Sampling

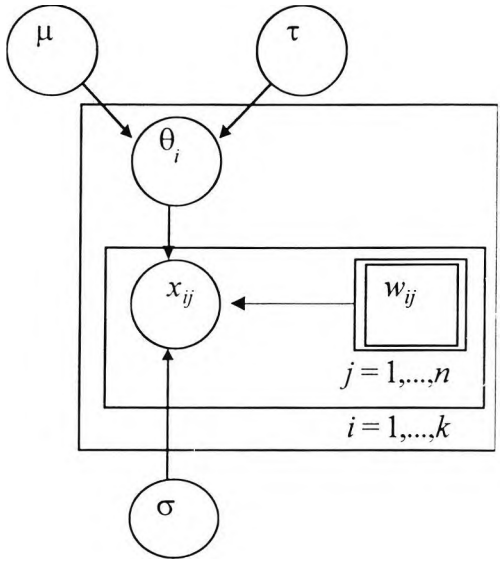
Gibbs sampling is a MCMC method for when π^* is defined based on any combination of the components of $\theta^{(k)}=(\theta_1,\ldots,\theta_{n_k})$, using their complete conditional distribution (when the parameters that are not being sampled are held fixed). The Markov chain has a dependency on the previous state governed by the complete conditional distribution. As an extreme case, if all the components of $\theta^{(k)}$ are updated at once, it means that no complete conditional distribution is used (there is no parameter left to be fixed) and in this case the whole posterior can be calculated, with no need for the approximation. The approach used here is to consider a single component, which means that each component θ_i for $i=1,\ldots,n_k$ is updated separately.

This method of stochastic simulation is one of the most powerful among MCMC methods. This is because the definition of π^* is governed only by the model, which is in contrast to the techniques reviewed next where an external distribution has to be defined to perform the implementation. It means that a sample from the exact distribution is taken with no need of performing any acceptance test, keeping all samples.

WinBUGS (Best, Spiegelhalter and Thomas (1998)) is a software package that implements Gibbs sampling. It is the newest version of BUGS (Bayesian inference Using Gibbs Sampling) which was first made available in 1992. This software works under Microsoft Windows®, which makes it easier to manipulate. Many useful tools for analysis are already included, and this helps to check if the simulation follows the rules cited here in section 2.2. There is also a set of software called CODA that produces some tests to check whether the simulation can be regarded as representative of the posterior distribution.

Directed acyclic graph (DAG) models are the basis of WinBUGS. Such schemes are often used in Bayesian analysis to give a better understanding of the models, particularly when the dependencies between the data and the parameters are complex. In order to illustrate this technique, figure 2.2 shows as example of a DAG.

Figure 2.2 – Example of a DAG



Circles stand for random variables ($x_{ij}, \theta_i, \sigma^2, \mu, \tau^2$), rectangles for constants (w_{ij}) and the big rectangles for the indexes (i and j). This graphical model shows that once the parameters θ_i are given, the data x_{ij} do not depend on μ or τ^2 any more. It also shows that once θ_i are given, they contain all the model information needed to update μ for instance. Graphical models are widely used in the following chapters, together with WinBUGS that appears in many examples in Chapter 3. In the applications of Chapters 5 and 6 specific programs had to be written instead.

Now, the algorithm of Gibbs sampling is as follows. Define:

$\pi(\theta_i \mid \theta_{-i}) = \pi(\theta_i \mid \theta_1, \theta_2, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_{n_k})$, for $i = 1, \ldots, n_k$.

Dropping superscript, so that $\theta = \theta^{(k)}$ and defining the starting point by $\theta^0 = (\theta_1^0, \ldots, \theta_{n_k}^0)$, it is possible to define a recursive simulation for the set of parameters, where the j^{th} sample vector $\theta^j = (\theta_1^j, \ldots, \theta_{n_k}^j)$ is sampled as follows:

$$\begin{aligned}
\theta_1^j &\sim \pi(\theta_1 \mid \theta_2^{j-1}, \ldots, \theta_{n_k}^{j-1}) \\
\theta_2^j &\sim \pi(\theta_2 \mid \theta_1^j, \theta_2^{j-1}, \ldots, \theta_{n_k}^{j-1}) \\
&\vdots \qquad\qquad \vdots \qquad\qquad (2.3) \\
\theta_{n_k}^j &\sim \pi(\theta_{n_k} \mid \theta_1^j, \ldots, \theta_{n_k-1}^j)
\end{aligned}$$

$$\theta^0 \rightarrow \theta^1 \rightarrow \theta^2 \rightarrow \ldots \rightarrow \theta^j \rightarrow \ldots\ldots \xrightarrow{D} \pi(\theta)$$

converging in distribution

Data augmentation comes naturally into this algorithm. So again the data are split into $Y=(Y^m, Y^o)$ and with Y^m fixed $\pi(\theta_i|\theta_{-i})$ is easy to sample from, for all $i=1,\ldots,n_k$. Then $p(Y^m|\theta, Y^o)$ is the sampling distribution for the missing data and it is also typically very easy to sample from. At each iteration the sample for the parameters is taken with the missing data fixed as a pre-sampled value. Then, with all parameters updated, each missing component is sampled.

Returning to the complete data case, the complete algorithm is given by the following scheme. With $\theta_{-i}^{j-1} = (\theta_1^j,\ldots,\theta_{i-1}^j,\theta_{i+1}^{j-1},\ldots,\theta_{n_k}^{j-1})$, a number, N^*, of updates are described in figure 2.3.

Figure 2.3 – Gibbs sampling algorithm

Initialise the vector set $\theta^0 = (\theta_1^0,\ldots,\theta_{n_k}^0)$

for j=1 to N^*

  for i=1 to n_k

    Sample θ_i^j from $\pi(\theta_i|\theta_{-i}^{j-1})$

  Next i

Next j

2.2.2 Metropolis-Hastings algorithm

Unfortunately it is not always true that all conditional distributions $\pi(\theta_i|\theta_{-i})$, $i=1,\ldots, n_k$, are known or easy to sample from. In this case, Gibbs sampling has to be dropped and Metropolis-Hastings should be used instead. Metropolis-Hastings is not used in the applications in this thesis, but since it is the basis to RJMCMC it is explained in this subsection.

Metropolis-Hastings is more general than Gibbs sampling, and so it is applicable to a wider number of problems. However, it is more dependent on the choices of the analyst, which interferes with the model by choosing the distribution to sample from. Suppose that no complete posterior conditional distribution $\pi(\theta_i|\theta_{-i})$, is known for all $i=1,\ldots,n_k$, and that Metropolis-Hastings is used to perform inference. In this case, a sampling distribution $q(\theta_i|\theta_{-i})$ is defined for all $i=1,\ldots,n_k$, and since the sample does not come from the real

complete conditional posterior distributions $\pi(\theta_i|\theta_{-i})$ for $i=1,\ldots,n_k$ an acceptance test has to be used.

The new distributions $q(\theta_i|\theta_{-i})$ $(i=1,\ldots,n_k)$ have to be chosen and there are many ways of deciding which distributions to use. Chib and Greenberg(1994) gives a good discussion on this topic. After defining $q(\theta_i|\theta_{-i})$ for $i=1,\ldots,n_k$ each sample from this approximation distribution has to be submitted to an acceptance test.

Now, suppose that samples of θ_i, $i=1,\ldots,n_k$ are needed. To start the j^{th} sample, define $\theta_{-i}^{j-1} = (\theta_1^j,\ldots,\theta_{i-1}^j,\theta_{i+1}^{j-1},\ldots,\theta_{n_k}^{j-1})$. The new value θ_i' $(i=1,\ldots,n_k)$ is sampled from $q(\theta_i|\theta_{-i}^{j-1})$ to substitute the old value. For each component the new value is accepted with the following probability:

$$\min\{1, \alpha(\theta_i', \theta_i^{j-1})\} = \qquad (2.4)$$

$$\min\{1, \text{(posterior ratio)} \times \text{(proposal ratio)}\}$$

where:

$$\alpha(\theta_i', \theta_i^{j-1}) = \frac{\pi(\theta_i'|\theta_{-i}^{j-1})q(\theta_i^{j-1}|\theta_i', \theta_{-i}^{j-1})}{\pi(\theta_i^{j-1}|\theta_{-i}^{j-1})q(\theta_i'|\theta_i^{j-1}, \theta_{-i}^{j-1})} \qquad (2.5)$$

If $q(\theta_i|\theta_{-i}) = \pi(\theta_i|\theta_{-i})$, then $\alpha(\theta_i', \theta_i^{j-1})=1$ and Metropolis Hastings turns out to be Gibbs sampling. The algorithm for N^* updates is as shown in figure 2.4:

Figure 2.4 – Metropolis-Hastings algorithm

Initialise the vector set $\theta^0 = (\theta_1^0,\ldots,\theta_{n_k}^0)$,

for j=1 to N^*

  for i=1 to n_k

    Sample θ_i' from $q(\theta_i|\theta_{-i}^{j-1})$

    Sample u from uniform(0,1)

    If u < min{1, $\alpha(\theta_i', \theta_i^{j-1})$} then $\theta_i^j = \theta_i'$

    else $\theta_i^j = \theta_i^{j-1}$

  Next i

Next j

2.2.3 Reversible jump MCMC

As will be seen in the next chapter, most MCMC methods have already been implemented in actuarial science, in contrast to reversible jump MCMC

that has no published application up to now. RJMCMC was first proposed by Green(1995) in order to implement MCMC in cases where it is necessary to choose a model from a finite and countable set of models, when each model has a parameter set with different dimensionality. This method is particularly suitable for problems where "the number of things that you don't know is one of the things that you don't know" (Green (1995)).

Ordinary MCMC techniques, such as Gibbs sampling and Metropolis-Hastings, are not suitable for this problem because they do not make sense when the dimensionality of the parameter vector is not fixed. The reversible jump technique is a generalisation of the Metropolis-Hastings algorithm, and the name "reversible jump" comes from the fact that there is a bijection between the parameters values, whenever there is a jump between the spaces.

Suppose there is a set of models k, k=0,1,...,k_{max}, and that for each model there is a vector $\theta^{(k)}$ of unknown parameters. If the data Y are observed, there is a natural structure to the joint distribution of model k, the parameters and the data given by:

$$p(k,\theta^{(k)},Y) = p(k)p(\theta^{(k)} \mid k)p(Y \mid k,\theta^{(k)}) \qquad (2.6)$$

The posterior distribution is given by the relation:

$$\pi(k,\theta^{(k)}) = p(k,\theta^{(k)} \mid Y) = p(k \mid Y)p(\theta^{(k)} \mid k,Y) \qquad (2.7)$$

In addition, one measure to help in the choice of the model to be used is the Bayes factor of one model related to the other:

$$BF(k',k) = \frac{p(k' \mid Y)}{p(k \mid Y)} \div \frac{p(k')}{p(k)} \qquad (2.8)$$

The Bayes factor is used in Chapters 5 and 6 to help analyse the results. Since RJMCMC is a generalisation of the Metropolis-Hastings technique, an acceptance test has to be performed. This test has a similar form to the one defined in the simpler case, but since now there is a dimensionality change, a Jacobian part is also included.

Suppose that a move of type m is proposed, that changes the parameter set from $\theta^{(k)}$ to $\theta^{(k')}$ in a higher dimensional space. The new value is derived from $\theta^{(k')}(\theta^{(k)},u)$, where set u is defined as a random variable independent of $\theta^{(k)}$ and with dimensionality dependent on the difference in dimension between k' and k. The reverse of the move (from $\theta^{(k')}$ to $\theta^{(k)}$) can be accomplished by using the inverse transformation, so that the proposal is deterministic as required by RJMCMC (see Green(1995)). Now, this move has to be tested for acceptance.

Define $q_m(\theta^{(k)})$ as the probability of choosing move type m when in $\theta^{(k)}$, and $p(u)$ as the density function of u. Then the acceptance rate is given by:

$$\min\{1, \alpha(\theta^{(k')}, \theta^{(k)})\} = \qquad (2.9)$$

$$\min\{1, \text{(posterior ratio)} \times \text{(proposal ratio)} \times \text{(Jacobian)}\}$$

where:

$$\alpha(\theta^{(k')}, \theta^{(k)}) = \frac{p(\theta^{(k')}, k' \mid Y) q_m(\theta^{(k')})}{p(\theta^{(k)}, k \mid Y) q_m(\theta^{(k)}) p(u)} \left| \frac{\partial \theta^{(k')}}{\partial(\theta^{(k)}, u)} \right| \qquad (2.10)$$

The other way round, moving from $\theta^{(k')}(\theta^{(k)}, u)$ to $\theta^{(k')}$, has an acceptance rate given by the inverse of formula 2.10. Observe that this acceptance test is almost the same as the one in the Metropolis-Hastings algorithm, only including an extra component related to the Jacobian.

The algorithm for applying RJMCMC has varied in each application, given the numerous ways in which the move m can be defined. In the original work from Green (1995) an application is proposed in a one-dimensional multiple change point example. There k stood for the number of change points, or jumps, in $[0, L)$, $\theta^{(k)} = (l_0, l_1, l_2, \ldots, l_k, s_1, s_2, \ldots, s_k)$ where l_i is the intensity in the interval $[s_i, s_{i+1})$ for $i=1,\ldots,k+1$ and $s_0 = 0$, $s_{k+1} = L$ (a full definition of this model will be given in Chapter 4).

The algorithm in Green (1995) is written for N^* steps in figure 2.5:

Figure 2.5 – Green (1995) RJMCMC algorithm

Choose k_0

Initialise the vector set $\theta^{(k_0)}$

for j=1 to N^*

Choose one type of move m (defining k'_j and $\theta^{(k'_j)}$)

Sample u' from uniform(0,1)

If $u' < \min\{1, \alpha(\theta^{(k_j)}, \theta^{(k_{j-1})})\}$ then $k_j = k'_j$; $\theta^{(k_j)} = \theta^{(k'_j)}$

else $k_j = k_{j-1}$; $\theta^{(k_j)} = \theta^{(k'_{j-1})}$

Next j

Green (1995) m defined four types of move, with their probabilities defined via a function dependent on the prior distribution and actual value of k. These moves are defined as:

(a) Update one of the intensity components;

(b) Update one of the interval position;

(c) Birth of one jump;

(d) Death of one jump.

Choices (a) and (b) sample a random value for $l_{i'}$ and s_i respectively for the sampled order i' and i (i'=0,...,k_{j-1} and i=1,...,k_{j-1}). The use of the acceptance test in these moves indicates that a Metropolis-Hastings algorithm has been used.

Choices (c) and (d) change the dimensionality of the parameter set, where the parameter related to the position of the jump is sampled randomly (either the new one in move (*c*), or the removed one in move (*d*)). The parameter related to the intensity is defined by a relationship to the values previous to the move, which is random in (c), but deterministic in (d). The use of a deterministic function in the move related to decreasing the number of jumps is given by the "dimension-matching requirement" described in Green(1995), which is observed, as expected, in all applications published so far.

The algorithm just presented has only one component of $\theta^{(k)}$ updated in each move. Differently, Green and Richardson (1997) update the full set of parameters in each move for a mixture of normal distributions problem. The means and variances of the normal distributions are all considered unknown and allocation parameters are also included in the model. The k ordered groups (order defined by the means) are predefined and each observation is then allocated to a group. By this definition some groups could have no observations, the so called empty components.

The algorithm used in Green and Richardson (1997) is as follows. Firstly the set of parameters $\theta^{(k)}$ is fully updated using Gibbs sampling. Secondly, moves *m* are carried out separately to the empty or not components, but only to change dimensionality since they had already been updated before. The move *m* has only two movements: creating, (c), or deleting, (d), a group. Their probabilities are fixed at 0.50, except when k=0 or k=k_{max} when the probabilities are (respectively) equal to 1.

The final algorithm for N^* updates is given by:

Figure 2.6 – Final RJMCMC algorithm

Choose k_0

Initialise the vector set $\theta^{(k_0)}$

for j=1 to N^*

    Generate a new $\theta^{(k_j)}$ by updating each component

    Sample (c) or (d) for a mixture component

    Sample (c) or (d) for an empty component

Next j

Denison, Mallick and Smith(1998) also present an application using RJMCMC. This is a curve fitting problem and the model uses the methodology of piecewise polynomials. For each step, the choice of move m is defined as:

(a) Update a jump position;

(b) Birth of one jump;

(c) Death of one jump.

Choice (a) requires a sample of a new value for the jump position. (b) and (c) change the dimensionality of the parameter set, where the parameter related to the position of the jump is again sampled randomly (either the new one or the choice of the one to be removed). The parameter related to the intensity was not sampled in any of the moves, but calculated as the minimum square estimator in that specific interval. This was used instead of the usual sampling procedure because "A complete Bayesian approach (...) leads to a serious computational burden, especially when many knots are required to fit the curve adequately, and comparative studies have shown that the least square estimation approach leads to no significant deterioration in performance for overall curve estimation." (Denison, Mallick and Smith(1998)).

2.3 Chapter conclusion

The basic Bayesian statistics results and the MCMC class of simulation have been explained in this chapter. Further, an algorithm has been presented for each of the revised methodologies in the MCMC class. With all these procedures at hand, the choice of the algorithm to be used in Chapter 4 can be justified.

In the next chapter an explanation of how Bayesian theory has developed in actuarial science will be given. Most of the stochastic simulation applications use Gibbs sampling.

Chapter 3

Bayesian models in actuarial science

In this chapter the development of Bayesian modelling in actuarial science is given. It is shown how Bayesian statistics was first introduced in actuarial science via credibility theory, and how it has developed into highly elaborate models in recent years. The main objective of this chapter is to build the basis for the basic model, which is the core of this thesis and which is fully explained in the following chapter.

This review is even more necessary in the actuarial application of Bayesian theory since there is no broad bibliographical review in any book or paper. Among some of the works that present partial reviews, Klugman (1992) and Smith *et al.* (1996) could be cited.

This chapter is constructed in the following way. In the first section some considerations are given about the practical side of Bayesian applications. In section 3.2 some traditional areas are reviewed, showing their development in the Bayesian approach. In section 3.3 some more elaborate models are reviewed, which are part of the basis of the model in this thesis. In the last section a conclusion for this chapter is presented.

3.1 Practical considerations on Bayesian models

Although many developments have occurred in Bayesian statistics, very few practitioner actuaries are aware of them and even fewer make use of them. In fact, since the advent of credibility theory, which has at its core Bayesian statistics, this statistical philosophy has not been broadly used in practice. It was in 1914 that the first paper on credibility theory was published, with

actuaries being one of the first practitioners to use the Bayesian philosophy. Since then many developments in credibility theory have occurred, but it is probably the only tool based on Bayesian theory used in an office environment, and even this is rare. However, judgement is used on an everyday basis and it is often argued that in this way an informal Bayesian approach is used.

Three main reasons could be listed to justify why actuaries do not use Bayesian theory more often: model comprehension, implementation and portability. In the first reason the application of this theory needs a full understanding of the model, including, among many other features, the choice of prior distributions and the interpretation of parameters. In this way an actuary has to become a skilled statistician, which can be off-putting for many practitioners.

After understanding the model, it is not guaranteed that its implementation is easy: most of the time, the calculations are complicated and extensive. And in many cases an analytical solution can not even be found, but must be substituted by a simulation based implementation. It is well known that practical actuaries prefer not to rely on an approximation when performing calculations.

Unfortunately, even after applying the model to a specific problem the implementation of the same structure to a different data set could also be highly complicated. Such lack of portability happens mostly with models that do not use much aggregation (called individual models), which are the models with innovative approaches such as the ones used in Boskov and Verrall (1994), Haastrup (1997) and in the present work. Again, a practical actuary relies on portability in order not to spend too much time applying the model to new problems or to new data.

Although the practical side of actuarial science has not been influenced by Bayesian science, the academic side has developed many models. The most interesting applications are the models based on individual policies rather than aggregated data and this thesis focuses on these. A new stochastic simulation technique is used in order to create more applications of models that focus on the raw data of individual policies. These models could be difficult to apply to a different problem, but it is hoped that their ideas and results are attractive enough to stimulate some future investigation.

Now some examples are explored. In the next section traditional models in credibility, reserving and graduation are reviewed, showing the development

of ideas from purely analytical solutions to the use of simulation. This is the basis for the complete departure from traditional models presented in section 3.3.

3.2 Traditional models

This section looks at some traditional models of actuarial theory in some traditional areas: credibility theory, claims reserving and graduation (subsections 3.2.1, 3.2.2 and 3.2.3 respectively). These areas have been chosen mostly by the amount of papers published in each of them, but credibility theory is also chosen given its historical importance. In each of these risk assessment procedures the original and usual model is transformed in order to explore the Bayesian structure. Usually it happens via a prior distribution, by including structure and values.

3.2.1 Credibility theory

Credibility theory was first introduced by Mowbray (1914), almost at the same time as the Casualty Actuarial Society was created. At that time the actuaries had to define a premium for a new insurance product – "workmen's" compensation -- so they based the tariff on a previous kind of insurance which was replaced by this new one.

As new experience arrived, a way of including this information was formalised, mixing the new and the old experiences. This mixture is the basis of credibility theory, which searches for a credibility estimator that balances the new but volatile data, and the old data that have historical support. Most of the research until the mid XX century went in this direction, creating the branch of credibility theory called limited fluctuation.

The turning point in this theory, and the reason why it is used nowadays, happened when actuaries realised that they could bring this mixture idea inside a portfolio. This new branch of credibility theory is called greatest accuracy, and it searches for an individual estimator (or a class estimator), but still using the experience for the whole portfolio. Such an estimator would consider the "own" experience on one side, but giving more confidence to it by also including a more "general" one on the other side. In a way, it formalises the mutuality behind insurance, without the loss of the individual experience.

There are many papers discussing this theory, but the one by Bühlmann (1967) is generally seen as a landmark. In this paper credibility theory was completely formalised, giving its basic formula and philosophy. Since then, many models have been developed and a review can be found in Venter (1986). Now, as an example of how credibility works, the Bühlmann and Straub (1970) model will be written and applied.

The data set is taken from Klugman (1992). The observations are the number of claims (y_{ij}) for 133 occupations (i=1,...,133) in workers' compensation insurance with 7 years experience (j=1,...,7). The respective amount of the payroll (w_{ij}) is also known and is used as a weight for each occupational class.

Modelling the frequency ratio x_{ij} (y_{ij}/w_{ij}) via the Bühlmann and Straub model gives the following distributions:

$$x_{ij} \mid \theta_i \sim \text{normal}(\theta_i, \sigma^2/w_{ij})$$
$$\theta_i \sim \text{normal}(\mu, \tau^2) \qquad (3.2)$$

for all i and j, and σ^2, μ and τ^2 known. The appropriate graphical model is shown in figure 2.2 of Chapter 2. In this model $p(x_{ij} \mid \theta_i)$ stands for each class experience, $p(\theta_i)$ for the overall portfolio information, implying that each class mean comes from the same distribution. Now, with $\bar{x}_i$ as the observed mean and z_i as the credibility factor for class i, the credibility estimator for the class ratio θ_i is:

$$\bar{x}_i \times z_i + \mu \times (1 - z_i) \qquad (3.3)$$

The solution proposed by Bühlmann and Straub (1970) is to estimate the values of σ^2, μ and τ^2 from the observations, substituting these values and coming out with the solution for the formula above. With $\hat{z}_i$ as the estimated value of z_i, after including the values for the variances and $\bar{\bar{x}}$ as the overall observed mean, formula (3.3) changes to:

$$\bar{x}_i \times \hat{z}_i + \bar{\bar{x}} \times (1 - \hat{z}_i) \qquad (3.4)$$

It may not be clear where the prior information has been inserted into this model. The reason for this is that the formula (3.4) was developed in order to balance the information of the class's own observed experience, $\bar{x}_i$, with the observed overall one, $\bar{\bar{x}}$. In this model the distribution $p(\theta)$ does not play a role of a real Bayesian prior, but its parameters are substituted by the values calculated from the data set.

As seen in Chapter 2 this is one form of empirical Bayes estimation. In order to have a fully subjective Bayesian solution another level of distribution

would have to be included. This would contain information about the parameters σ^2, μ and τ^2, which are considered unknown, with the third level on formula (3.2) expressed as $p(\sigma,\mu,\tau)$. This new distribution is the prior distribution for the unknown parameters in the previous distributions. Unfortunately, unless very strong assumptions for $p(\sigma,\mu,\tau)$ are included, it is not possible to derive the posterior distribution for θ.

In order to get closer to a pure Bayesian approach, Klugman (1992) included priors for σ^2, μ and τ^2, but in a "non-informative" way. No analytical solution is available and an approximation technique (Gaussian quadrature) was used. In this solution a program had to be specifically written in order to carry out the model implementation and, depending on the approximation technique chosen, the calculations could take 2 hours. Both solutions[1] are shown for some classes in the following table:

Table 3. 1 – Results for analytical solutions of credibility theory

	Solution				Forecasting			
Class	$\sum_{j=1}^{6} w_{ij}$	$\bar{x}_i$	Bühlmann and Straub	Klugman	w_{i7}	y_{i7}	Bühlmann and Straub	Klugman
4	0.037	0.0	0.03949	0.04045	-	0	-	-
11	1,053.126	0.04446	0.04345	0.04422	229.83	8	9.99	10.16
112	93,383.54	0.00188	0.00201	0.00193	18,809.7	45	37.81	36.30
70	287.911	0.0	0.02059	0.01142	54.81	0	1.13	0.63
20	11,075.31	0.03142	0.03164	0.03151	1,315.37	22	41.62	41.45
89	620.968	0.42997	0.29896	0.36969	79.63	40	23.81	29.44
					Forecast error[2]		15.55	13.20

When including information in $p(\sigma,\mu,\tau)$, the solution is more difficult to calculate. In order to do so, stochastic simulation is applied via BUGS by Pereira (1998). Scollnik (1996) and Smith (1996) also used similar procedures. The Bühlmann and Straub model is written in a WinBUGS (or BUGS) program in figure 3.1 with the following set of prior distributions, which had a non-informative objective:

$$\begin{aligned} \mu &\sim \text{normal } (0,\ 10^5) \\ 1/\tau^2 &\sim \text{gamma } (0.001,\ 0.001) \\ 1/\sigma^2 &\sim \text{gamma } (0.001,\ 0.001) \end{aligned} \qquad (3.5)$$

The implementation of this model took 5 minutes on a fairly old computer, with a total of 2500 simulations, where the first 500 were discarded to eliminate the effects of the initial values.

[1] Although data were observed for 7 years the two solutions only use 6 years to do the calculations.

Figure 3.1 – WinBUGS program

```
model BühlmannStraub; [3]
const
  N = 130,  # number of classes
  U = 6;    # number of observed years
var
  mu,
theta[N],Y[N,U],tau,sigma,w[N,U],sigc[N,U];
  data in "datafile";
  inits in "initialfile";
{
   mu ~ dnorm (0, 1.0E-5);
   tau ~ dgamma (3.0E-3,1.0E-3);
   sigma ~ dgamma (3.0E-3,1.0E-3);
   for (i in 1:N) {
     theta[i] ~ dnorm (mu,tau);
     for (j in 1:U) {
       sigc[i,j] <-sigma*w[i,j];
       Y[i,j] ~ dnorm (theta[i],sigc[i,j]);
     }
   }
}
```

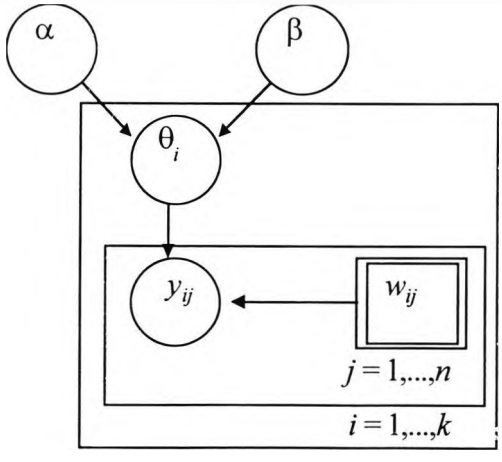
Now an extension is proposed. Since the observations are numbers of claims it is more suitable to model the data using a Poisson distribution rather than normal distributions. This is a direct generalisation of the previous model, where the new model uses non-informative prior distributions and is written as (also represented as graphical model in figure 3.2):

$$y_{ij} \mid \theta_i \sim \text{Poisson}(\theta_i \times w_{ij})$$

$$\theta_i \mid \alpha, \beta \sim \text{gamma}(\alpha, \beta) \qquad (3.6)$$

$$\alpha \sim \text{uniform}(0.01, 50) \text{ and } \beta \sim \text{uniform}(0.01, 50)$$

Figure 3.2 – DAG Poisson distribution model



[2] Forecast error $= \sum_{i=1}^{133} (Forecast - Y_{i7})^2 / w_{i7}$

[3] In WinBUGS instead of the variance, the precision (1/variance) is used for the normal distribution

In this way the Bühlmann and Straub model has been dropped and a more suitable model has been constructed. The new model did not take much longer than the previous one to be implemented with the same amount of data. Since one of the main quantities of interest is the forecast of the number of claims for the 7th year, this is done in WinBUGS by sampling directly the values of y_{i7}, for i=1,..., 133, which are included in the model as missing data. The table below gives the results, where the value of the deviance is related directly to the forecast value of y_{i7}.

Table 3. 2 – Results for Normal and Poisson models

Observed data			Normal		Poisson	
Class	w_{i7}	y_{i7}	Forecast	Deviance	Forecast	Deviance
4	-	0	-	-	-	-
11	229.83	8	10.15	7.28	10.21	3.57
112	18,809.67	45	38.24	66.06	35.64	6.63
70	54.81	0	0.60	3.47	0.254	0.57
20	1,315.37	22	41.24	16.84	41.48	6.55
89	79.63	40	29.56	4.11	32.82	6.17
Forecast error			13.22		12.44	

Comparing these values with the ones found without stochastic simulation, it is observed that the Normal solution (Bühlmann and Straub model with non-informative prior) is almost the same as the previous ones and very close to the Klugman(1992) solution as expected. The benefit of using the Poisson distribution can be seen in the smaller forecast error found in this case. And it is also observed that in many classes the deviance was smaller when the Poisson distribution was assumed.

It was seen that an improvement can be obtained by using stochastic simulation. Not only was the forecast error smaller, but also it was possible to define a more realistic model than the previous one. Prior information was also used without much complication. Continuing the proposed chapter structure, in the next subsection the development of the claims reserving problem will be presented.

3.2.2 Claims reserving

Claims reserving is one of the most important branches in the general insurance area of actuarial science. Usually a macro model, where data are

accumulated by underwriting and development year, is used, and the data are given in a triangular format. One of the features of those models is the small amount of data available for the later development years, which gives a large degree of instability to any estimate. Actuaries overcome this problem through professional judgement when they chose factors or consider benchmarks.

In this subsection another way of including this subjective information is given, which is more formal, statistically speaking, since it uses a prior distribution. The approach used here is the chain ladder technique, which is one of the most popular macro methods to predict claims reserves. But in the following examples no inclusion of the tail factor is considered.

The data come from Taylor and Ashe (1983), and the exposure factor for each underwriting years and the claims data are given below, where the influence of the exposure has to be taken out from the claim amount before any analysis.

Table 3. 3 – Claims reserving data

Underwriting year \ Development year										
	357848	766940	610542	482940	527326	574398	146342	139950	227229	67948
	352118	884021	933894	1183289	445745	320996	527804	266172	425046	
	290507	1001799	926219	1016654	750816	146923	495992	280405		
	310608	1108250	776189	1562400	272482	322053	206286			
	443160	693190	991983	769488	504851	470639				
	396132	937085	847498	805037	705960					
	440832	847651	1131398	1063269						
	359480	1061648	1443370							
	376686	986608								
	344014									

Exposure: 610 721 697 621 600 552 543 503 525 420

In Kremer (1982), which is a paper on credibility theory, the chain ladder is proved to be similar to the two way analysis of variance linear model expressed by:

$$x_{ij} = \ln(y_{ij}) \qquad (3.7)$$

with x_{ij} independent normal(θ_{ij}, σ^2),

where $\theta_{ij} = \mu + \alpha_i + \beta_j$

and y_{ij} is the incremental value of the claims for row (underwriting year) i and column (development year) j.

The solution of Kremer (1982) is to calculate the MLE of the unknown parameters together with the estimate of σ^2. In Verrall (1990), which is the paper reviewed here, the same model is used but in a claims reserving context

and with a Bayesian solution being applied. In fact three Bayesian solutions are presented: "pure Bayes without prior information", "pure Bayes with prior information" and "empirical Bayes". In order to have an analytical solution, none of these models includes a prior distribution for the variance parameters.

A prior distribution is attached to the model in (3.7) and this is rewritten here in matrix notation:

$$X \mid \theta \sim \text{normal } (K\theta, \sigma^2 I)$$

$$\theta \mid \theta_1 \sim \text{normal } (\theta_1, \Sigma) \qquad (3.8)$$

where

$X = (x_{11},\ldots,x_{1n},x_{21},\ldots,x_{2n-1},\ldots,x_{n1})$, $\theta = (\mu, \alpha_2,\ldots, \alpha_n, \beta_2, \ldots, \beta_n)$,

$\Sigma = \text{diag}(\sigma_\mu^2, \sigma_\alpha^2,\ldots,\sigma_\alpha^2, \sigma_\beta^2,\ldots, \sigma_\beta^2)$, $\theta_1 = (\mu^*, \alpha_2^*, \ldots, \alpha_n^*, \beta_2^*, \ldots, \beta_n^*)$,

K is the design matrix to produce the model in (3.7),

I is the identity matrix, σ^2, σ_μ^2, σ_α^2, σ_β^2 are known variances,

$\alpha_1 = \beta_1 = 0$ for uniqueness (see Verrall (1990) for details).

The "pure Bayes without prior information" uses a non-informative prior approach. In this way, σ_μ^{-2}, σ_α^{-2} and σ_β^{-2} go to zero and the model solution gives exactly the same results as the classical and usual MLE used in Kremer (1982).

But more information could be inserted straight into this second level distribution, instead of using the non-informative one for all parameters. This is the "pure Bayes with prior information" approach and it is applied by changing θ_1 and Σ in order to keep the non-informative approach for parameters $(\mu,\beta_2, \ldots, \beta_n)$, but not for the row parameters. Proper prior distributions for $(\alpha_2,\ldots,\alpha_n)$ are defined, but they are hard to choose, since there is no intuitive explanation related to them. In this example the following set of prior distributions (based on the result obtained at the MLE model) was chosen[4]:

$$\alpha_i \sim \text{normal } (0.3, 0.05); \qquad (3.9)$$

for all $i = 2,\ldots,n$.

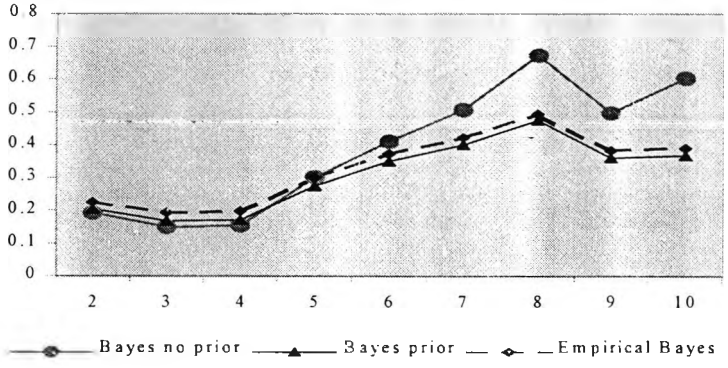
The third approach, "empirical Bayes" is based on the credibility theory assumption, that there is some dependency among the parameters related to the row and they are not really independent as before. So, in formula (3.6) the non-informative approach is kept for $(\mu,\beta_2, \ldots, \beta_n)$, $(\sigma_\mu^{-2} = \sigma_\beta^{-2} = 0)$, but a

[4] which is the same as $\alpha_i^* = 0.3$ for $i=2,..,n$ and $\sigma_\alpha^2 = 0.05$.

different one is imposed for the row parameters. Now, instead of defining a distribution like (3.7), the general distribution (3.6) is kept and another level of prior distribution is added to $(\alpha_2^*,\ldots,\alpha_n^*)$, with a non-informative approach. In this way no prior value is given, but only a dependency among the row parameters is imposed.

All three models were applied to this data set. "Pure Bayes without prior information", which is the equivalent to the MLE solution by Kremer (1982), had the worst performance when compared to the other two in all analyses done by Verrall (1990). "Pure Bayes with prior information" and "empirical Bayes" also had a better smoothness to the row parameters as can be seen in figure 3.3:

Figure 3.3 – Graph of claims reserving results

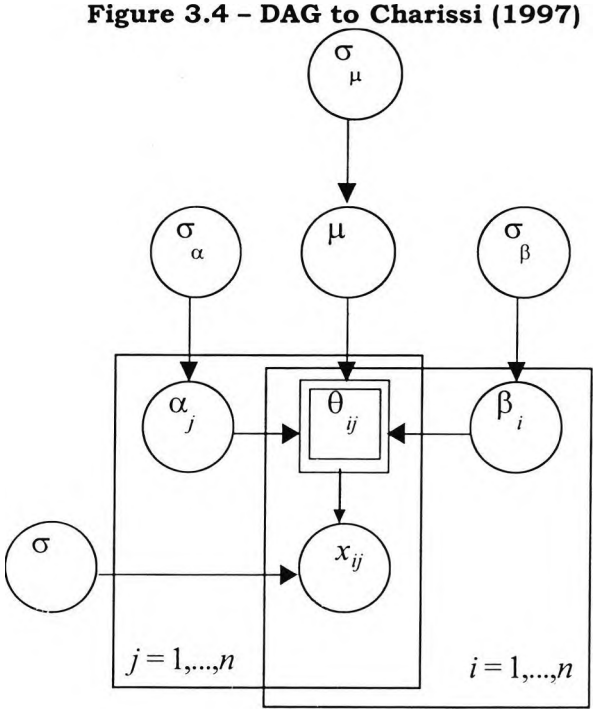


Not much research has been done in order to implement chain ladder based models using WinBUGS. This is mainly due to the amount of missing values presented in the triangular format (in the usual format the outstanding claims are treated as missing values in WinBUGS). So in order to use triangular data, the model was implemented either using specifically written programs, or by imposing very strong assumptions. Other researchers have used new models, which would not use the data in the triangular format, but the individual claim experience. An overview of what has already been done in this direction will be given in section 3.3.

Two works using triangular data and stochastic simulation are cited now. The first one is Charissi (1997) where the "pure Bayes without prior information" model in Verrall (1990) is reanalysed using BUGS, with a proper prior distribution for each of the parameters. These are included in the second level, and independently of the chosen distribution, each one had to be centred on the values observed in the data, with quite a low variance.

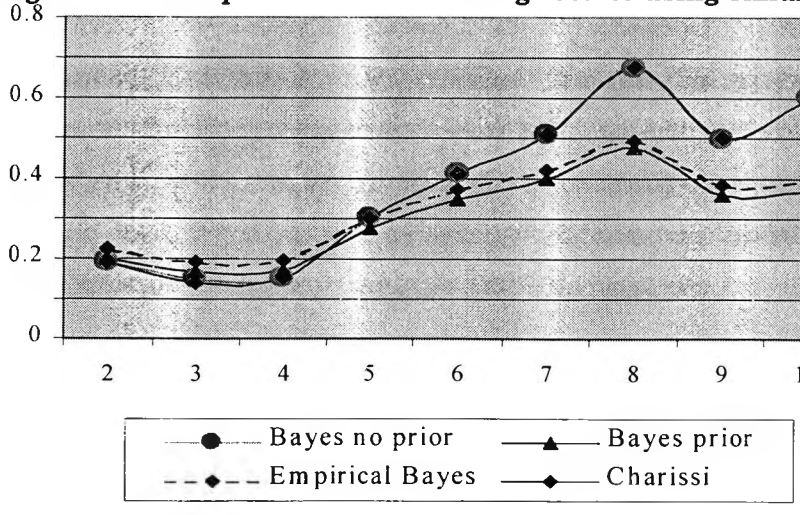
The DAG model would be as in figure 3.4:

Figure 3.4 – DAG to Charissi (1997)



The data from Taylor and Ashe (1983) were reanalysed and the results for the posterior mean for the row parameter are plotted in the figure 3.5 together with the values found before in Verrall (1990). On one hand, it is easy to see that the set of chosen prior was not able to influence greatly the mean of the row parameters (or even the other ones), keeping the same result as the one found in "pure Bayes with no prior". But, on the other hand, in this new analysis the influence of the prior was enough to decrease the standard error of the parameters by an average of 30% compared to the previous approach.

Figure 3.5 – Graph of claims reserving results using simulation



The second paper is Ntzoufras and Dellaportas (1997). Gibbs sampling is again used as the simulation technique, but although this paper was prepared after the development of BUGS, a specific implementation program

was used instead. Five models were presented in the paper and all of them were applied to the same data set. This set includes the inflation rate for the observed calendar years and two incremental development triangles: amount and number of claims. With all of this information to hand they proposed new models that would take into consideration the number of claims in order to predict the claim amounts, which would be deflated before any analysis. Only one model among all five is fully explained here.

"Log-normal & Poisson model" is a direct generalisation of Kremer (1982). Now, instead of using only the information from the amount of claims, the history of the number of claims (n_{ij}) reported in row i and column j is also taken into consideration. The model in (3.7) is changed to:

$$
\begin{aligned}
& x_{ij} = \ln(y_{ij}) \\
& x_{ij} | \theta_{ij}, \sigma^2 \sim \text{normal}(\theta_{ij}, \sigma^2) \\
& \theta_{ij} = \mu + \alpha_i + \beta_j + \ln(n_{ij}) \\
& n_{ij} | \lambda_{ij} \sim \text{Poisson}(\lambda_{ij}) \\
& \ln(\lambda_{ij}) = \mu^* + \alpha_i^* + \beta_j^*
\end{aligned}
\qquad (3.10)
$$

with constraints and prior distributions for μ, α_i, β_j, μ^*, α_i^*, β_j^*, σ^2 fully described in the original paper.

An analysis was performed with all models, and it was shown that for the specific data used the models that included also the number of claims, like the one explained above, had a better prediction than the ones that did not use such information. This was mainly due to the long tail characteristic of the data set, where claims were still being reported after 7 years of occurrence.

This subsection has shown that the flexibility of the simulation approach was able to allow also the inclusion of the development of number of claims in the chain ladder model. Now the last subsection on the development of traditional models via Bayesian theory will discuss graduation. Although it is a pension and life insurance area and this thesis is on general insurance, it is important to include it to show the smoothing models that have appeared.

3.2.3 Graduation

Graduation is an important part of the job of a life actuary and many methods have been developed in order to carry it out. Using the definition from Haberman (1996) "graduation may be regarded as the principles and methods by which a set of observed probabilities are adjusted in order to

provide a suitable basis for inference to be drawn and further practical computations to be made".

The usual data set to which graduation is applied includes an estimate of the number of policyholders exposed to risk in the observation period (usually one year) and the number of deaths which occurred. In order to illustrate this, the following sample was taken from London (1985):

Table 3.4 – Graduation data

Age (i)	Frequency rate (x_i)	Number of:	
		Policyholders (w_i)	Deaths (y_i)
63	0.00928	9,487	88
64	0.01226	10,770	132
65	0.01100	24,267	267
66	0.01120	26,791	300
67	0.01481	29,174	432

The mortality rate, x_i, is defined as $x_i = y_i / w_i$.

Whittaker graduation, Whittaker (1923), is one of the most well known methods among actuaries. This can be considered as the first Bayesian approach to graduation, since it can be derived using Bayes theorem. But no real prior subjective information was formally used in the first development of this model.

A model that could be seen as a step before Whittaker graduation is the Kimeldorf and Jones (1967) model explained in London (1985). This model is not fully written here, but states that the observed frequency of death is modelled as:

$$X|\theta \sim \text{normal}(\theta, B)$$
$$\theta \sim \text{normal}(\mu, A) \qquad (3.11)$$

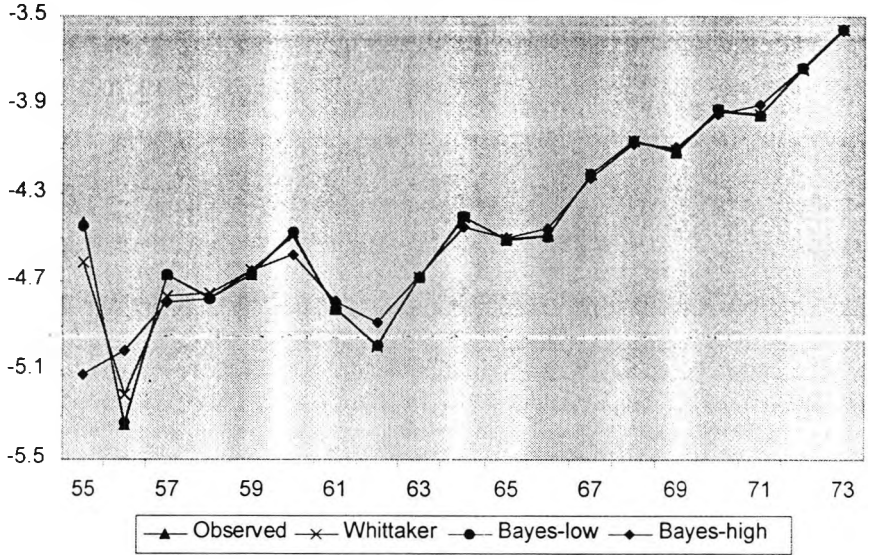
where

$X = (x_1, \ldots, x_n)$, $\theta = (\theta_1, \ldots, \theta_n)$, $\mu = (\mu_1, \ldots, \mu_n)$,

n is the number of ages and A and B are known covariance matrices.

μ is taken from another life table and B is fixed. The covariance matrix A is defined by the analyst and it controls the amount of smoothness. Some other possible formats are discussed in London (1985). The graduated values are obtained as the posterior mean of θ and the graph of the estimates in the example analysed in London (1985) is shown on a log scale in figure 3.6.

Figure 3.6 – Graduation example comparison graph



The two Bayes results show how to control the model, with higher and lower level of smoothness, depending on the chosen value of A. Bayes-low is so close to the observed data that it is even hard to distinguish them.

Klugman (1992) brings a different approach to the Whittaker model. Instead of using prior information from another table, as in London (1985), a relationship is imposed among the parameters in θ. In order to do this, a design matrix is included transforming the model into:

$$X|\theta \sim \text{normal}(\theta, B)$$

$$K\theta \sim \text{normal}(0, A) \qquad (3.12)$$

Where K is the matrix that produces the z^{th} differences of a sequence of numbers. Choosing properly the values for A and B and letting $z = 3$ gives the posterior mean as the same solution as the one proposed by the Whittaker model. But in the new Whittaker approach not only the estimator of θ was found, but also its covariance matrix. In this way a confidence region could be easily found.

In fact, one of the first applications of the model expressed in formula (3.9) was the calculation of a reserve, where a confidence interval was also presented. The case when a prior distribution is given for A and B is also analysed in Klugman (1992).

The paper by Carlin (1992) uses Gibbs sampling technique to graduate not only mortality table but also the ageing factor cost related to health insurance. In both of these applications some restrictions were imposed on the model structure, such as the growth on mortality expected in adulthood. Here only the mortality example is explained.

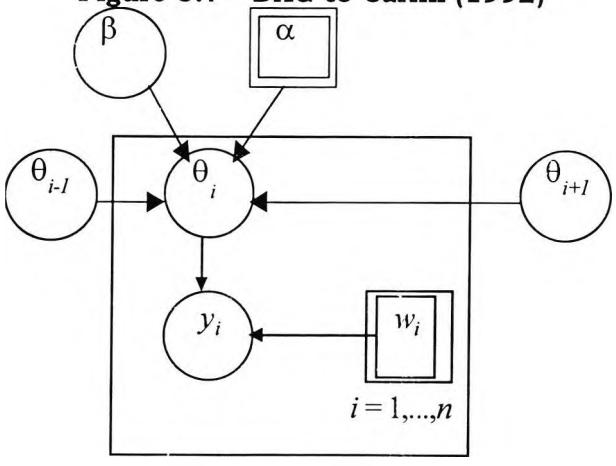
The paper was developed before BUGS was implemented, so a specially written program carried out all the calculations. In the graduation problem the data set has ages from 35 to 64, so 30 ages were observed. The model states that the number of deaths y_i in age i+34 for i=1,...,30 has a Poisson distribution with intensity given by $\theta_i \times w_i$ where w_i is the number of policyholder in i. The model is written as:

$$y_i|\theta_i \sim \text{Poisson}(\theta_i \times w_i)$$
$$\theta_i|\beta \sim \text{gamma}(\alpha, \beta) \qquad (3.13)$$

Where $\theta_1 > 0$, $\theta_{30} < B$, $0 < \theta_2 - \theta_1 < \ldots < \theta_{30} - \theta_{29}$, B and α fixed, using a suitable prior distribution for β. A graphical model for this model is shown in figure 3.7, where the imposed order among the parameters θ is also represented.

Some constraints were also imposed on the model and the more interested reader should refer to the original paper in order to see these in full. The results are also compared with the ones obtained by the Whittaker model and the author comments that "Whittaker results are fairly similar to the Bayes results, though the Whittaker rates tend to be influenced more by the unusually low rate at age 63". The model was able to keep the parameters θ increasing with age, although this was not observed for all ages in the crude rates.

Figure 3.7 – DAG to Carlin (1992)



An application of BUGS to graduation can be found in Kouyoumoutzis (1998). In this work a number of models were investigated and the one explained here is based on a third degree polynomial regression analysis expressed by:

$$y_i|\theta_i \sim \text{Poisson}(w_i \times \theta_i) \qquad (3.14)$$

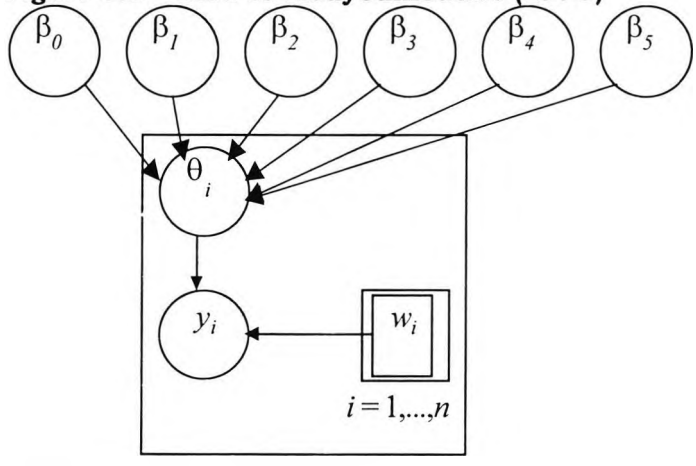
with $\ln(\theta_i)$ expressed as:

$$\beta_0+\beta_1 i+\beta_2(3i^2-1)/2+\beta_3(5i^3-3i)/2+\beta_4(35i^4-30i^2+3)/8+\beta_5(315i^5-350i^3+75i^2)/40,$$

and β_j~ normal (0,1000) for j=0,...,5.

The time needed to run the simulation was again very small and the smoothed values fitted the data well. The graphical model is show below in figure 3.8.

Figure 3.8 – DAG to Kouyoumoutzis (1998)



In this section 3.2 a review of traditional models revised to use a Gibbs sampling approach has been given. Different, new models were incorporated by the inclusion of simulation into the modelling process. It is expected that the more actuaries are able to use WinBUGS, and more generally Gibbs sampling, the more revisions of traditional models will emerge.

In the next section completely new ideas are presented. The assumptions used in macro models are completely dropped and models with approaches closer to the process itself is used.

3.3 Modern Bayesian models

In the previous section well known models were discussed and rewritten in order to give a Bayesian approach. But one of most appealing features of a Bayesian analysis is the broader set of models that can be built, models which do not have a classical equivalent approach. This is the basis of this thesis and also of a few other theses and papers (Norberg (1993), Boskov and Verrall (1994), Arjas and Haastrup (1996), Haastrup (1997) for instance).

It may turn out (and this is something that remains to be seen) that the most important of these new ideas is the ability to model at the individual policy level. This approach, an "engineering approach", when assumptions are made directly in the process itself rather than on the aggregated data, fits fairly easily within a Bayesian model.

In this section two examples are presented. The first one is the use of spatial models in the rating by area problem, not using individual data but only the loss ratio and exposure by area. The individual data are considered in the second model, which is an application to claims reserving, but now considering the individual data, instead of the usual triangular format. Both models use the simulation approach, but neither could use WinBUGS and a specific implementation program had to be written.

3.3.1 Rating by postcode area

There are many factors that could influence the frequency or cost of a claim and that should be taken into consideration when defining the value of the premium. One of these is the area where, for example, a car is used or parked most often and this characteristic is usually taken into account through the neighbourhood where the policyholder lives.

Neighbourhood could have many interpretations, but here postcode is used. In an office environment it is common to aggregate postcodes with similar experiences in the same class. At the end of this procedure a small number of classes is derived, but the vicinity information is not formally taken into account by the model.

Taylor (1989) published the first paper with some statistical basis, which addressed how to carry out this aggregation using the vicinity information. He adapted a two-dimension splines model to the postcode problem, with a totally non-Bayesian approach.

In this chapter a review of Boskov and Verrall (1994) is presented. That paper used a Bayesian approach, applying spatial models which are mainly used in epidemiology and satellite image restoration, among other fields. The basis for such models is that areas that are close together are more likely to be similar in risk than areas that are far apart.

The aim of the model is to find a value for risk parameter (θ_i), that is smoothed oven the whole area (that contains n postcodes) but considering only information from its neighbours. The data contain the observed loss ratio (x_i) for each postcode area i, and they are assumed to have a normal distribution as follows:

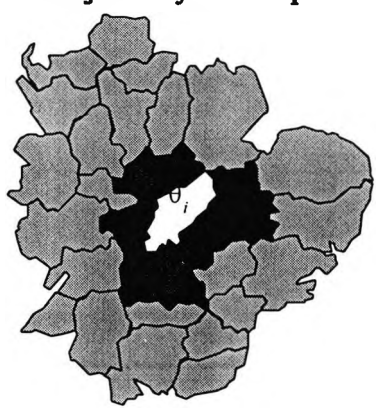
$$x_i \mid \theta_i \sim \text{normal}\ (\theta_i,\ \sigma/w_i) \qquad i = 1,\ldots,n \qquad (3.15)$$

Where w_i is the exposure for postcode area i. Instead of using the variance as a variable as in some models seen before in this chapter, σ is a

constant chosen by the analyst fixing the required level of smoothness. The bigger σ, the smoother the result for the posterior mean of θ_i.

The most important idea of the model comes in the definition of the second level of distributions, when a relationship among the risk parameters θ_i is defined. For each postcode risk θ_i an adjacency set is defined as in figure 3.9, where the darker areas are included in the neighbourhood of the risk.

Figure 3.9 – Adjacency set to postcode risk θ_i



So the risk parameter of each postcode is defined to be normally distributed, centred on the average of all risk parameters in the adjacency set. All risk parameters are defined at the same time, influencing their neighbours as well.

This model does not have a possible analytical solution, and a simulation approach was used in order to find the posterior of θ_i. A MCMC method was used and the full model explanation can be found in the original paper. It is really interesting to observe that the risk parameters really took some information from the neighbours, and a smoothed map of the London postcode area was obtained.

In the following subsection, models considering individual data are presented. The solution found in Arjas and Haastrup (1996), which is the only applications reviewed here, is derived through simulation. It took a long time to perform the implementation, what is considered a barrier to future practical use.

3.3.2 Claims reserving

The majority of methods for estimating claims reserves are based on macro models, where the data are aggregated in a discrete time and triangular format like the chain ladder model. Micro models, where the continuous time and individual policyholder characteristics are statistically taken into

consideration are not usual. The only office-based procedure that takes into consideration some individual information is the case reserve definition, when the claim characteristic is used by an expert to predict its final value without any statistical modelling.

Individual models and continuous time are highly connected since the aggregation of data usually requires a more concise information on time as well. One of reasons why this individual characteristic is not used in statistical models could be the difficulties that surround any calculation on an individual claim basis. The fluctuation related to any individual estimate, could also be a good justification for the lack of use of such models. The key question would be to use such information, but in a more robust way.

Before analysing the example in Arjas and Haastrup (1996), it would be helpful to think of the claim process. Jewell (1989) analyses the occurrence of claims together with the delay of reporting. In this way the number of claims which have a reporting delay – IBNR – is estimated via a Bayesian model. Time is considered as continuous.

Norberg (1993) proposes a more comprehensive model, further developed in a later paper Norberg (1999). In his model not only the occurrence of a claim is modelled, but also its development. The occurrences are modelled as a non-homogeneous Poisson process with the corresponding development as position-dependent marks. Formally, a claim is a pair (T, Z) in $(0,\infty) \times \Omega$, where T is the time of occurrence of the claim and Z is the mark describing its development and defined as:

$$Z = (U, V, X, \{X(v'); 0 \leq v' < V\}) \qquad (3.16)$$

where:

U is the waiting time from occurrence until notification;

V is the waiting time from notification until final settlement;

X is the final claim amount ($X=X(V)$);

$X(v')$, the indemnity paid in respect of the claim at time v' after its notification.

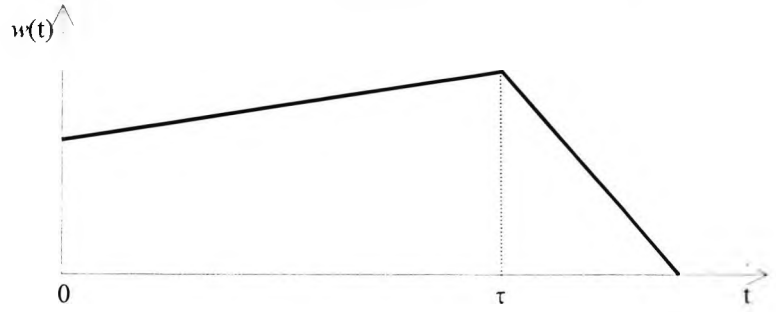
Fixing the time of reserve evaluation at τ (called the present time in many of these papers), suppose that there are N claims that have occurred in $(0, \tau]$. If W is defined as the total exposure related to the business written in that period of time (expressed by the area in figure 3.10), the distribution of the total number of claims is given by:

$$N \sim \text{Poisson}(W); \qquad (3.17)$$

where $W = \int_0^{\infty} \omega(t)dt$

and $w(t)$ is the exposure function.

Figure 3.10 – Weight function scheme



So $0 < T_1 < T_2 < \ldots < T_N$ are the moments of occurrence of the claims that will follow a non-homogeneous Poisson process given by:

$\{T_i\}_{1 \le i \le N} \sim \text{Poisson}(w(t);\ t \ge 0)$

Where the total claim process has the following structure:

$\{T_i, Z_i\}_{1 \le i \le N} \sim \text{Poisson}(w(t), P_{Z|t};\ t \ge 0)$

The dependence of the frequency of claims only on $w(t)$ can be generalised by including a random fluctuation, which is chosen to be of the multiplicative form in the application on Chapter 6 and in Arjas and Haastrup(1996). In these models the intensity changes to $w(t)\varphi(t)$ and W is rewritten as:

$$W = \int_0^{\infty} w(t)\varphi(t)dt\,. \qquad (3.18)$$

In order to define $P_{Z|t}$ Norberg (1993) decomposes Z, categorising the claim at a specific time τ into disjoint sets: *settled*, *reported-not-settled*, *incurred-not-reported* or *covered-not-incurred*. The first one does not need any reserve, but the following ones give rise to the RBNS (reported but not settled), IBNR (incurred but not reported) and UPR (unearned premium reserve). Those four components can be viewed as arising from independent marked Poisson processes (which facilitates the calculation) whose intensities and mark distributions are expressed as (from theorem 2 in Norberg (1993)):

$$\{T_i^j, Z_i^j\}_{1 \le i \le N} \sim \text{Poisson}(w^j(t), P^j_{Z|t};\ t \ge 0) \qquad (3.19)$$

$$w^j(t) = w(t)\, P_{Z|t}(\Omega^j)$$

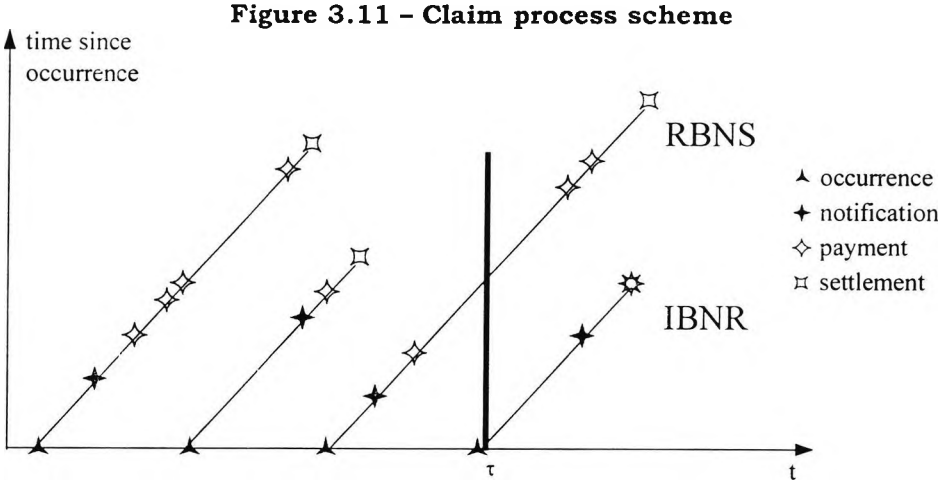
$$P^j_{Z|t}(dz) = \frac{P_{Z|t}(dz)}{P_{Z|t}(\Omega^j)},\ z \in \Omega^j$$

$$\text{with } \sum_{j \ge 1} P_{Z|t}(\Omega^j) = 1,\ \bigcup_{j \ge 1} \Omega^j = \Omega \text{ and } \bigcap_{j \ge 1} \Omega^j = \varnothing$$

Decomposition is the focus of Norberg (1999), where this idea is extended to "quite general countable decompositions". As an example this last paper includes the decomposition by year of notification and occurrence is proposed in order to find a connection with discrete models. Such work also highlights a hierarchy in the mark Z. Its first three components (U, V, X) are primary to the claim. By knowing its full development the partial payment process could be viewed as a secondary interest. In this way the probability distribution of Z will, usually, be constructed in two steps. The first related to U, V, X and the second to $X(v)$.

Now a claim scheme is represented in figure 3.11, showing how each claim could be different from another and also highlights the partial payment process, which for most types of insurance is more usual than a single payment.

Figure 3.11 – Claim process scheme



The mark Z could also include more information and Arjas and Haastrup(1996) proposed an application of this model, where the marks Z also include some covariates and were given by:

$$Z = (S, A, U, V, X, \{X(v); 0 \leq v' < V\}) \qquad (3.20)$$

where:

U, V, X and $X(v')$ are as in (3.16);

S is the sex of the policyholder;

A is the age of the policyholder.

Claim frequency and severity are modelled separately. For claim frequency, age, sex, report delay and calendar time of occurrence are included, and for claim severity the analysis uses partial payments, which are only dependent on the number of partial payments and the time since latest partial payment if any, or else the time since notification. MCMC simulation is used in order to obtain the estimated posterior distributions. By some conditional

independency results, IBNR and RBNS claims reserves are calculated separately.

The way of handling missing values in a Bayesian framework is also explored and the IBNR claims are considered as missing. Since simulation is used, it is possible to sample at each step the number of claims that have already occurred and that are missing (IBNR) and their corresponding amounts. At the end of the simulation a sample of IBNR numbers and values is available and its posterior distribution can be approximated. The amount of RBNS claims is calculated in the same way.

The results of this model will be further explored in Chapter 6. This model suggests many ideas for further development. If individual information could be taken into account in a statistical model, it means that characteristics of the claim itself could also be formally considered.

3.4 Chapter conclusion

As was shown throughout this chapter, many models have been developed in a Bayesian framework. Some of them were only an extension of well known models, but others included new ideas to the actuarial analysis. More than 20 papers were reviewed, being only a sample of a large variety of works, but which illustrate the range of applications

In the following chapter a basic model is built, using the ideas expressed in section 3.3 and the simulation technique RJMCMC. In chapter 5 the first application is presented, but it is in chapter 6 that the papers by Norberg (1993, 1999) will be used to build the model for that specific application.

Chapter 4

Theoretical framework

In the previous chapter a review of some Bayesian models in actuarial science was given. Now a model based on individual data is built, continuing the idea described in section 3.3. In this fourth chapter, a full explanation of the ideas and theory behind the basic model used in the applications in Chapters 5 and 6 is presented, including the implementation techniques and algorithm.

It is noted that the structure of the model is quite simple, based on a piecewise constant parameter. However, the fact that the dimensionality of the parameter set is not fixed in advance makes its implementation quite complicated. The way chosen to implement this model is via reversible jump Markov chain Monte Carlo (RJMCMC), a generalisation of the Metropolis-Hastings stochastic simulation technique explained previously in section 2.2.3.

As a brief idea of the application of this model, suppose that it is desired to transform a continuous variable limited to the interval $[0,L)$ into a few groups. The basic model defines, at the same time, how many jumps there should be, where they should be located and the risk parameter related to each group. This is a change to the usual approach of fixing first the number of jumps and then defining their locations and risk parameters. In this usual approach the parameters (number of jumps, their location and risk parameters) are defined separately, which causes interpretation problems depending on the dependency among the parameters.

The outline of the chapter is the following. Section 4.1 explains the basic model. In section 4.2 the basic model is presented in two special cases: a

change-point and a mixture of normal distributions models. After understanding the model and having the implementation techniques, the algorithm is presented in section 4.3. Section 4.4 presents the chapter conclusion.

4.1 Basic model

Since the applications in Chapters 5 and 6 are based on models that use a common idea, in this section the basic idea is explained. The model presented in this section will be called from now on as the "basic model". In each one of the applications in the following chapters some changes are made, but the ideas are kept as in this basic one.

The basic model considers that a piecewise constant distribution is applied to a parameter, say $\varphi(t)$, which depends on the covariate t located in a bounded set Ω derived from real numbers set $\Re$, in the continuous case, or integer numbers set Z, in the discrete case. The parameter $\varphi(t)$, generally called the level, takes a piecewise constant value over a set of intervals, defined by the number of jumps in Ω. This is a simple model, but its structure is complicated because it is not known in advance how many intervals $(k+1)$, or jumps (k), there should be in Ω. In addition, it is not known where the intervals, or jumps, should be located, or the level $\varphi(t)$ related to them.

Generally speaking, the model can be written in the following way:

$$Y|\varphi(t) \sim \text{distribution}(\varphi(t)),\ t \in \Omega \qquad (4.1)$$

where:

Y is the data set ($Y=(y_1,y_2,\ldots,y_n)$);

$\varphi(t) = l_j$ if $t \in [s_j, s_{j+1})$ for $j=0,\ldots,k$ with $\varphi(inf(\Omega)) = l_0$, $\varphi(sup(\Omega)) = l_k$;

(with fixed $s_0 = inf(\Omega)$, $s_{k+1}=sup(\Omega)$)

In order to help with the comprehension of the model, its DAG is displayed in figure 4.1.

Once k is fixed the parameters of the model are given by $\varphi(t)$, $t \in \Omega$, and defined by a piecewise function, which is constant and not stochastic. In this way the unknown parameters of the model could be expressed by the following set:

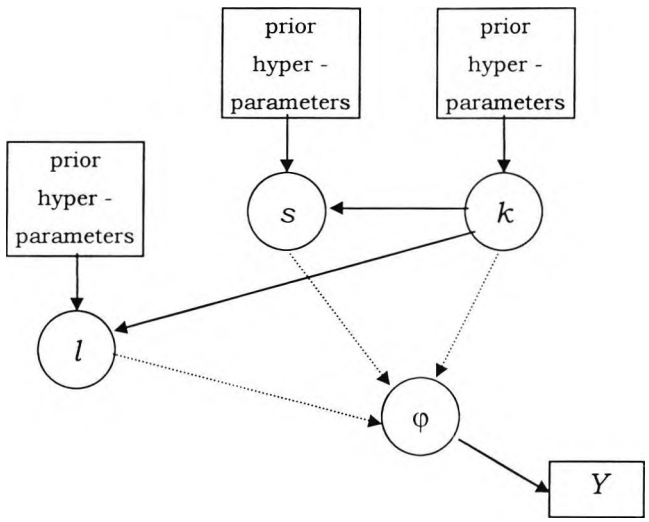
k is the unknown number of jumps (k+1 groups);

$l_0, l_1, l_2,\ldots, l_k$ are the levels related to each group;

$s_1, s_2,\ldots, s_k$ are the times of change of level.

Since k is not a constant, the dimensionality of the set of parameter is not fixed. Given k, there is a set of unknown parameters $\theta^{(k)} = (l_0, l_1, l_2,\ldots, l_k, s_1, s_2,\ldots, s_k)$ with dimensionality $n_k = 2k+1$.

Figure 4.1 – Basic model DAG

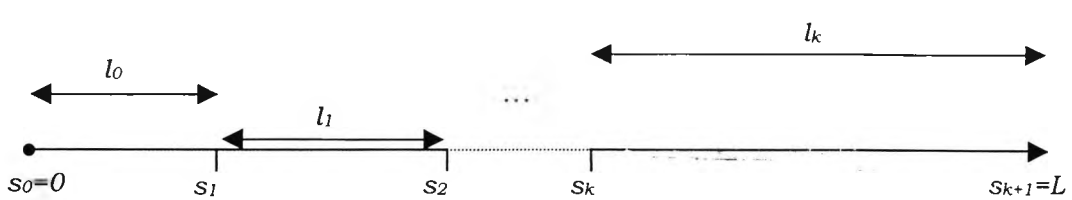


Among many applications, two main ones could be noted in such a model. The first one is the already cited search of the number of groups, their locations and levels. Use of the model above can provide the analyst with these three answers at once. Such an application is the basis for the definition of age groups in the automobile insurance premium in Chapter 5.

In the second application the analyst is more interested in the smoothing property of the model. In a structure like the one above, the change in the number of jumps influences the location of the group intervals, allowing the level of one specific group to be influenced by the level of the other group. This smoothing is the main characteristic for the applications in Chapter 6.

Supposing t to be continuous and $\Omega = [0, L)$, the figure 4.2 shows how the model works for k jumps.

Figure 4.2 – Model scheme



In all applications, frequency and severity of claims are considered. They are considered independent and the above model is applied to each one of them separately. The model is adapted to the case of analysing severity and frequency of claims for the case where t is either continuous or discrete. For

the frequency case the assumption of claims arriving from a Poisson process is used, and in the severity case a normal distribution is assumed. The specification of the full application is in the following chapters, but in the next subsections their basis is built. In order to guide the reader through the work, the application, together with its location and type of covariate t is specified in the table below:

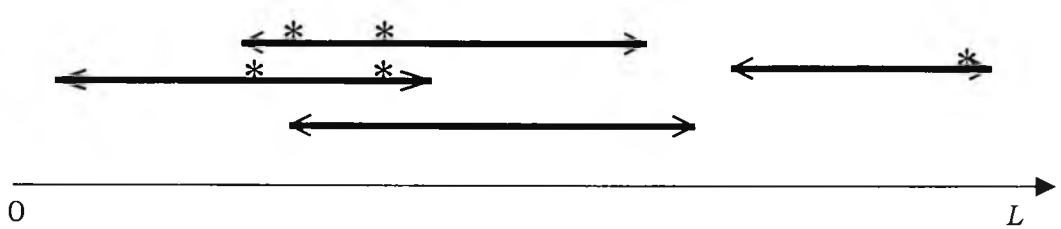
Figure 4.3 – Distribution of models over the thesis

	Problem	t
Chapter 5	Grouping ages	*Age* as continuous
Chapter 6	Claims reserving	*Age, sex* as discrete *Reporting delay* and *calendar time* as continuous

Now the full formulae are derived for frequency and severity separately. The covariate t is considered continuous throughout the whole explanation, and at the end of the chapter the changes are shown for the discrete case. For simplification on the calculations, in the continuous case the moments of jumps s_j, $j=1,\ldots,k$, are not allowed to occur at the same moment as the observations.

The type of data also changes depending on the application, but in all cases a structure like figure 4.4 is observed. For each policyholder (represented by an arrow) the exposure is given, and in the case of claims their value and the time of occurrence in $[0,L)$ are also included (represented as a star).

Figure 4.4 – Claim structure



4.1.1 The model for the claim frequency

The occurrence of a claim is presumed to follow an inhomogeneous Poisson process with intensities $\varphi(t)$, $t \in [0,L)$. Supposing there are n claims observed in this interval the essential part of the likelihood is given by:

$$\prod_{i=1}^{n} \varphi(y_i) \exp\{-\int_0^L w(t)\varphi(t)dt\} \qquad (4.2)$$

where

y_i is the time of observation of each claim i, i=1,2,3,...,n;

$y_i \in [0,L)$, and

$w(t)$ is the exposure function.

The formula presented in (4.2) can be rewritten in order to explore some of the characteristics of the model. Since $\varphi(t) = l_j$ if $t \in [s_j, s_{j+1})$ for j=0,...,k, the likelihood can be rewritten as:

$$\prod_{j=0}^{k} l_j^{n_j^{j+1}} \exp\{-l_j w_j^{j+1}\} \qquad (4.3)$$

where

w_j^{j+1} is the total exposure in $[s_j, s_{j+1})$ ($w_j^{j-1} = \int_{s_j}^{s_{j+1}} w(t)dt$), and

n_j^{j+1} is the total number of claims in $[s_j, s_{j+1})$.

This approach is in between parametric and non-parametric models, since it allows the estimated $\varphi(t)$ to reflect the behaviour of the data in a way that is not possible under strong parametric assumptions.

In the prior it is necessary to specify a distribution for k, s_j, l_0, and l_j for j=1,...,k. The prior for k is defined to be a Poisson distribution with rate λ.

$$p(k) = \frac{\exp^{-\lambda} \lambda^k}{k!} \qquad k \le k_{max} \qquad (4.4)$$

The hyper-parameter k_{max} is included in order to guarantee that only a finite number of models are considered.

Given k, it is possible to define a prior distribution for the values of l_0, l_j and s_j for j=1,..,k. s_j is distributed, for j=1,...,k, as the even-numbered order statistics from $2k$+1 points uniformly distributed in $(0,L)$. It was decided to only pick the even ordered ones in order to avoid that two successive jumps are very close together. l_j follows a Gamma distribution with parameters (α,β) and this distribution was chosen as the conjugate prior distribution for the likelihood. This set of priors is the same as used in Green(1995).

Before expiaining the procedure in the severity case, a discussion of the time of occurrence of claims is necessary. One of the assumptions of a Poisson process, widely use to model occurrence of claims, is the zero probability of more than one claim occurring at the same instant in time. Although the Poisson process has been shown to fit the claim occurrence very well, this

assumption is not realistic for real data. It happens again here in the data sets used in the following chapters.

4.1.2 The model for the claim severity

The second model for t continuous is defined for the case where the value of the claim depends on the covariate t. A Normal distribution is assumed and the variance is assumed to be a known value. The level parameter $\varphi(t)$, which stands for the distribution mean in the severity case, follows the basic model. In this way, the model states that the data follow a mixture of normal distributions, without fixing the number, k, of distributions involved in the mixture.

It is assumed that claim values follow a normal distribution Normal($\varphi(t)$, σ^2) with variance σ^2 fixed and equal in all groups. The reasons for using a constant and equal variance are discussed later in this subsection, and the problems of this limited approach are discussed throughout the applications in Chapters 5 and 6.

The likelihood is written as:

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\sigma^2\pi}} \exp\{-\frac{(h(y_i)-\varphi(y_i))^2}{2\sigma^2}\} \qquad (4.5)$$

with $h(y_i)$ as the claim value.

Again this expression could be simplified by using the structure of the model. Since $\varphi(t) = l_j$ if $t \in [s_j, s_{j+1})$ for $j=0,\ldots,k$, the likelihood could be written as:

$$\left(\sqrt{2\sigma^2\pi}\right)^{-n} \exp\{-2^{-1}\sigma^{-2}\sum_{j=0}^{k}(hs_j^{j+1} - 2h_j^{j+1}l_j + n'_j(l_j)^2)\} \qquad (4.6)$$

where

n'_j is the number of claims in $[s_j, s_{j+1})$, with $n=n'_0+\ldots+n'_k$;

h_j^{j+1} is the sum of all values of claims that occurred in $[s_j, s_{j+1})$, and

hs_j^{j+1} is the sum of square of all claims that occurred in $[s_j, s_{j+1})$.

The prior distribution set is the same as defined in subsection 4.1.1 for all the parameters except $\varphi(t)$. The relationship with s_j, $j=1,\ldots,k$ is the same as described in the frequency problem and the prior distribution for l_j follows a normal distribution, Normal(μ,τ^2), for $j=0,\ldots,k$.

The value of σ^2 was chosen to be fixed and equal in all groups to simplify the calculations. Because of the structure of the model the algorithm

is quite slow, as will be seen in the applications in Chapters 5 and 6. The use of a stochastic σ^2 would increase this calculation time, which caused implementation problems.

4.2 Complete algorithm

In Chapter 2 all the implementation techniques were discussed, with an algorithm following each stochastic simulation technique. Now, with the full model described for the case where t is continuous, in the frequency and severity case (subsections 4.1.1 and 4.1.2 respectively) a complete algorithm is given for this specific model. This algorithm involves MCMC in all parameter updates.

Recall that the aim is to simulate a sample from the posterior distribution $p(k,\theta^{(k)}|Y)$. Hence, it is necessary to define a way of moving the sampler between the different dimensionality spaces, and updating the values of all parameters at the same time. At the end of a large number of simulations, the posterior frequency distribution of the number of jumps k is obtained and also a large sample from the posterior distribution for the other parameters $\theta^{(k)} = (l_0, l_1, l_2, \ldots, l_k, s_1, s_2, \ldots, s_k)$.

In order to achieve this, at each step the Gibbs sampling is used to update all components of $\theta^{(k)}$ with the number of jumps, k, fixed from the previous step as in the approach of Green and Richardson (1997). Then a move m for k is chosen from:

(a) Keep the number of jumps, k, the same;

(b) Birth of a jump;

(c) Death of a jump.

If the move changes the dimensionality of $\theta^{(k)}$ ((b) and (c)) then the acceptance test from RJMCMC technique is used, and if the move is accepted the value of $\theta^{(k)}$ changes. If the move m keeps the value of k the same, then the updated value via Gibbs sampling for $\theta^{(k)}$ is kept and another step starts again.

For N^* updates, the algorithm is given as in figure 4.5.

Figure 4.5 – Gibbs sampling algorithm

Initialise the vector $\theta^{(k_0)}$, with k_0 jumps.
for j=1 to N^*
    Update $\theta^{(k'_j)}$ using Gibbs sampling
    Define k'_j (or choose the type of move m)
    If $k'_j = k_{j-1}$ then $k_j = k'_j$
    else do acceptance test
Next j

This is a different procedure from Green (1995) and Denison, Mallick, and Smith(1998), when the move m is chosen first and only the components of $\theta^{(k)}$ directly involved on the update of k are sampled. It means that in each step only part of $\theta^{(k)}$ is updated, and since an acceptance test is performed, an update is not even guaranteed. The acceptation rate for RJMCMC in the applications seen so far (subsection 2.2.3) is around 8-17%. The benefit of using the approach adopted in this thesis is that the whole parameter vector $\theta^{(k)}$ is updated in every step.

The algorithm is divided into two sampling sets: the updating of $\theta^{(k)}=(s_1,\ldots, s_k, l_0,\ldots, l_k)$ and then k. When updating k, a change in the dimensionality n_k could occur and in this case RJMCMC is used. The case of updating $\theta^{(k)}$ is done in two steps: the position of a jump (s_j, j=1,..., k) and the respective level (l_j, j=0,..., k). Gibbs sampling is used to update $\theta^{(k)}$ when k is fixed.

4.2.1 Updating $\theta^{(k)}$

In order to update $\theta^{(k)}$, k is supposed fixed. For the parameters related to the position of jump (s_j, j=1,..., k) the use of a step function is needed, which makes this update the most time demanding in all applications. For the level parameters (l_j, j=0,..., k) the update is more straightforward. Both the procedures in the position and level parameters are explained, differing in formulae for the frequency and severity cases (subsections 4.2.1.1 and 4.2.1.2).

All parameters are sampled via Gibbs sampling. So it is necessary to define complete conditional distributions for s_j, l_0, l_j for $j=1,\ldots,k$. Starting with the levels, consider updating l_j. The complete conditional distribution is given by:

$$\pi(l_j \mid l_0,\ldots,l_{j-1}, l_{j+1},\ldots,l_k, s_1,\ldots, s_{k+1}, k, \textit{hyper-parameters, constants}) =$$

$$\pi(l_j \mid s_j, s_{j+1}, k, \textit{hyper-parameters related to } l_j) = \pi(l_j \mid .)$$

This simplification is possible given the Markovian structure of the process. Once the distribution $\pi(l_j \mid .)$ is fully specified the value of l_j related to the interval $[s_j, s_{j+1})$ can be updated.

In the same way it is possible to show that the moment of a jump s_j has complete conditional distribution given by the following relationship:

$$\pi(s_j \mid s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_{k+1}, l_0, \ldots, l_k, k, \textit{hyper-parameters, constants}) =$$

$$\pi(s_j \mid s_{j-1}, s_{j+1}, l_{j-1}, l_j, k) = \pi(s_j \mid .)$$

Before splitting the calculations according to types of data, a general procedure for this case is defined. Following Arjas and Gasbarra (1994) a partition is defined on (s_{j-1}, s_{j+1}), where the new value for s_j is located. Hence, the position of s_j is determined by considering the partition of (s_{j-1}, s_{j+1}) induced by the ordered observations $y(t)$ in that interval. Suppose there are n_{j-1}^{j+1} observations in (s_{j-1}, s_{j+1}), split it into $n_{j-1}^{j+1}+1$ disjoint intervals and denote the partition by $I_1,\ldots,I_{n_{j-1}^{j+1}+1}$. Then in each interval I_i, $i=1,\ldots,n_{j-1}^{j+1}+1$, the conditional probability density of s_j has a form proportional to $\exp[a_i f(s_j)]c_i$ and this can be normalised by dividing by the constant $const_j$, where:

$$const_j = \sum_{i=1}^{n_{j-1}^{j+1}+1} c_i \int_{I_i} \exp[a_i f(t)]dt \qquad (4.7)$$

where the function $f(t)$ has the following format:

$$f(t) = \begin{cases} t \text{ for frequency data} \\ 1 \text{ for severity data} \end{cases} \qquad (4.8)$$

In this way, $const_j$ has an analytical solution for any $t \in (s_{j-1}, s_{j+1})$ and s_j has its complete conditional posterior distribution for all $j=1,\ldots,k$.

Now the approaches in this thesis and in Arjas and Gasbarra (1994) will differ. For comparison the scheme proposed by Arjas and Gasbarra (1994) is explained and shown in figure 4.6. Suppose an initial set of values $(l_0^0, s_1^0, l_1^0, s_2^0, \ldots, s_k^0, l_k^0)$ is given and that a posterior sample is needed for $\theta^{(k)}$. Then in Arjas and Gasbarra (1994) the initial level l_0 is updated, followed by the next moment of a jump s_1, then l_1, then s_2, ... until l_k is updated. In this way, it is possible to observe that the levels l_j, $j=0,\ldots, k-1$, are not sampled

based on the updated interval value $[s_j, s_{j+1})$ but based on the old position values. In that paper it is not a problem to have this characteristic, but it would cause distortion to the type of analysis that is performed in the applications in this thesis.

In order to avoid such problems, an adaptation of the sampler in that paper is used. Their sampler is used to update only s_j, $j=1,\ldots,k$, in the following way: l_0, s_1, l_1, s_2,..., s_k, l_k are updated as before. After defining this sample new updates on the levels are performed. So, the final Gibbs sampler for this set of parameters with N^* updates is given by the scheme in figure 4.7, where the move that updates k is explained in subsection 4.2.2.

Figure 4. 6 - Scheme in Arjas and Gasbarra (1994)

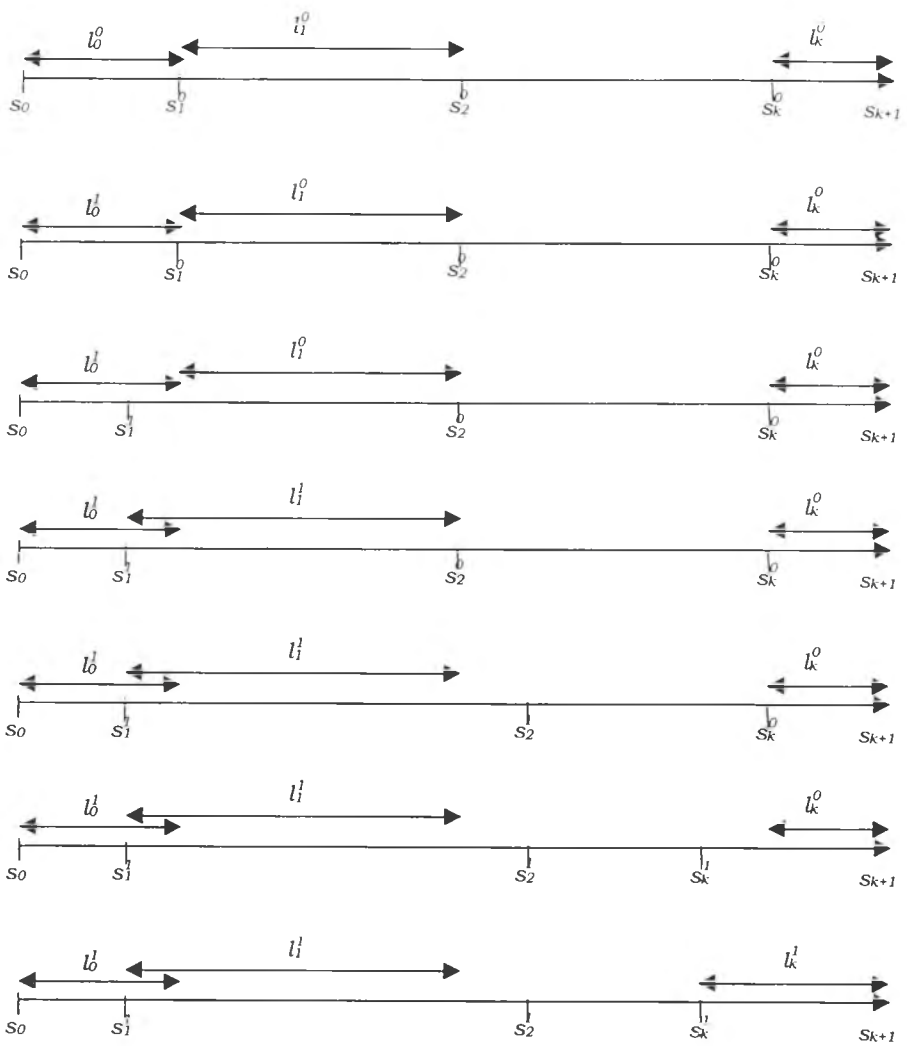


Figure 4.7 – Gibbs sampling algorithm for $\theta^{(k)}$

Initialise the vector set $(k^0, l_0^0, s_1^0, l_1^0, s_2^0, \ldots, s_{k^0}^0, l_{k^0}^0)$.

for j=1 to N^*

Update l_0, s_1, l_1, s_2,..., s_k, l_k in this order

Update l_0, l_1, ..., l_k in this order

Update k

Save values for all parameters

Next j

After defining the Gibbs sampler for both level and position parameters, the specifications of $\pi(s_j|.)$ and $\pi(l_j|.)$ are given for the severity and frequency cases.

4.2.1.1 Frequency

In the frequency case the complete conditional distribution for the level parameter is given by:

$$\pi(l_j | l_0,\ldots,l_{j-1}, l_{j+1},\ldots,l_k, s_1,\ldots, s_{k+1}, k, \lambda, \alpha, \beta) = \pi(l_j | s_j, s_{j+1}, k, \alpha, \beta) = \pi(l_j|.)$$

with:

$$\pi(l_j|.) \sim \text{Gamma}[\alpha + x_j^{j+1}, \beta + w_j^{j+1}] \qquad (4.9)$$

where

x_j^{j+1} is the total number of claims observed in $[s_j, s_{j+1})$, and

w_j^{j+1} is the total exposure in $[s_j, s_{j+1})$

The complete conditional distribution for the moment of jumps is given by:

$$\pi(s_j | s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_{k+1}, l_0, \ldots, l_k, k, \lambda, \alpha, \beta) = \pi(s_j|.)$$

with:

$$\pi(s_j|.) = l_{j-1}^{x_{j-1}^j} l_j^{x_j^{j+1}} \exp\{-w_{j-1}^j l_{j-1} - w_j^{j+1} l_j\}.p(s_j) \qquad (4.10)$$

where:

x_{j-1}^j is the total number of claims observed in $[s_{j-1}, s_j)$

w_{j-1}^j is the total exposure in $[s_{j-1}, s_j)$

$p(s_j)$ is the prior distribution for s_j, and

w_j^{j+1} and x_j^{j+1} as defined above.

4.2.1.2 Severity

In the severity case the complete conditional distributions for the level parameter is given by:

$$\pi(l_j \mid l_0,\ldots,l_{j-1}, l_{j+1},\ldots,l_k, s_1,\ldots, s_{k+1}, k, \lambda, \mu, \sigma, \tau) = \pi(l_j \mid s_j, s_{j+1}, k, \mu, \sigma, \tau) = \pi(l_j \mid .)$$

with:

$$\pi(l_j \mid .) \sim \text{Normal}(\mu^*, \tau^{2*}) \qquad (4.11)$$

where

$$\tau^{-2*} = \frac{n_j^{j+1}}{\sigma^2} + \frac{1}{\tau^2} \text{ and } \mu^* = \left(\frac{h_j^{j+1}}{\sigma^2} + \frac{\mu}{\tau^2}\right) \times \tau^{2*}$$

And the complete conditional distribution for the moment of jumps is given by:

$$\pi(s_j \mid s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_{k+1}, l_0, \ldots, l_k, k, \lambda, \mu, \sigma, \tau) = \pi(s_j \mid .)$$

with $\pi(s_j \mid .) \propto$:

$$\exp\left\{-\frac{1}{2\sigma^2}\left[-2l_{j-1}h_{j-1}^{j} + n_{j-1}^{j}l_{j-1}^2 - 2l_j h_j^{j+1} + n_j^{j+1}l_j^2\right]\right\} \qquad (4.12)$$

where:

h_{j-1}^{j} is the total value of claims observed in $[s_{j-1}, s_j)$

h_j^{j+1} is the total value of claims observed in $[s_j, s_{j+1})$, and

n_j^{j+1} and n_{j-1}^{j} are as defined above.

4.2.2 Updating *k*

After updating $\theta^{(k)}$, the number of jumps, k, is sampled. Since this affects the dimensionality of the parameter $\theta^{(k)}$, reversible jump MCMC is used. So it is necessary to define which of the moves (a), (b) or (c) in section 4.2 should be chosen, together with the probabilities of each of these possibilities. Following Green(1995) it is defined that:

(a) η_k is the probability of keeping the same number of jumps,

(b) b_k is the probability of changing from k to k+1 jumps, and

(c) d_k is the probability of changing from k to k-1 jumps.

These probabilities must satisfy: $b_k + d_k + \eta_k = 1$ for each k, and moves (b) and (c) are also called birth and death of a jump.

Define:

$$b_k = c.\min\{1, p(k+1)/p(k)\} \qquad (4.13)$$

and

$$d_k = c.\min\{1, p(k-1)/p(k)\} \qquad (4.14)$$

for $k = 1,2,\ldots,k_{\max}-1$ where $\eta_k = 1 - b_k - d_k$.

When $k=0$, $b_0 = c.\min\{1,\lambda\}$ $\eta_0 = 1 - b_0$, leaving the probability of death equal to zero. When $k = k_{\max}$ the opposite is done, leaving the probability of birth equal to zero and $d_{k_{\max}} = c.\min\left\{1, \frac{k_{\max}-1}{\lambda}\right\}$ and $\eta_{k_{\max}} = 1 - d_{k_{\max}}$. c is dependent on the values chosen for k_{max} and λ. In Green (1995) c = 0.40 and for simplicity this value is used in all the applications in this thesis.

Once these probabilities are defined the acceptance test has to be constructed. In move (a) nothing is done, since $\theta^{(k)}$ have already been updated before. In the birth move (b), the position of a new jump is selected from a uniform distribution on Ω. Define this value as s^*, and suppose that it is in the interval (s_j, s_{j+1}). If it is accepted s'_{j+1} is set to s^*, and $s_{j+1}, s_{j+2}, \ldots, s_k$ are relabelled as $s'_{j+2}, s'_{j+3}, \ldots, s'_{k+1}$, with corresponding changes to the labelling of the levels. The new levels l'_j, l'_{j+1} are defined on the intervals $[s_j, s^*)$ and $[s^*, s_{j+1})$ through a relation that is governed by a sample from an uniform distribution and a weight depending on the new intervals. Different relations are used in this thesis, but all of them follow a structure like the following:

$$g(s_j, s^*) f(l'_j) + g(s^*, s_{j+1}) f(l'_{j+1}) = g(s_j, s_{j+1}) f(l_j), \qquad (4.15)$$

and

$$\frac{l'_{j+1}}{l'_j} = \frac{1-u}{u} \qquad (4.16)$$

where u is drawn from the uniform distribution (0,1).

The acceptance probability for this move is given by:

min{1, (likelihood ratio) × (prior ratio) × (proposal ratio) × (Jacobian)}

Where the Jacobian is defined by the change from (l_j, u) to (l'_j, l'_{j+1}). The likelihood ratio, which is given by dividing the likelihood for the new set of parameters over the old one, is straightforward and not defined here. The prior ratio becomes:

$$\frac{p(k+1)}{p(k)} \frac{2(k+1)(2k+3)}{L^2} \frac{(s^* - s_j)(s_{j+1} - s^*)}{s_{j+1} - s_j} \frac{p(l'_j)p(l'_{j+1})}{p(l_j)}; \qquad (4.17)$$

where only the prior for (l_j, l'_j, l'_{j+1}) changes depending on whether the model is working on severity or frequency. The proposal ratio becomes:

$$\frac{d_{k+1} L}{b_k (k+1)}; \qquad (4.18)$$

In order to define the Jacobian, formula (4.15) has to be fully specified. The function *f(l)* is given by:

$$f(l_j) = \begin{cases} l_j & \text{in the severity case} \quad (4.19) \\ \ln(l_j) & \text{in the frequency case} \quad (4.20) \end{cases}$$

And since the definition of this formula influences the acceptance probability of the new level l_j, the function $g(.)$ has two formats, both used on the frequency case.

$$g(s_j, s_{j+1}) = \begin{cases} s_{j+1} - s_j & (4.21) \\ w_j^{j+1} & (4.22) \end{cases}$$

In all cases the Jacobian is given by:

$$\frac{(l'_j + l'_{j+1})^2}{l_j}. \quad (4.23)$$

In the death move (c), the jump to be deleted is sampled with equal probability from the existing jumps. The acceptance probability for the corresponding death move has the same form with the appropriate change of labelling of the parameters, and the ratio terms inverted.

The final algorithm for updating the set of parameters in the claim frequency problem is given by figure 4.8:

Figure 4.8 – Complete RJMCMC algorithm

Initialise the vector set $(k^0, l_0^0, s_1^0, l_1^0, s_2^0, \ldots, s_{k^0}^0, l_{k^0}^0)$.

for j=1 to N^*

Update l_0, s_1, l_1, s_2,..., s_k, l_k in this order

Update l_0, l_1, ..., l_k in this order

Choose move type m

If move type is (b) or (c) sample values for the components of $\theta^{(k)}$ affected by this change:

If the move is accepted, change the dimensionality of $\theta^{(k)}$

Else keep old values of $\theta^{(k)}$

Else do nothing

Save values for all parameters

Next j

4.3 Model for the discrete case

The model for the discrete case does not change considerably from the continuous case. In the birth move (b), the value of jump is selected from a uniform distribution on Ω, which is equal to (0,1,2,...,L) in the discrete case. The same adaptation occurs for the priors of s_j, for $j=1,...,k$, which is distributed as the even-numbered order statistics from $2k+1$ points uniformly distributed in $(0,1,...,L)$.

Besides these changes, most of the calculations are easier to perform. As an example, to update the positions s_j, for $j=1,...,k$, the split into disjoint ordered groups is done automatically by using a discrete variable instead.

4.4 Chapter conclusion

In this chapter the whole basis for the applications in Chapters 5 and 6 was built. The model, implementation and algorithm were presented and their theory explained.

It is important to highlight the use, in the algorithm proposed here, of the combination of RJMCMC and Gibbs sampling. In this way it is more likely that a good sample is taken than when only the acceptance test proposed in RJMCMC is used.

Chapter 5

Investigation of the grouping of ages

It is sometimes the case in actuarial work that a large quantity of data arises relating to a specific problem. The transformation of such data into information for rating or reserving, as some examples, is one of the main issues in actuarial science.

In order to summarise and investigate this information the actuary, or statistician, often has to decide on some kind of aggregation. Sometimes this procedure only takes into account the rating structure used in the insurance market. If a review of such a structure is needed, some well-known stochastic techniques are used and the type of covariate analysed usually governs which statistical technique is appropriate.

In the case of a discrete variable or factor, many techniques are available. These include linear regression and cluster analysis among many others, but they will not be revised here since they are not the subjects of this thesis. However, in the case of a continuous variable the available techniques are limited, and in many cases a transformation to discrete values has to be carried out before performing the transformation into factors. For example, Lemaire (1977) used stepwise linear regression for aggregation in actuarial context.

Clustering analysis is also a technique applicable to this problem. In most cases, this approach involves the use of a dummy variable applied whenever it is necessary to group a continuous variable. The problem with such an approach is that the analyst has to have a good a priori opinion of how to define the groups that will be tested. Artificial intelligence has also been applied to this subject in the research literature, where the discretization

technique attempts to find the right places to set up interval borders in a continuous variable.

In this chapter the model described in chapter 4 is used to group a continuous variable that is limited to some interval [0, L], where the initial point is defined to be zero without lost of generality. Now, it is considered how many intervals there should be in [0, L], and where these would be best located. The level risk for each group is also evaluated at the same time. The method uses a Bayesian approach, and hence the point estimate is replaced by the posterior distribution for the relevant quantities. In other words, the level risk is assumed to be a piecewise constant function over [0,L] and the task is the complete specification of this function.

A continuous variable in insurance appears in many types of cover, but age is the most common one and influences the occurrence and value of claims in many type of insurance. This is the covariate, which is considered in the application in this chapter, where the model described is applied to the problem of how to group ages in motor insurance. These claims are split into bodily injury (BI) and motor damage (MD) coverage, analysing the frequency and severity of claims separately. They are all considered independent and for each of them the algorithm set out in figure 4.8 is used.

After updating them all and deciding for each one the groups to be used, the overall risk premium is calculated by:

$$\text{Risk premium}(t) = MD_{freq}(t) \times MD_{sev}(t) + BI_{freq}(t) \times BI_{sev}(t) \qquad (5.1)$$

Where $MD_{freq}(t)$, $MD_{sev}(t)$, $BI_{freq}(t)$, $BI_{sev}(t)$ assume values depending on the age t, or in other words, on the group in which the age t is located.

The results from this model are compared with the results from Verrall and Yakoubov (1998) where fuzzy set theory was used for the same data set to also perform the aggregation into groups. This other approach considered age to be discrete instead of continuous, and the groups were defined only over the frequency of claims, with severity playing no role.

The outline of the chapter is the following. In section 5.1, the data used in the application is presented. Section 5.2 refers to chapters 2 and 4 in order to explain the type of acceptance test that is used in the different cases implemented in section 5.3. In section 5.4 the final premium and groups are defined, with a comparison in section 5.5 with the results from Verrall and Yakoubov (1998). In the last section a chapter conclusion is presented.

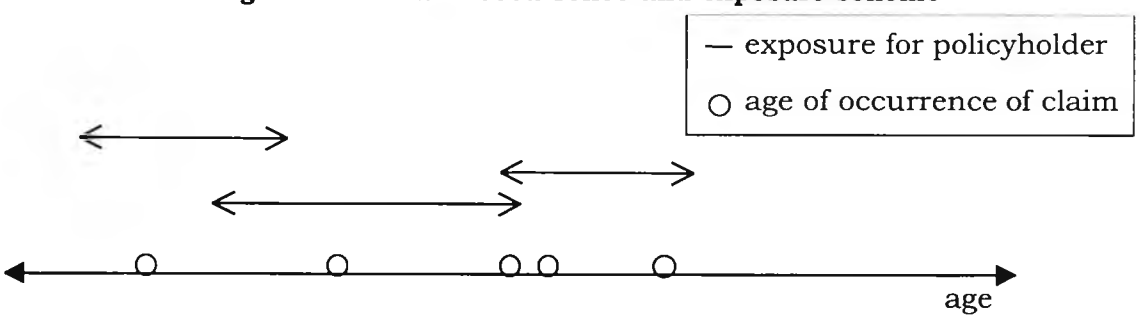
5.1 Data

The data come from a portfolio of motor insurance business with 66596 policies classified by age as well as a number of other factors. For each policy there is information related to the bodily injury frequency and severity of claims, the motor damage frequency and severity of claims and also the exposure. The effects of all the factors, except obviously age, should be removed before any analysis, but they were not sufficiently significant to justify any procedure. Hence, it is supposed that age is the only factor that influences the claims occurrence and severity.

Each policy could have more then one claim. So in formulae 4.2 and 4.5, n is related to the number of claims. For bodily injury n=624 while for motor damage n=9674.

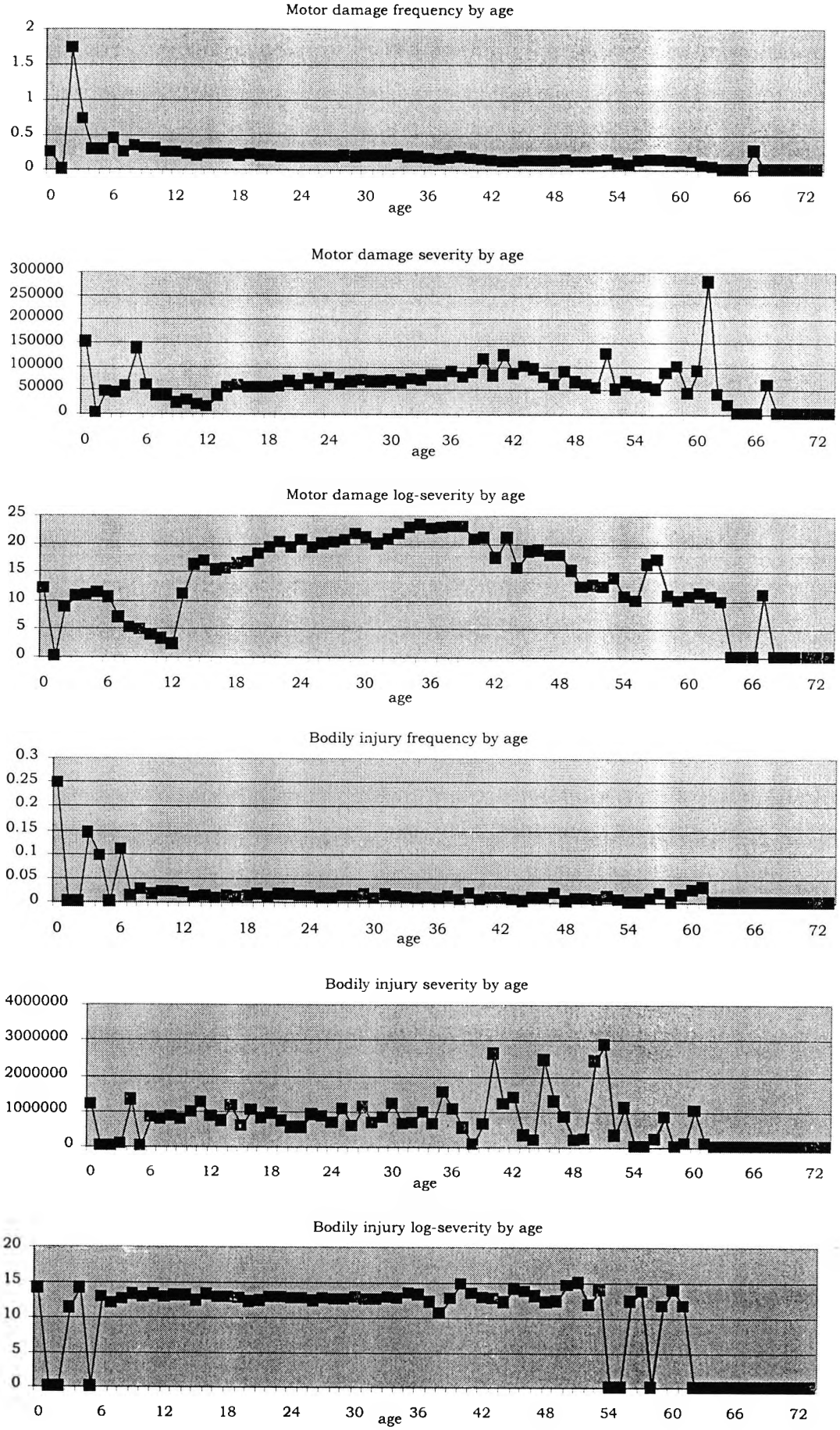
Age is recorded to two decimal places, and exposure is measured as a fraction of a year. The actual age of the policyholder when the claim occurred (or the actual moment in time that the claim occurred) was not available, and so it was necessary to use an approximation. It was supposed that claims occurrences were uniformly distributed over the exposure of each particular policy, with the respective value also equally split. The nature of the data is illustrated in figure 5.1, where circles represent the moment of occurrence of the claim and the lines the exposure, or duration, of each policy.

Figure 5.1 – Claim occurrence and exposure scheme



The maximum exposure was one year and the age was some number in the interval [19.39, 92.98]. Before starting the analysis, ages were transposed by subtracting 19.39 from all ages, changing the interval to [0, 73.59]. Three data sets were available for the analysis. The first related to the exposure, and for both bodily injury and motor damage a data set containing four pieces of information was used. The information in each of these data sets was the age of the policyholder at the moment of occurrence of a claim, the total number of claims at this age, the sum of the total claim value and the sum of the logarithm of the claim value.

Figure 5.2 – Summary plots of automobile insurance data



In order to illustrate the nature of the data, figure 5.2 shows the relative frequency (number of claims divided by exposure) and the average claim value (also in logarithm basis) in the case when age is considered as a discrete variable.

Table 5.1 also shows some statistics from the database.

Table 5.1 – Statistics of the automobile insurance data

	BI-frequency	MD-frequency	BI-severity	BI- log severity	MD-severity	MD- log severity
Mean	-	-	871,805	12.74	67,734	16.93
SD	-	-	1,080,408	1.53	82,474	10.16
Occurrence Rate	0.0124	0.1921	-	-	-	-

Before applying the model the values of the claims were also transformed, with bodily injury severity divided by 10,000 and the motor damage severity by 1,000. In this way the order of magnitude of the values were simply decreased.

5.2 Description of the models

Eight models were implemented to find a premium for motor insurance depending on ages. They follow completely the model described in chapter 4. Half were related to motor damage, with the same models also adapted to bodily injury. For each type of coverage, frequency and severity of claims were modelled separately, with two models for each of them.

Recalling from (4.15) in the previous chapter, the models change in the way the following relationship is defined:

$$g(s_j, s^*)f(l'_j) + g(s^*, s_{j+1})f(l'_{j+1}) = g(s_j, s_{j+1})f(l_j) \qquad (5.2)$$

The models related to frequency differ in the way the function $g(s_j, s_{j+1})$ is defined. For claim severity, there are also two for each type of claim, differing in the way $f(l_j)$ is defined. Such a change is related to the values that could be assumed by l_j (j=0,...,k) that in the lognormal case can be negative, invalidating the proposed function in (4.19).

In the end the following models were used, with the formulae and abbreviation also indicated in table 5.2.

Table 5.2 – Models covered in this chapter

Model	Abbreviation	$f(l_j)$	$g(s_j, s_{j+1})$
Bodily injury frequency	BIf	4.20	4.21
Bodily injury frequency weight	BIfw	4.20	4.22
Bodily injury severity	BIs	4.19	4.21
Bodily injury severity log	BIslog	4.20	4.21
Motor damage frequency	MDf	4.20	4.21
Motor damage frequency weight	MDfw	4.20	4.22
Motor damage severity	MDs	4.19	4.21
Motor damage severity log	MDslog	4.20	4.21

In each model the number (k), position (s_j, $j=1,\ldots, k$) and level (l_j, $j=0,\ldots, k$) of the jumps are updated via MCMC techniques, with a large sample as a final result. Then an analysis is performed and the groups are chosen for each parameter that is included in the risk premium.

5.3 Implementation of the models

After the eight models have been defined, their implementations can be carried out. The constant parameters should be defined, giving values to L and σ and to the set of hyper-parameters: k_{max}, λ, μ, τ, α and β. Given the range of the data, L was defined as 73.59. The Poisson rate was chosen as $\lambda=3$, and, controlling the number of jumps, k_{max} was chosen to be 30, a value which was not reached during the simulations. The values for λ and k_{max} have been shown not to highly influence the model, (see comments in Green (1995), Green and Richardson (1997) and Denison, Mallick and Smith (1998)) but it is also true that a large value for the number of groups would not summarise the data enough.

For simplicity, the values for λ and k_{max} were chosen to be the same for all eight models in table 5.2. For the values of the prior distribution parameters the values in table 5.3 were chosen in order to be as non-informative as possible. The values for the standard deviation σ were chosen based on the data.

Table 5.3 – Hyper-parameters and constants values

	BIf	BIfw	MDf	MDfw	BIs	BIslog	MDs	MDslog
α	12	12	10	10	-	-	-	-
β	1,000	1,000	200	200	-	-	-	-
μ	-	-	-	-	50	50	50	50
τ	-	-	-	-	100	100	100	100
σ	-	-	-	-	150	20	150	20

It is necessary to define all the values before starting the simulation. So, initial points must be defined, where the number of jumps were chosen to be equal to 5 for all parameters. In the same way the position of jumps were defined as 5 equally spaced jumps in (0,73.59), taking care not to have jumps at the moment of occurrence of any claim. The values for the levels of jumps were sampled at random from the prior distributions. The algorithm summarised in figure 4.8 was applied to each of the 8 models.

The simulation was updated for 10,000 steps on a Sparc ultra 1 140MHz with 64Mb memory. For each model, one step j in the algorithm in figure 4.8 took around one minute, which is rather large when compared to usual MCMC calculations. The longer time can be justified by two main reasons. Firstly, the size of the data set is quite large, since the nature of the information is at an individual level. Secondly, at the moment of updating the position of jumps, the Gibbs sampler methods requires the subdivision of $[0,L)$ into intervals with fixed exposure and number of claims, which are quite numerous.

The acceptance rate for the birth and death moves ((b) and (c) in subsection 4.2.2 respectively) are shown in table 5.4.

Table 5.4 – Type of moves and acceptance levels

Model	Kept Move(a)	Birth move(b)	Accepted	Rate	Death move(c)	Accepted	Rate
BIf	3,525	3,024	542	17.92%	3,451	542	15.71%
BIfw	3,426	3,066	447	14.58%	3,508	450	12.83%
BIs	4,689	3,851	679	17.63%	1,460	682	46.71%
BIslog	5,027	3,976	685	17.23%	997	687	68.91%
MDf	3,349	2,848	107	3.76%	3,803	109	2.87%
MDfw	3,319	2,919	137	4.69%	3,762	139	3.69%
MDs	4,210	3,880	328	8.45%	1,910	333	17.43%
MDslog	3,806	3,537	270	7.63%	2,657	274	10.31%

The rate of acceptance is around the level observed in some other papers using RJMCMC (see references in subsection 4.2 and 2.2.3) for all models, except MDf and MDfw. This poor level influenced the application of these two models as seen in their analysis in the following sections.

After obtaining the sample, we may perform the analysis. The first characteristic to consider is if and when convergence has been achieved. From the 10,000 samples, the initial 1,000 were considered as burn-in iterations (M=1,000) and discarded. The analysis of the trace for each parameter of each model for the initial 1,000 and the remaining of the sample has shown that convergence was achieved by this point. This can been seen from the visual evidence in the trace plots.

If the MCMC tests of convergence are used straightforwardly in the reversible jump case, it can result in distortions since the meaning of each parameter, apart from k, changes at each iteration. As an example, the interpretation of the position of a jump depends on the existing number of jumps. Since there is no specific test of convergence that has been especially developed to RJMCMC, it was chosen to perform the analysis by following some of the rules for the analysis of a MCMC simulation, bearing in mind that in the reversible jump case the parameters change meaning at each iteration. Generally speaking, it is expected that the whole model converges to a specific number of jumps and that through the analysis of the total sample a convergence of all remaining parameters should also be observed.

After ensuring that convergence had been achieved, the analysis of the empirical posterior distribution for each parameter was carried out. In order to minimise the errors of interpretation, an analysis of the parameter values conditional and unconditional on the number of jumps was also performed. To summarise, the procedure for this analysis is as follows:

To check convergence:

- Plot a trace of the values and see if there is a good mixing and no undesirable trend;
- Compare the two models applied to the same data set and the results given by table 5.2. Then observe the posterior distributions proposed for the same problem with these two different ways of updating.

To decide the shape and parameters values of the posterior:

- Examine the histogram and Bayes factor of the number of jumps;

- For the chosen number of jumps, analyse the behaviour of the parameters (whether they are individually consistent with the graph for the whole analysis);
- Compare the plot for each single component unconditional and conditional on the chosen number of jumps. Then compare with the two overall plots, which are given by the empirical posterior distributions of all samples for the position and level parameters.

The analyses of the results for each type of model described on table 5.1 are given in the following sections. The results are divided according to each type of coverage, bodily injury (section 5.3.1) and motor damage (section 5.3.2). After performing the analysis of the results, the pure risk premium is calculated, where the choices of $BI_{freq}(t)$, $BI_{sev}(t)$, $MD_{freq}(t)$ and $MD_{sev}(t)$ for $t \in [0,73.59]$ are given in section 5.3.3.

An individual analysis is done for each model. Estimators for the number of jumps, together with the level and moment of occurrence of each jump have to be decided. Natural estimators of the positions and levels of jumps are the mode(s) and mean of the posterior distribution for each individual component.

In the case of the number of jumps a different approach was used. In theory, the best estimator should be the one with the highest Bayes factor. However, an analysis of the moment of occurrence of each jump and the level parameters also influence the decision. The number of jumps, k, should be chosen first, since it influences the analysis of the position and level parameters. For each l_0, l_1, l_2,..., l_k, s_1, s_2,..., s_k both the conditional and non-conditional empirical posterior distribution with the chosen k is plotted. It is noted that in most cases, these two distributions are not significantly different. This shows a convergence to the posterior distribution without depending on the value of k, which is a reassuring result.

5.3.1 Analysis of bodily injury claims

The analysis for all four models for bodily injury are performed separately in the following subsections, but their summaries and parameter estimates are given in advance in table 5.5. This table includes the chosen number of jumps, together with the number of jumps with the highest Bayes factor (BF). Means and modes are shown for the level and position of each

jump, with the values calculated over the total sample for each parameter (or without conditioning on the chosen number of jumps).

It can be observed that for BIfw and BIs the chosen values for the number of jumps were different from the ones with the highest Bayes factor. This choice was based on the facts that the analyses of the level and position of jumps were pointing to the chosen values, and that their Bayes factors were not significantly smaller than the highest one. (BIfw: $BF(3,2)=1.15$, $BF(2,3)=0.87$ and BIs: $BF(0,1)=1.66$, $BF(1,0)=0.60$).

Table 5.5 – Estimated values of the parameters in bodily injury

		BIf	**BIfw**	**BIs**	**BIslog**
Number of jumps	**Chosen Value**	2	2	1	0
	Highest *BF*	2	3	0	0
Position of jumps	**Mean**	12.467, 33.283	11.618, 27.856	33.082	-
	Mode	12.585, 32.721	12.585, 32.721	38.259	-
Level parameters	**Mean**	0.01871868, 0.01239891, 0.01028067	0.01964847, 0.01343926, 0.0114657	80.57695, 127.6362	21.50548
	Mode	0.01855026, 0.01198616, 0.009062508	0.01994755, 0.01279561, 0.009650974	80.18249, 140.8127	13.63785

The result in table 5.5 is further used to define the risk premium in section 5.4. Now, the analysis of the results for bodily injury is presented for the frequency and severity models.

5.3.1.1 Bodily injury frequency models

There are two procedures that analyse the bodily injury frequency data: BIf and BIfw. Both point to a result with 2 or 3 jumps when observing the number of jumps directly, but to 2 jumps when analysing the position and level parameters. In this way 2 jumps is the one chosen.

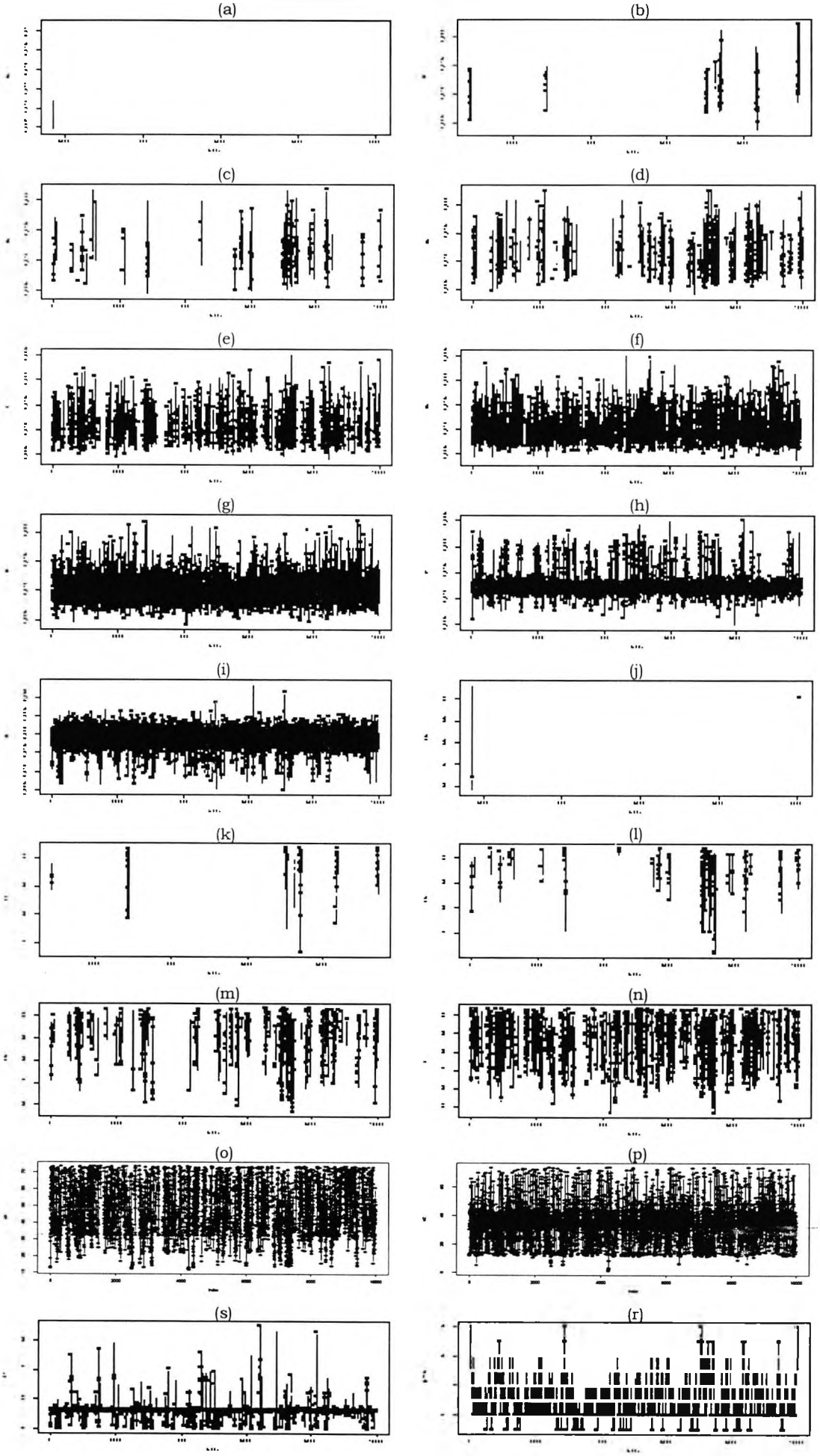
BIf

Convergence (traces in figure 5.3):

Convergence was quite fast. All traces are satisfactory, with good convergence and good mixing.

Number of jumps (plot (r)) – There is no obvious convergence, but a tendency of staying around the values 2-3. The degree of mixing is good.

Figure 5.3 – BIf trace plots

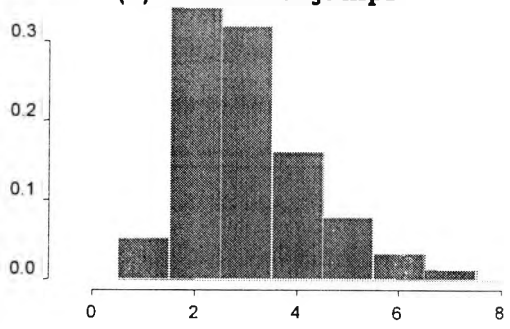


Position of jumps (plots (j) to (q)) – From s_4 to s_8 there is no clear convergence, since these parameters do not occur that often, but no problem was observed either. A tendency could be observed for s_3, s_4, s_5 and s_6 to be bigger than 30, 40, 45 and 50 respectively. s_2 is in [30, 40] and s_1 is in [10,15].

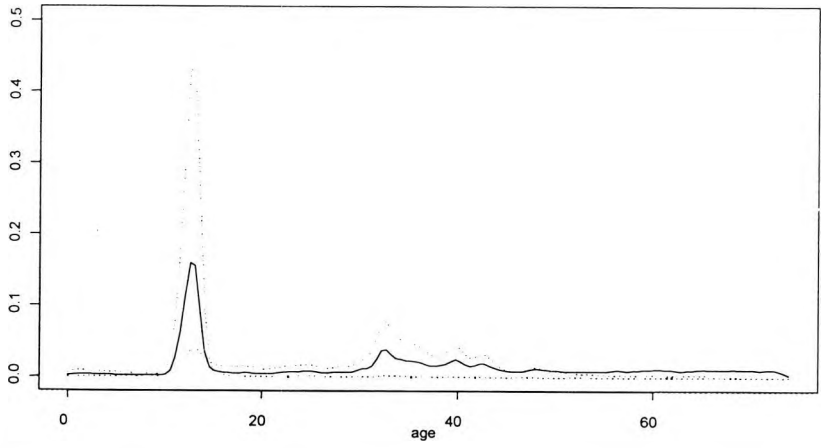
Level parameters (plots (a) to (i)) – From l_4 to l_8 there is no clear convergence, since these parameters do not occur that often, but no problem was observed either. l_3 has values around 0.010, l_2 around 0.010, l_1 around 0.012 and l_0 around 0.020. All l_i (i=0,...,3) achieved convergence and l_3 and l_2 assume roughly the same values, confirming that there is no need for the third jump.

Figure 5.4 – BIf empirical posterior distributions

(a) Number of jumps

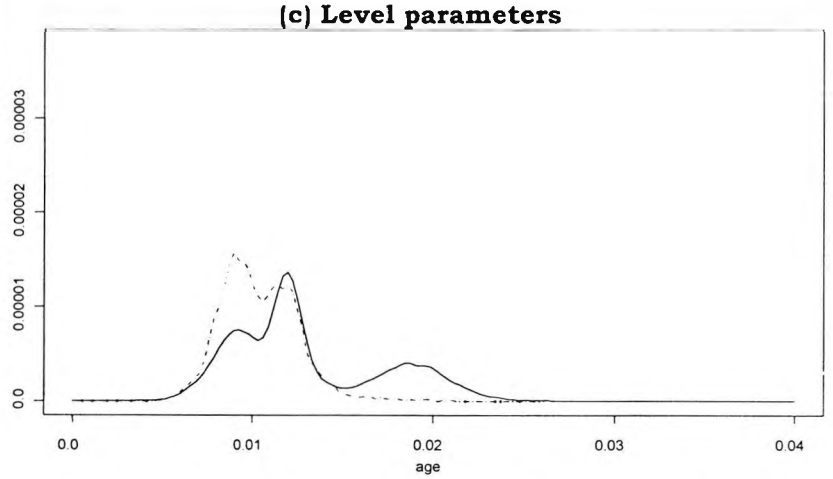


(b) Position of jumps



-·- s_1, s_2 conditional jump=2; ··· s_1, s_2 individually, — overall for all s_i i=1,...,8.

(c) Level parameters



-·- l_0, l_1, l_2 conditional jump=2; ··· l_0, l_1, l_2 individually, — overall for all l_i i= 0,...,8.

Analysis of empirical posterior distributions (Figure 5.4):

Number of jumps – 2 is definitely the highest mode and the chosen value, although the value 3 is also highly probable.

Position of jumps – The empirical a posteriori distributions for all observed positions of jumps has two modes, with the first one (s_1) very well defined, differently from the not so clear second one (s_2). When plotting the individual plot for each position parameter, the a posteriori distribution for s_2 has a more highly concentrated mode around 32. s_1 has the same posterior for all graphs with mode around 12.

Level parameters – The total sample has three modes, where the second one (l_1) is the most highly concentrated with values around 0.013. The first one (l_0) has a more defined mode (0.02) when plotting the individual graph. The last one (l_2) is highly influenced by the values of l_1, assuming values around 0.008.

BIfw

Convergence (traces in figure 5.5):

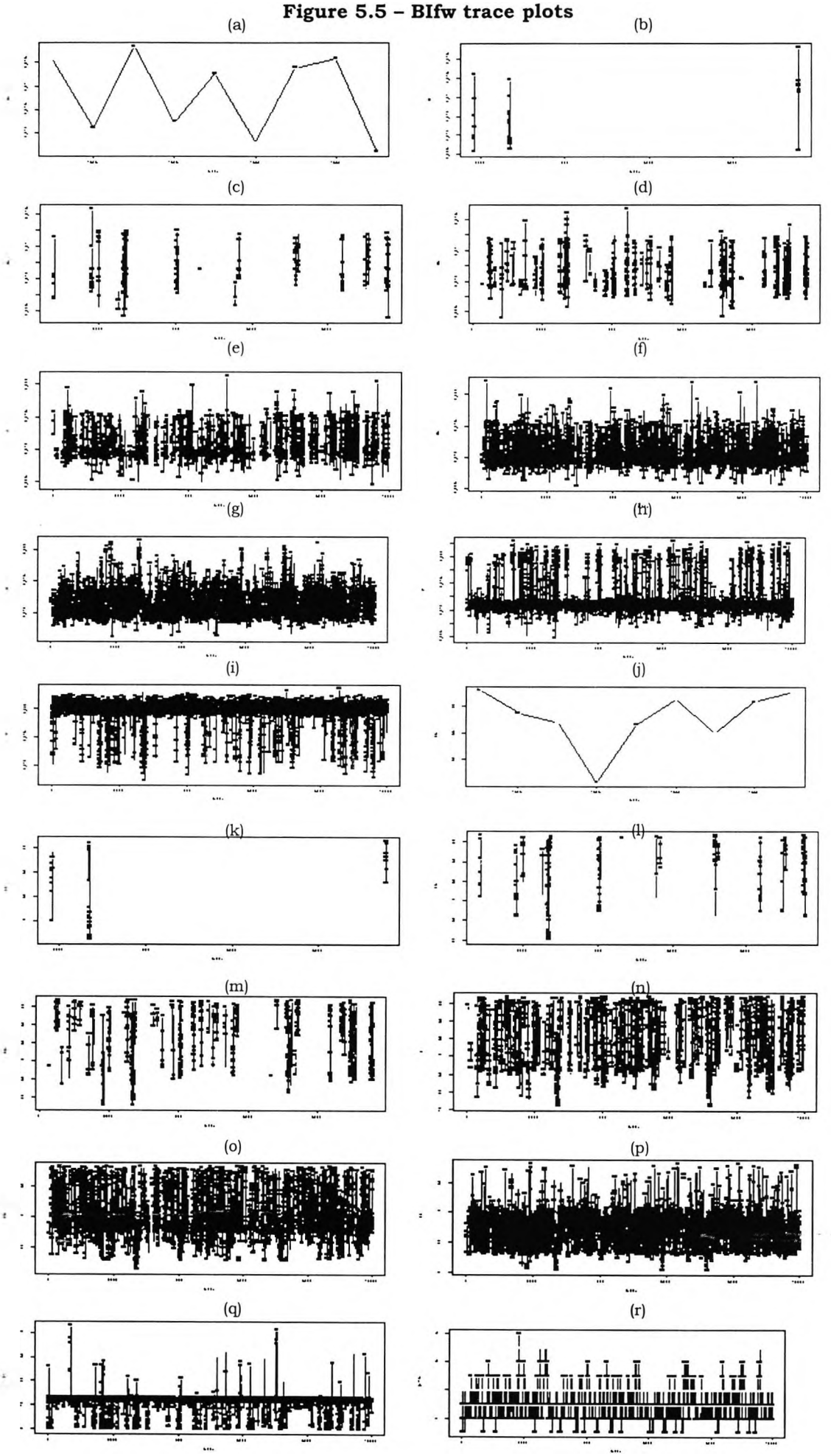
Convergence was quite fast. All traces are satisfactory, but sometimes there is no really good mixing. The convergence and mixing seems poorer than BIf.

Number of jumps (plot (r)) – Some convergence to values 2 and 3.

Position of jumps (plots (j) to (q)) – From s_4 to s_8 there is no clear convergence, since these parameters do not occur that often, but no problem was observed either. s_3 tends to be larger than 30, s_2 in [15, 40] and s_1 in [10,15].

Level parameters (plots (a) to (i)) – From l_4 to l_8 there is no clear convergence, since these parameters do not occur that often, but no problem

Figure 5.5 – BIfw trace plots



was observed. l_3 has values around [0.010,0.015], but not with very obvious convergence. l_2 has a similar behaviour, showing that the third jump is not really necessary. l_1 converges to 0125, but also with a high probability on values around 0.020. l_0 achieves convergence to values around 0.020.

Analysis of empirical posterior distributions (Figure 5.6):

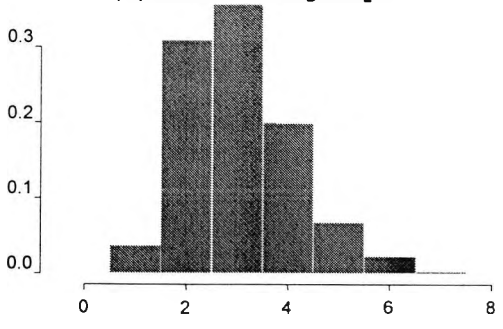
Number of jumps – The value 3 has the highest probability, but the value 2 is also high. 2 is the number of jumps chosen, based on the values for the position and level of jump that pointed to the existence of only 3 groups (or 2 jumps).

Position of jumps – The overall graph shows two modes. s_2 is located at values around 32 and it is much more concentrated when in the conditional distribution to k=2. s_1 has a very high probability of being around 13 in all graphs.

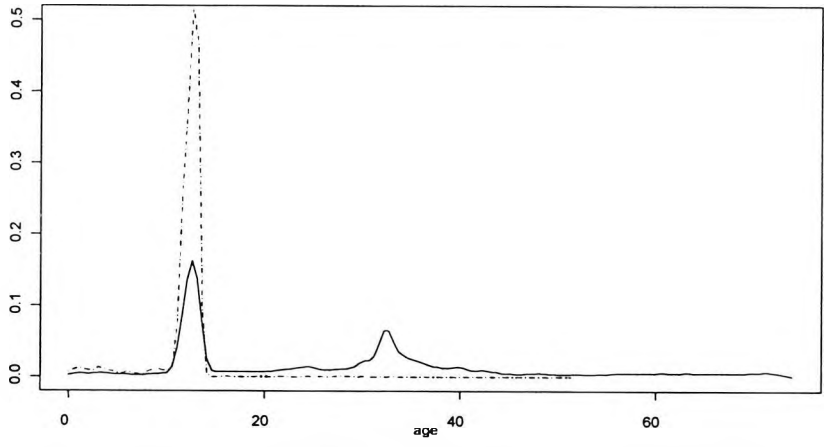
Level parameters – The overall graph points to three modes. l_2 is related to the first one, around 0.009, that is clearer when plotting the conditional distribution. l_1 is even higher concentrated around 0.012 when the conditional distribution is plotted, compared to the overall one. l_0 has also a clearer mode (0.02) in the conditional distribution.

Figure 5.6 – BIfw empirical posterior distributions

(a) Number of jumps

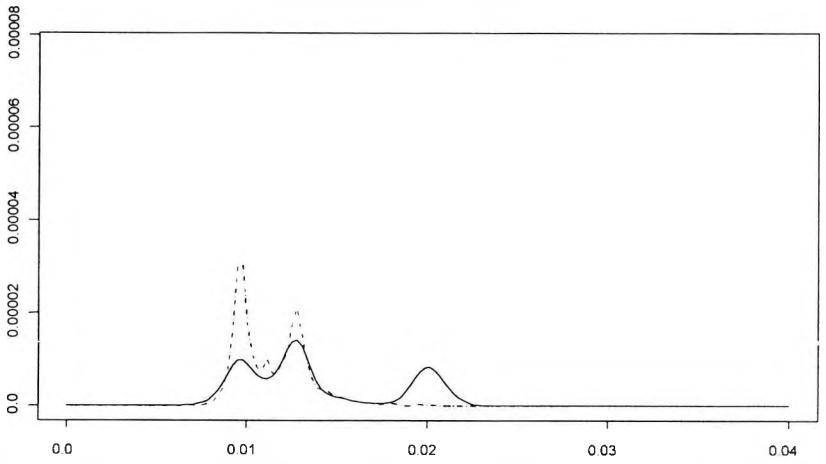


(b) Position of jumps



-·- s_1, s_2 conditional jump=2; ··· s_1, s_2 individually, — overall for all s_i i=1,...,8.

(c) Level parameters



--- l_0, l_1, l_2 conditional jump=2; ··· l_0, l_1, l_2 individually, — overall for all l i= 0,...,8.

5.3.1.2 Bodily injury severity models

There are two procedures for analysing the bodily injury severity data: BIs and BIslog. In the results for BIs a jump was chosen, but in BIslog no jump is indicated. The results had mostly a good mixing and reasonable convergence.

BIs

Convergence (traces in figure 5.7):

Convergence was quite fast. All traces are satisfactory, with good convergence and good mixing.

Number of jumps (plot (l)) – The values are very spread without any obvious convergence. The number of jumps has a high concentration on the values 0, 1 and 2.

Position of jumps (plots (g) to (k)) – s_3, s_4 and s_5 are not frequent enough to draw any conclusion. s_2 has a clearly tends to be larger than 30, without any clear convergence. s_1 is definitely in [30,40], with good mixing.

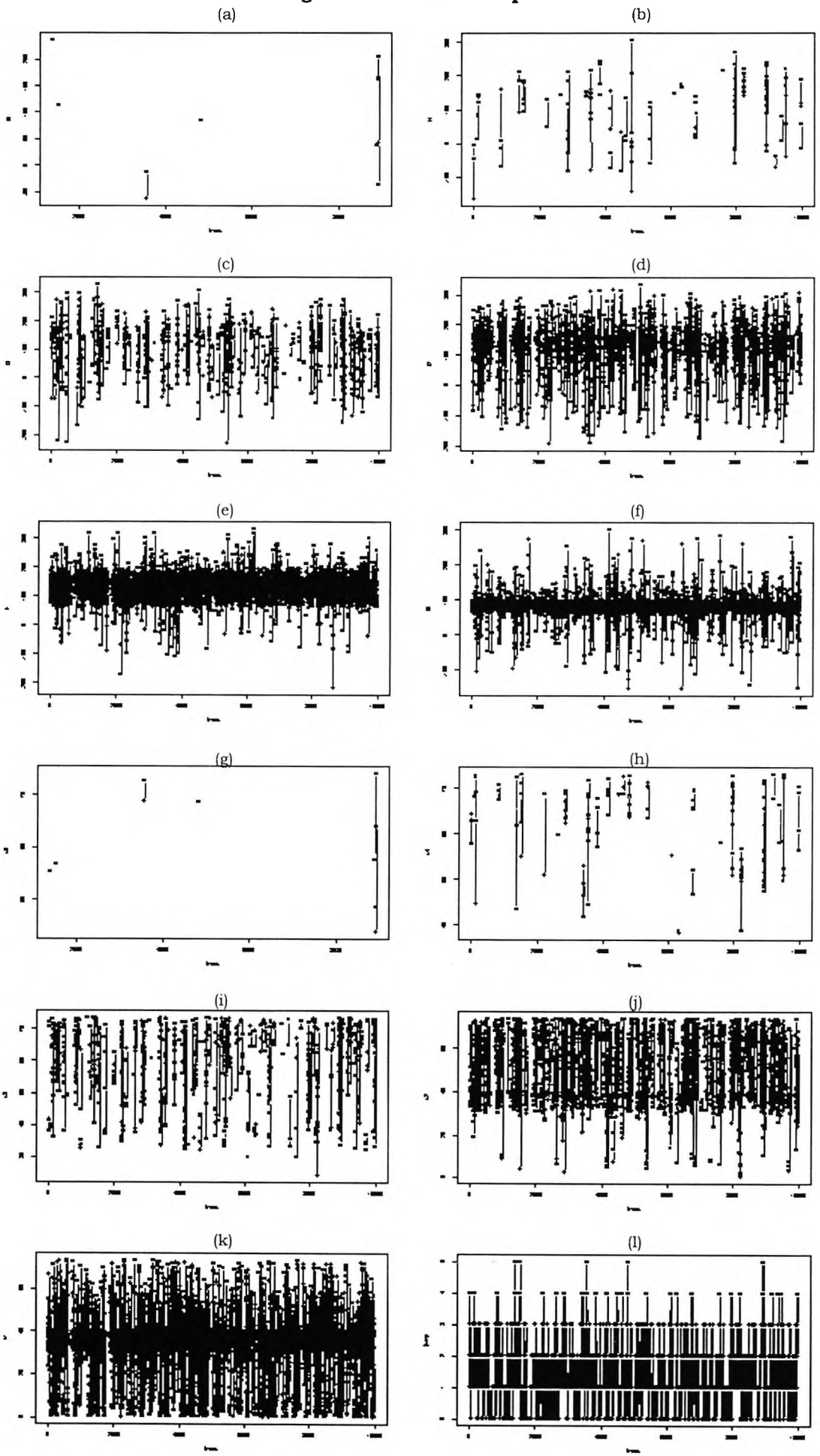
Level parameters (plots (a) to (f)) – l_3, l_4 and l_5 are not frequent enough to draw any conclusion. l_2 has a good mixing, but without any clear convergence. l_1 has values in [100,200], with convergence to this interval. l_0 converges to [60,100].

Analysis of empirical posterior distributions (Figure 5.8):

Number of jumps – The number of jumps with highest probability is 1, followed by 0.

Position of jumps – The overall graph has two clear modes (37 and 39), with very heavy tails. The individual plots seem more concentrated that the overall plot.

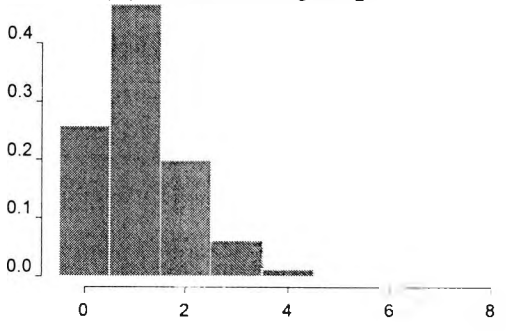
Figure 5.7 – BIs trace plots



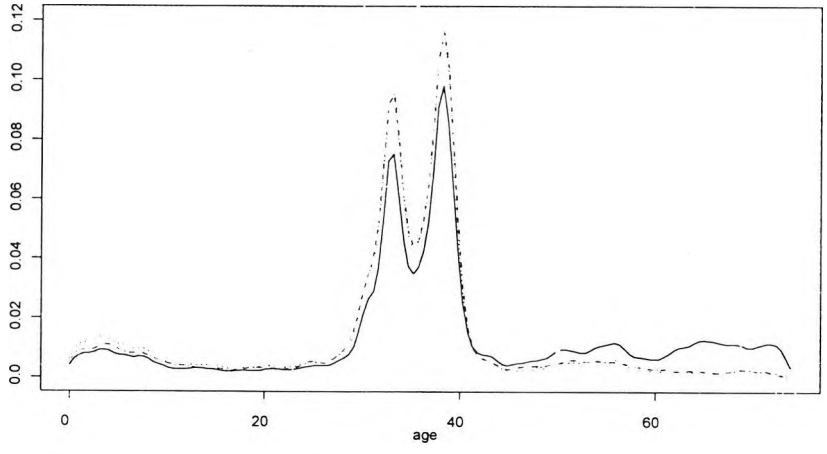
Level parameters – There are two modes in the overall plot, with the second mode more "spread" than the first. The individual graphs for l_0 have a nice shape, being highly concentrated around 70. In the case of l_1 the shape of the distribution is not so concentrated, with values around 140.

Figure 5.8 – BIs empirical posterior distributions

(a) Number of jumps

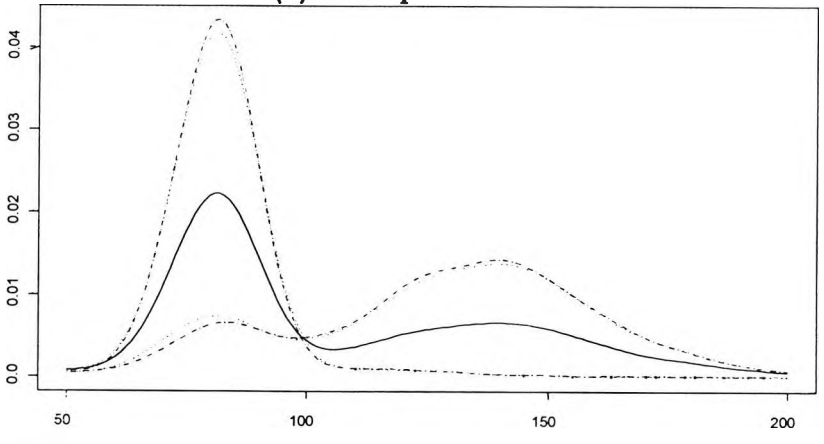


(b) Position of jumps



-·- s_1 conditional jump=1; ··· s_1 individually, — overall for all s_i i=1,...,5.

(c) Level parameters



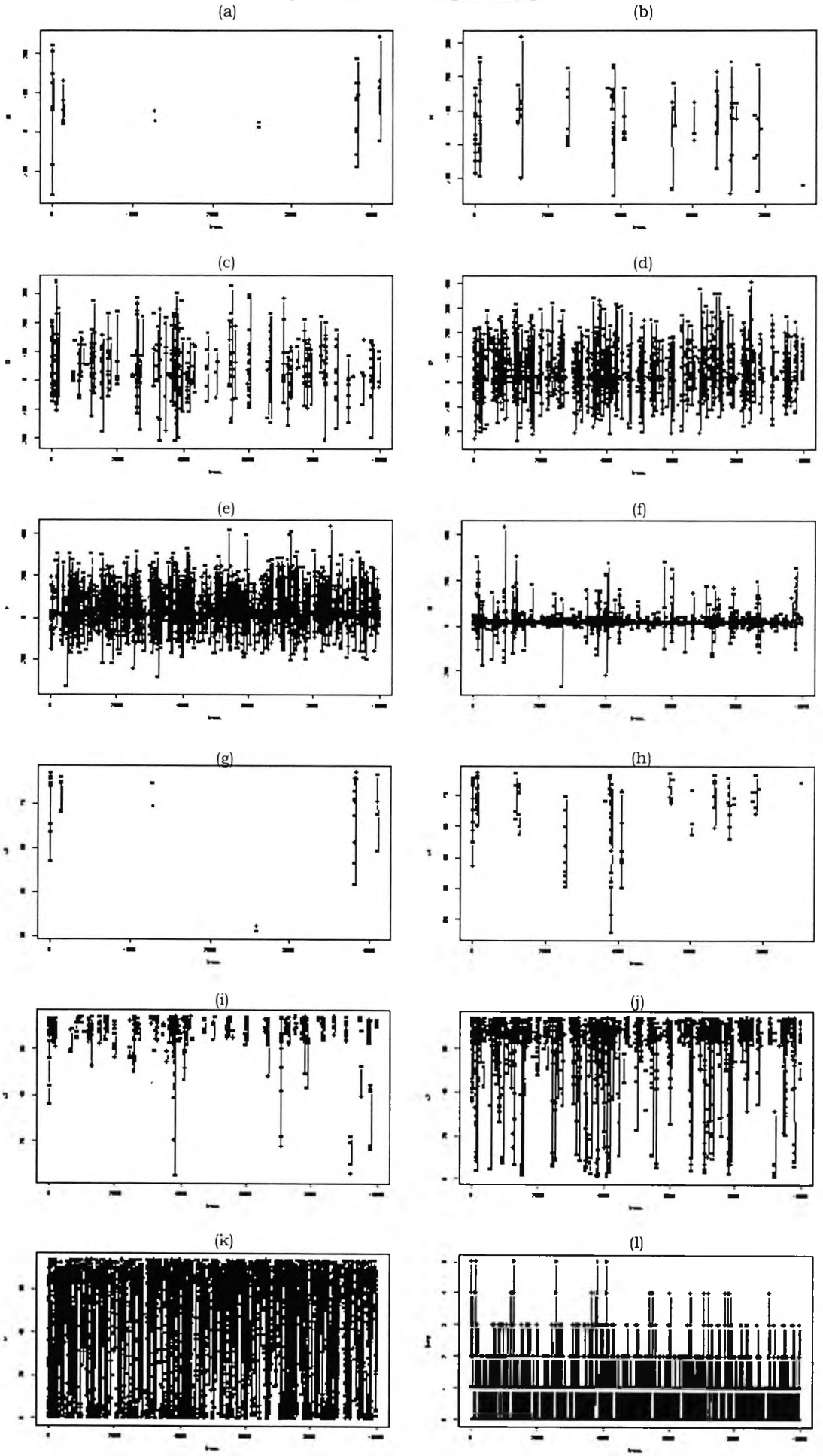
-·- l_0, l_1 conditional jump=1; ··· l_0, l_1 individually, — overall for all l_i i= 0,...,5.

BIslog

Convergence (traces in figure 5.9):

All the traces show a reasonable mixing, but convergence was not achieved for the position of jumps.

Figure 5.9 – BIslog trace plots



Number of jumps (plot (l)) – The number of jumps goes from values 0 to 5, with a higher concentration around 0-1.

Position of jumps (plots (g) to (k)) – No clear convergence was observed, where jumps from s_2 onwards were not that frequent. s_1 seems to have two areas of higher sampling: close to 0 and then to 60.

Level parameters (plots (a) to (f)) – All the ranges are very spread for all level parameters. It seems that convergence is achieved around 0 for all the levels.

Analysis of empirical posterior distributions (Figure 5.10):

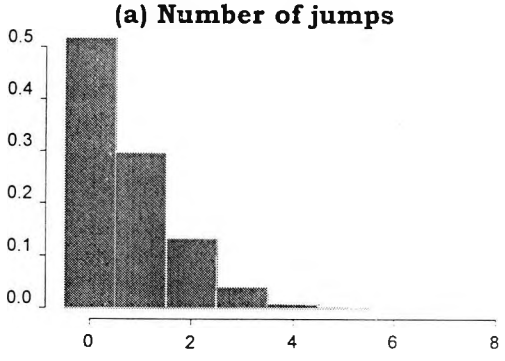
Number of jumps – The value for the number of jumps with highest posterior probability is 0.

Position of jumps – The posterior distribution for the position of jumps has a strange shape, here the modes for the position of jumps are close to the borders.

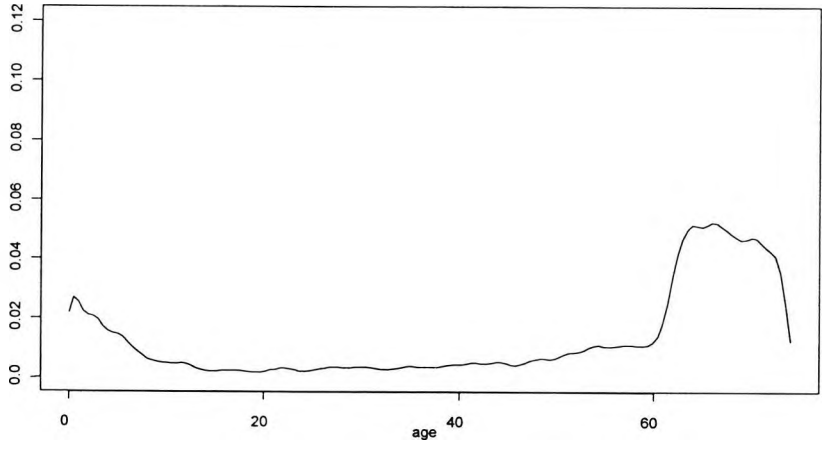
Level parameters – The posterior distribution for the level parameters is similar to a normal distribution with mean (mode) around 18.

Figure 5.10 – BIslog empirical posterior distributions

(a) Number of jumps

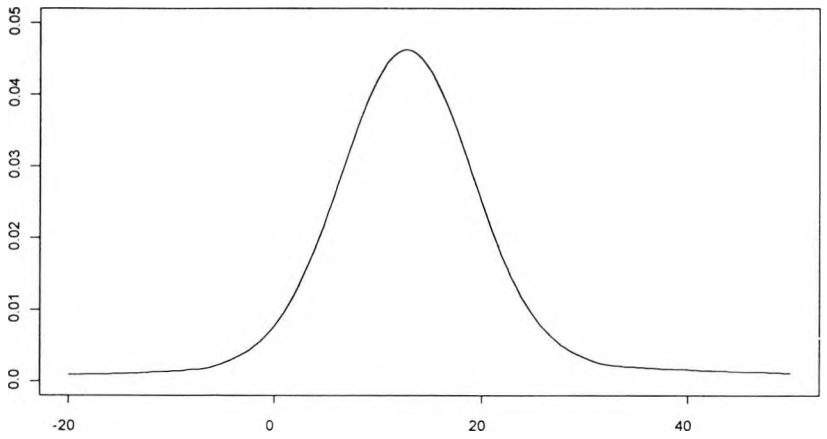


(b) Position of jumps



— overall for all s_i i= 1,...,5.

(c) Level parameters



— overall for all l_i i= 0,...,5.

5.3.2 Analysis of motor damage claims

Again, the analysis for all four models for motor damage are performed separately in the following subsections, but their summaries and parameter estimates are given in advance in table 5.6. This table is similar to the one in section 5.3.1 for bodily injury. In contrast to the previous analysis, here all the chosen values for k are the ones with the highest Bayes factors.

Table 5.6 – Estimated values of the parameters in motor damage

		MDf	MDfw	MDs	MDslog
Number of jumps	**Chosen Value**	3	3	1	1
	Highest *BF*	3	3	1	1
Position of jumps	**Mean**	12.899, 33.945, 56.36827	14.76384, 39.49449, 61.62486	13.20706	12.71871
	Mode	13.592, 40.272, 62.422	12.58503, 35.7415, 62.42177	13.59184	13.59184
Level parameters	**Mean**	0.2666808, 0.19588, 0.1369522, 0.06396099	0.260086, 0.1876084, 0.119178, 0.0468519	28.2927, 60.66479	5.022029, 15.64755
	Mode	0.2656115, 0.1917391, 0.1292134, 0.04307191	0.2662027, 0.1921859, 0.1285799, 0.03958556	23.94861, 66.32374	5.492665, 18.80877

These statistics are further used to define the risk premium in section 5.4. Before that, the analysis of the empirical posterior distributions for all four models using motor damage data is presented in the following subsections 5.3.2.1 and 5.3.2.2, for frequency and severity data respectively.

5.3.2.1 Motor damage frequency models

There are two procedures for analysing the motor damage frequency data: MDf and MDfw. Both point toward the same result: 3 jumps, but their mixing is really poor.

MDf

Convergence (traces in figure 5.11):

All the traces really have a bad mixing, and they are composed of big blocks of values. One of the blocks is more frequent than the others, and it could be argued that in this case a local convergence is observed.

Number of jumps (plot (n)) – The mixing was not as good as seen in the other models. The number of jumps seems fixed on the value 3.

Position of jumps (plots (h) to (m)) – s_4, s_5 and s_6 are not frequent enough to draw any conclusion. s_3 has two groups around 62 and around 40. s_2 has blocks around 20, 35 and 40. s_1 seems to converge to values smaller than 15.

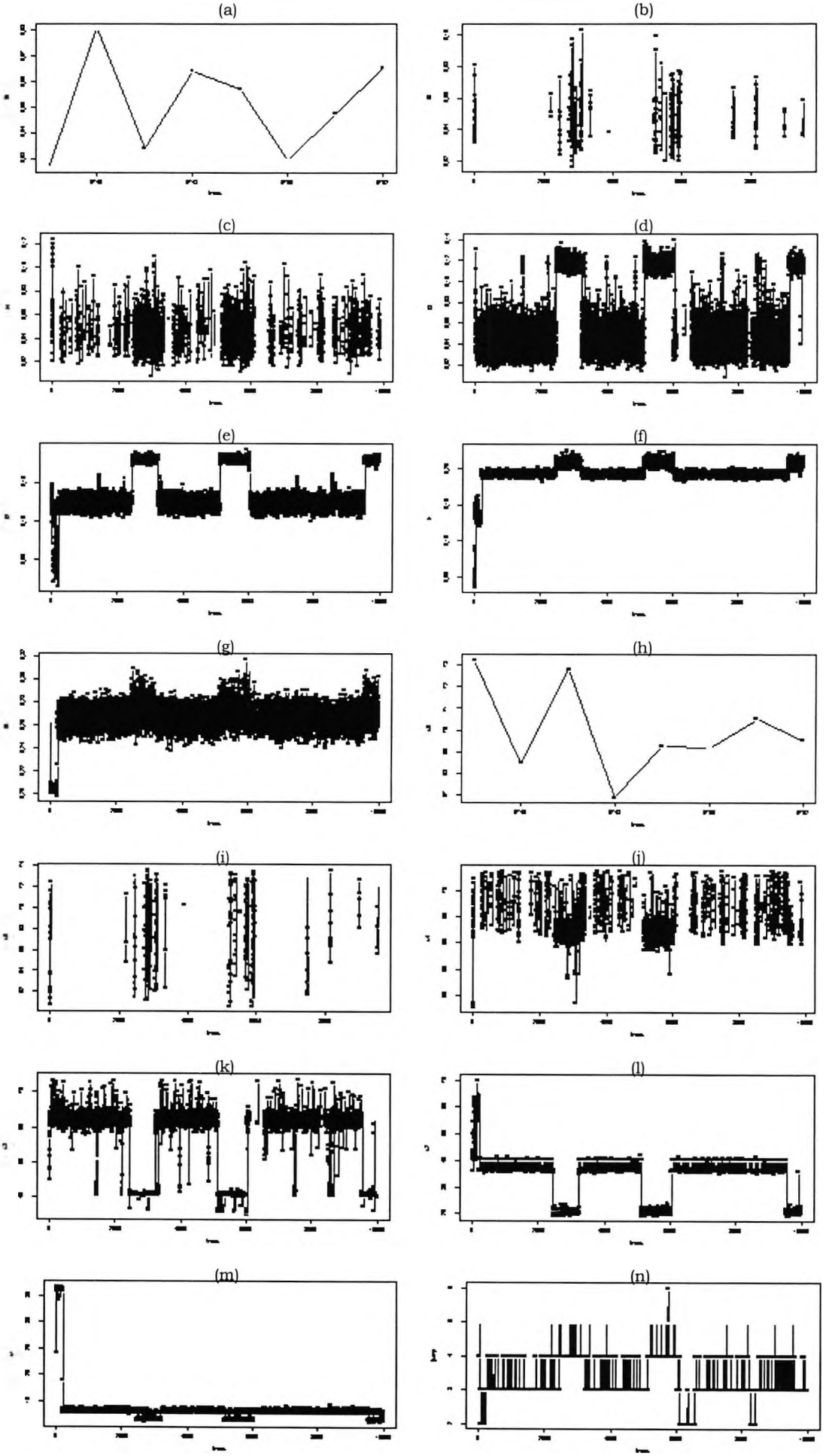
Level parameters (plots (a) to (g)) – l_4, l_5 and l_6 are not frequent enough to draw any conclusion. l_3 has two main blocks; 0.12 when s_3 is close to 40 and 0.04 when s_3 is around 60. l_2 has two blocks depending on s_2: when s_2 is around 20, l_2 is close to 0.18, when s_2 is around 35/40, l_2 is close to 0.13. l_1 is around 0.20 when s_1 is closer to 12, and then a bit bigger when s_1 is smaller. l_0 is the one that achieves the best convergence, around 0.27.

Analysis of empirical posterior distributions (Figure 5.12):

Number of jumps – The value 3 has the highest probability, as expected from the trace plot.

Position of jumps – The overall graph points to five modes well defined (approximate values 12, 20, 37, 40, 62). s_3 has 2 modes (40 and 62) when plotting the individual plot without conditioning on the number of jumps being equal 3. When conditioning on the number of jumps, there is an absolute maximum at 62. s_2 always has two strong modes 37 and 40, with some high probability on 20 as well. s_1 has 2 modes (12.75 and 13.7), while the conditional distribution has one stronger mode at 12.75. s_2 brings the greatest problem for recognition.

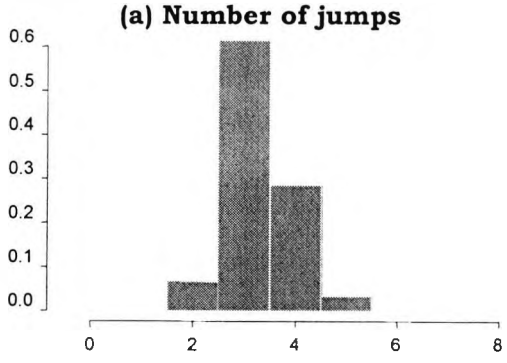
Figure 5.11 – MDf trace plots



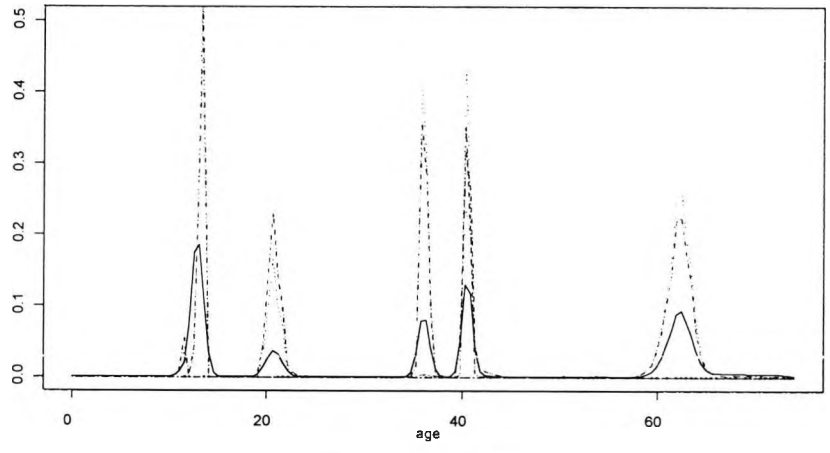
Level parameters – The overall graph points to four main areas of higher probability. The first one, related to l_3, has two modes in the individual graph, around 0.045 and 0.120, with the conditional density pointing only to the first one. l_2 has three modes when individually plotted (0.11, 0.13 and 0.18), differently from the conditional graph where the two first are stronger. l_1 has two modes (0.195 and 0.210) and l_0 has mode 0.27, both with the same behaviour in any graph.

Figure 5.12 – MDf empirical posterior distributions

(a) Number of jumps

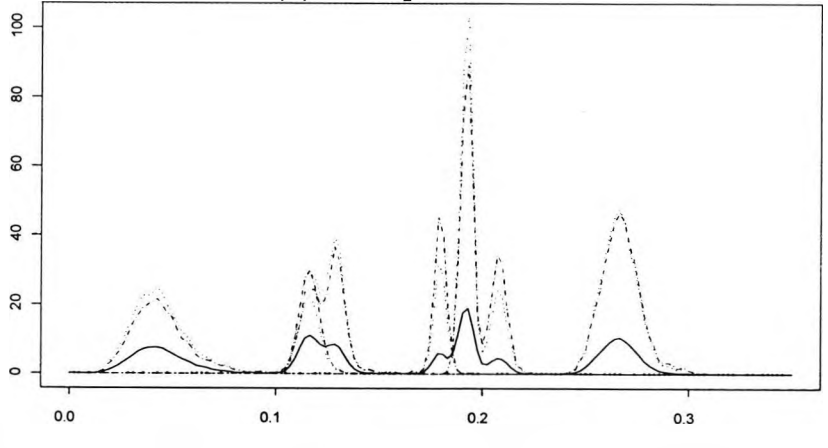


(b) Position of jumps



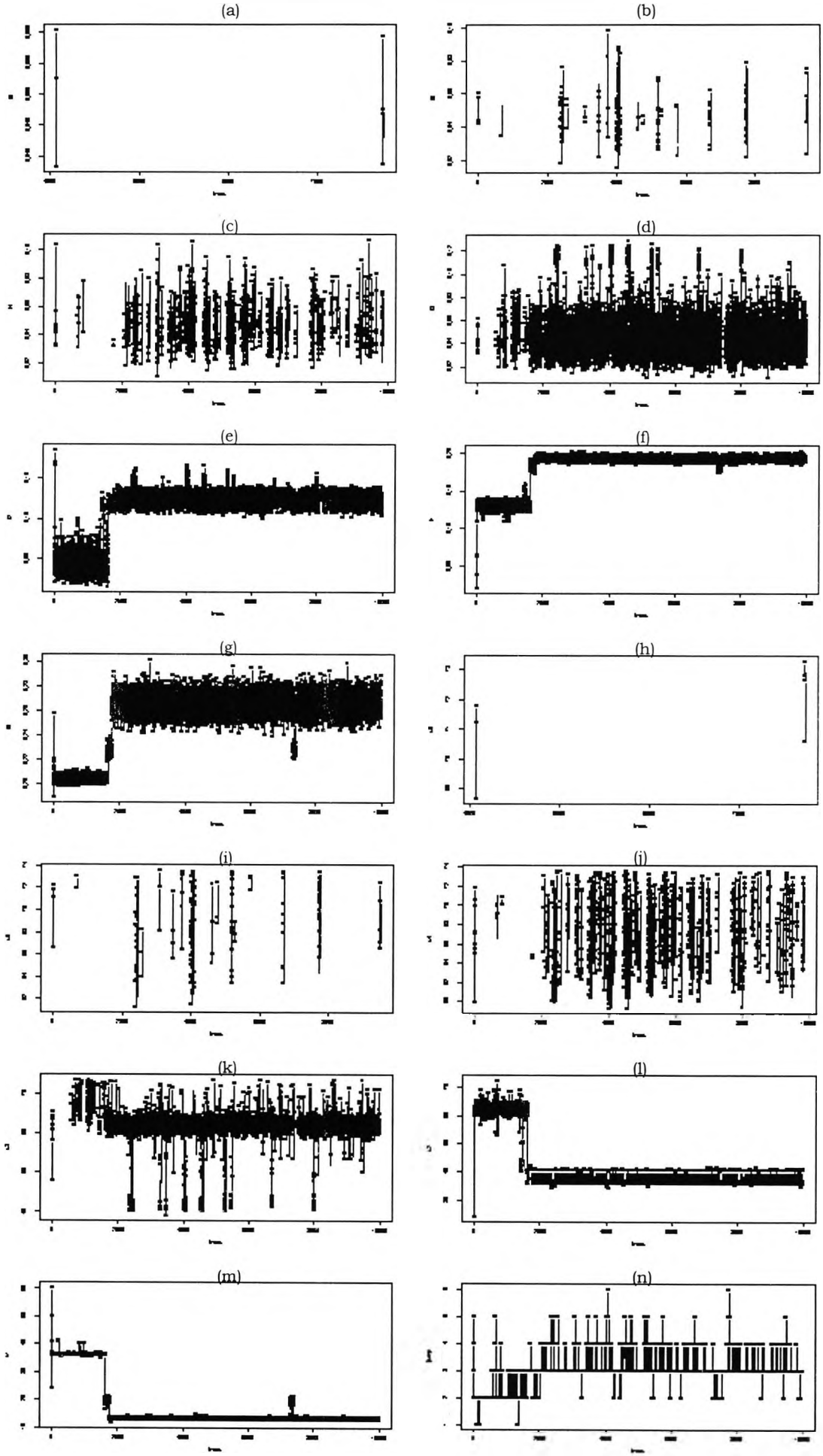
-·- s_1, s_2, s_3 conditional jump=3; ··· s_1, s_2, s_3 individually, — overall for all s_i i=1,...,6.

(c) Level parameters



-·- l_0, l_1, l_2, l_3 conditional jump=2; ··· l_0, l_1, l_2, l_3 individually, — overall for all l_i i= 0,...,6.

Figure 5.13 – MDfw trace plots



MDfw

Convergence (traces in figure 5.13):

Again all the traces have a really bad mixing, being a collection of long blocks of values. The overall result is quite similar to the previous one, confirming the conclusions.

Number of jumps (plot (n)) – As before, the mixing is poor, and the values are closer to 3.

Position of jumps (plots (h) to (m)) – s_4, s_5 and s_6 are not frequent enough to draw any conclusion. s_3 seems the one with the best mixture, being definitely around 62. s_2 has three clear blocks: an initial one around 60, then varying in between 37 and 40. s_1 has blocks starting in 37 and then moving to values around 13.

Level parameters (plots (a) to (g)) – l_4, l_5 and l_6 are not frequent enough to draw any conclusion. l_3 has quite good mixing around 0.04. l_2 starts on values around 0.05 and then goes to 0.12 and stays there. l_1 again starts at values around 0.12, but goes to 0.19 with l_0 starting around 0.21 and then moving to 0.27.

Analysis of empirical posterior distributions (Figure 5.14):

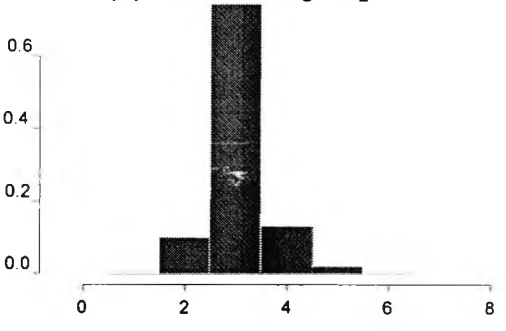
Number of jumps – As in the previous model, 3 is the number of jumps with highest probability.

Position of jumps – Now there are four modes in the overall graph, and the previous mode with value around 20 disappears. The conditional distributions are highly concentrated, showing modes around the following values for the respective positions of jump: s_3 (62), s_2 (37 and 40) and s_1 (13).

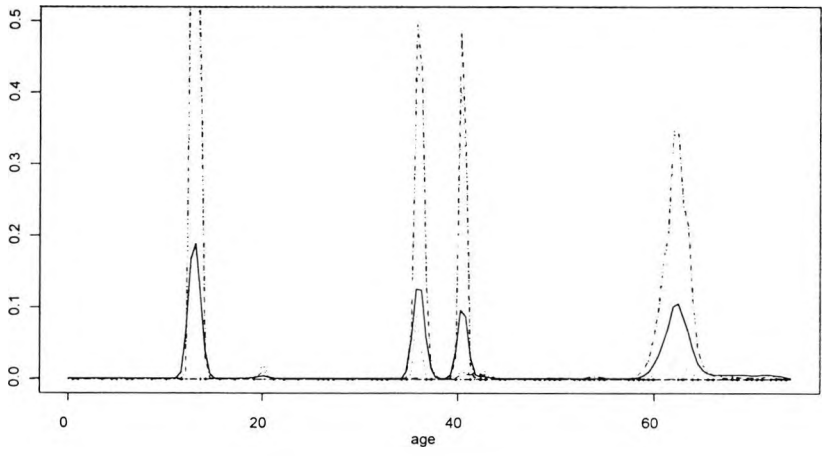
Level parameters – The overall and conditional distributions have five modes. The first (around 0.045) has the lowest probability, being related to l_3. l_2 has two (0.12 and 0.13), l_1 has one (0.19) and l_0 has one mode (0.27).

Figure 5.14 – MDfw empirical posterior distributions

(a) Number of jumps

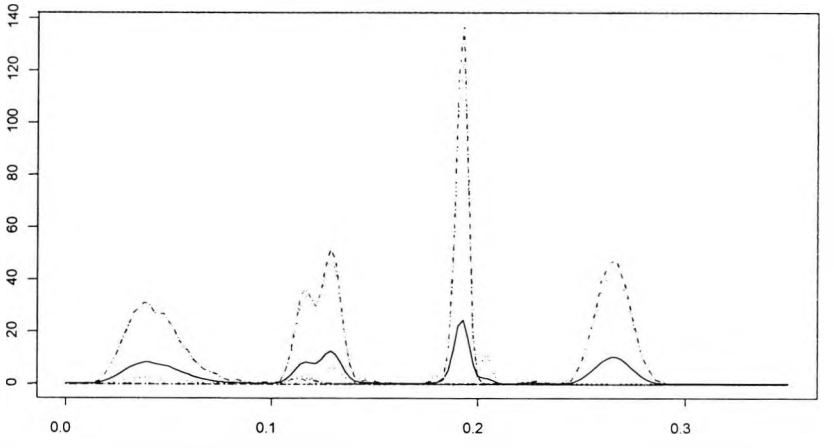


(b) Position of jumps



-·- s_1, s_2, s_3 conditional jump=3; ··· s_1, s_2, s_3 individually, — overall for all s_i i=1,...,6.

(c) Level parameters



-·- l_0, l_1, l_2, l_3 conditional jump=2; ··· l_0, l_1, l_2, l_3 individually, — overall for all l_i i= 0,...,6.

5.3.2.2 Motor damage severity models

There are two procedures analysing the motor damage severity data: MDs and MDslog. They both indicate only one jump.

MDs

Convergence (traces in figure 5.15):

Number of jumps (plot (l)) – The degree of mixing is good, with values being concentrated around 1-2.

Position of jumps (plots (g) to (k)) – s_3, s_4 and s_5 are not frequent enough to draw any conclusion. s_2 has a clear tendency of being in [20,40] and s_1 around 18.

Level parameters (plots (a) to (f)) – l_3, l_4 and l_5 are not frequent enough to draw any conclusion. l_2 has a convergence to values around 100, l_1 to values around 60 and l_0 to values around 25.

Figure 5.15 – MDs trace plots

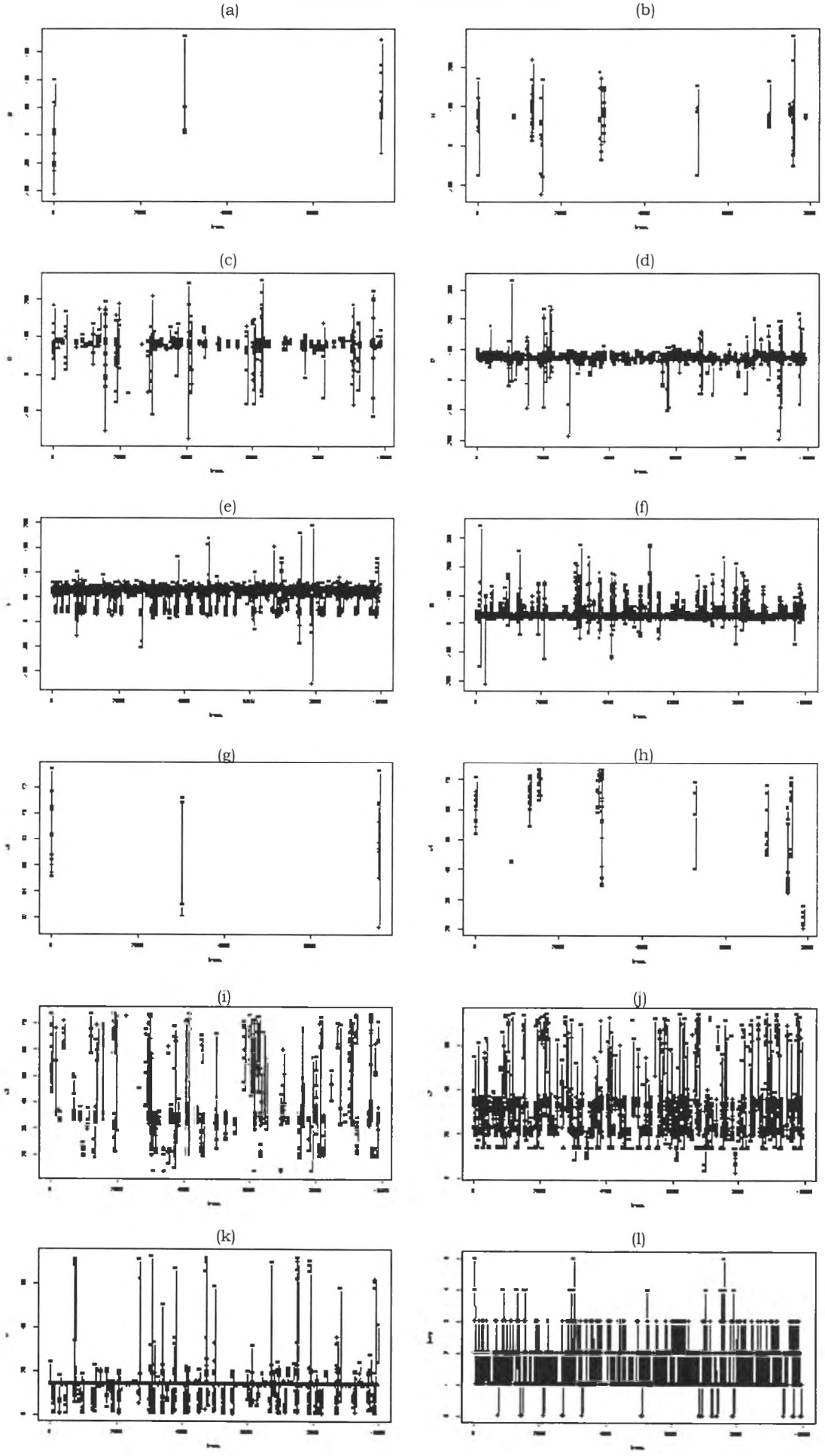
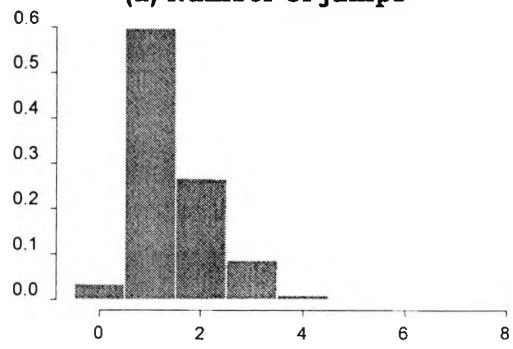
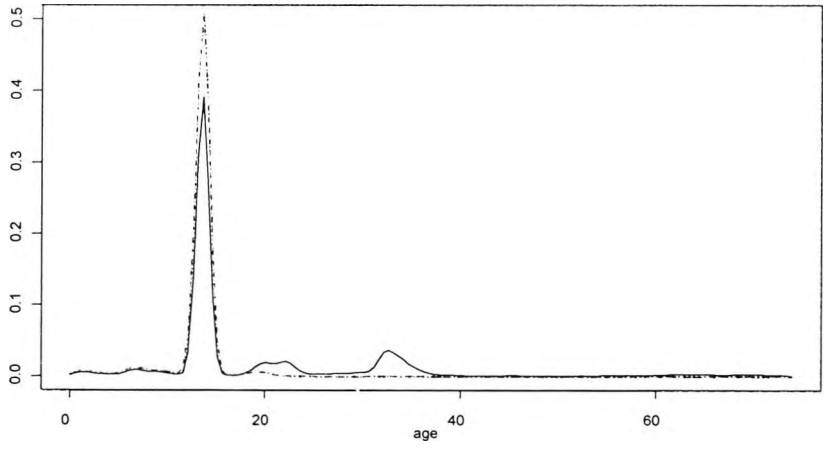


Figure 5.16 – MDs empirical posterior distributions

(a) Number of jumps

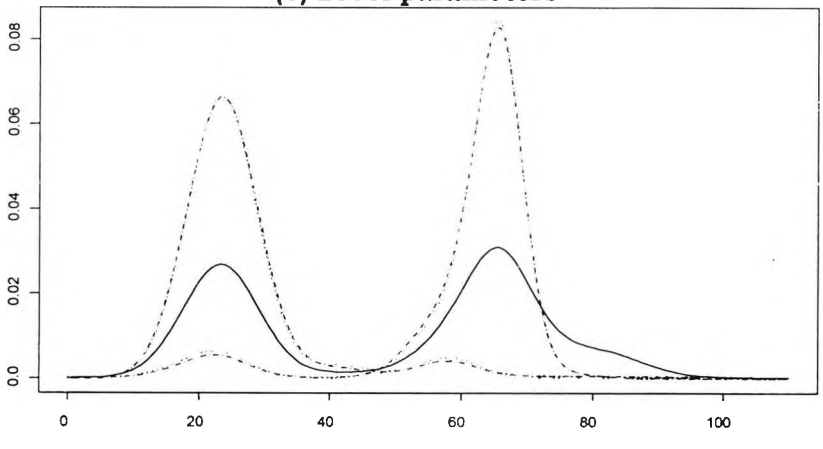


(b) Position of jumps



-·- s_1 conditional jump=1; ··· s_1 individually, — overall for all s_i i=1,...,5.

(c) Level parameters



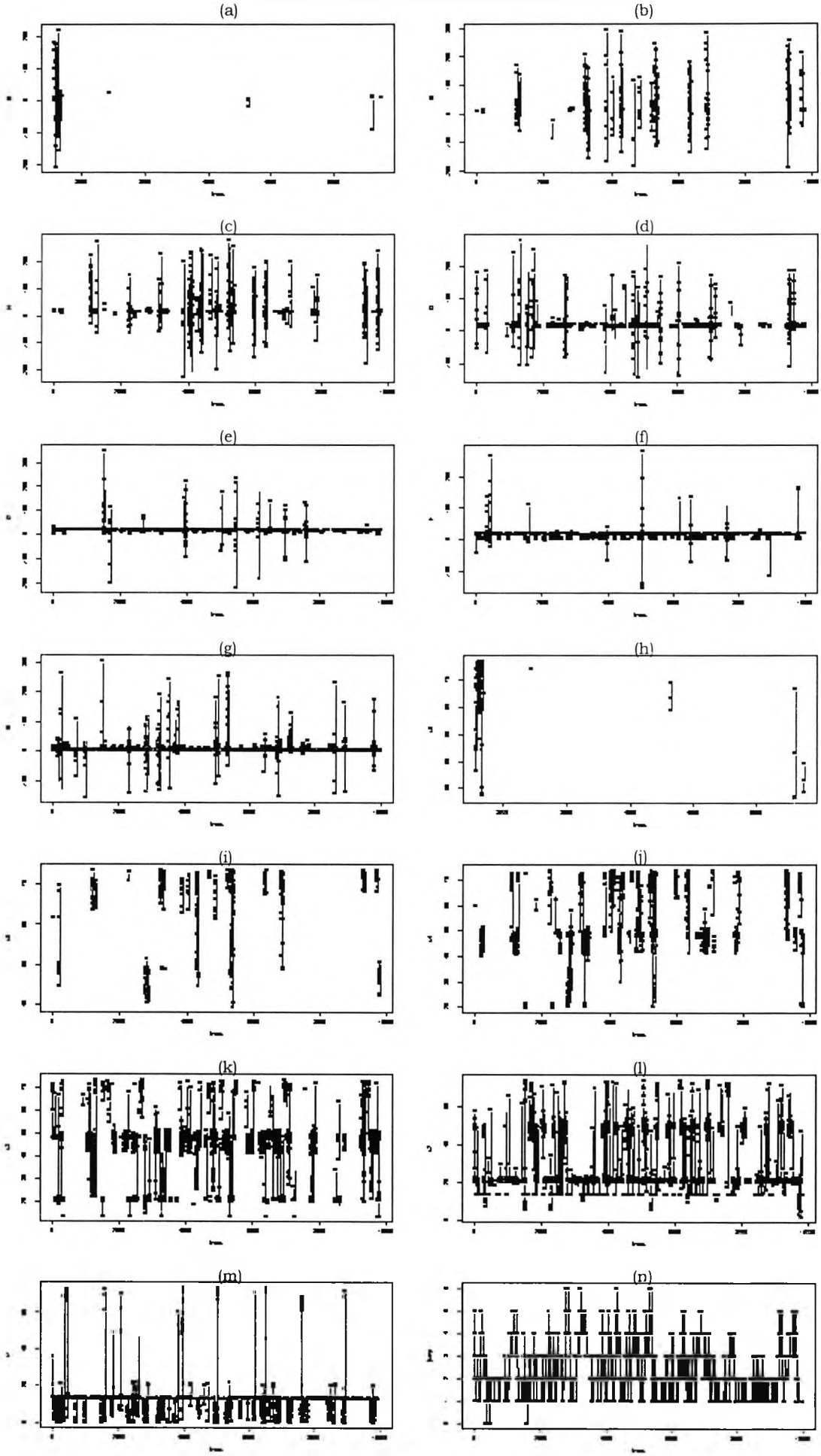
-·- l_0, l_1 conditional jump=1; ··· l_0, l_1 individually, — overall for all l_i i= 0,...,5.

Analysis of empirical posterior distributions (Figure 5.16):

Number of jumps – The number of jumps with the highest probability is the value 1.

Position of jumps – In all graphs it is possible to observe one absolute mode, with value around 13. The conditional distribution is highly concentrated around the mode.

Figure 5.17 – MDslog trace plots



Level parameters – The overall graph shows two modes, with the conditional distributions highly concentrated on them. l_0 is highly concentrated on values around 25 and l_1 is highly concentrated around 80.

MDslog

It is difficult to make a decision since the number of jumps indicated is 1 or 2, but there are three modes in the overall graph for the position of jumps. In the end one jump was chosen, by analysing the overall graph for the level parameters.

Convergence (traces in figure 5.17):

Number of jumps (plot (n)) – The degree of mixing is good, with values around 1 and 2.

Position of jumps(plots (h) to (m)) – s_3, s_4, s_5 and s_6 are not frequent enough to draw any conclusion, but these parameters do have some groups pointing to a bad mixing. s_2 has a clear tendency of having values 15, 20 and 50, but without any clear convergence. For s_1, the clear tendency is of being 15 or smaller.

Level parameters (plots (a) to (g)) – l_3, l_4, l_5 and l_6 are not frequent enough to draw any conclusion. All graphs have a large range, making it difficult to reaffirm convergence. l_2 would tend to values around 20, l_1 around 20 and 0, and l_0, 0.

Analysis of empirical posterior distributions (Figure 5.18):

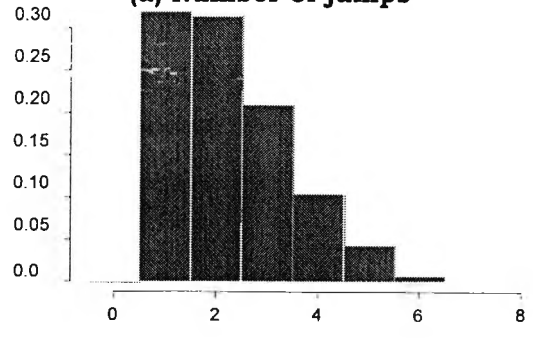
Number of jumps – 1 and 2 are the most probable number of jumps. The chosen is 1, given the observed values of position and level parameters.

Position of jumps – The overall graph indicates 3 modes with s_1 around the value 15, s_2 around the value 20 and s_3 around the value 47.

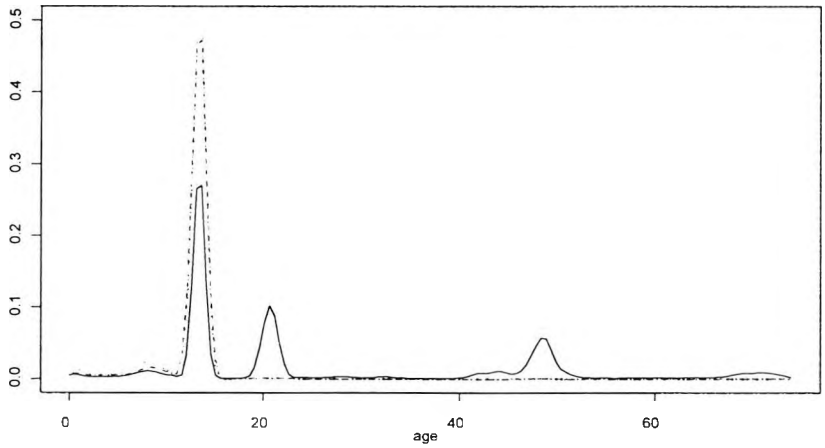
Level parameters – The overall graph shows two modes very close together. When plotting the conditional values, their existence is clearer. l_0 is highly concentrated on values around 5 and l_1 is highly concentrated around 19.

Figure 5.18 – MDslog empirical posterior distributions

(a) Number of jumps

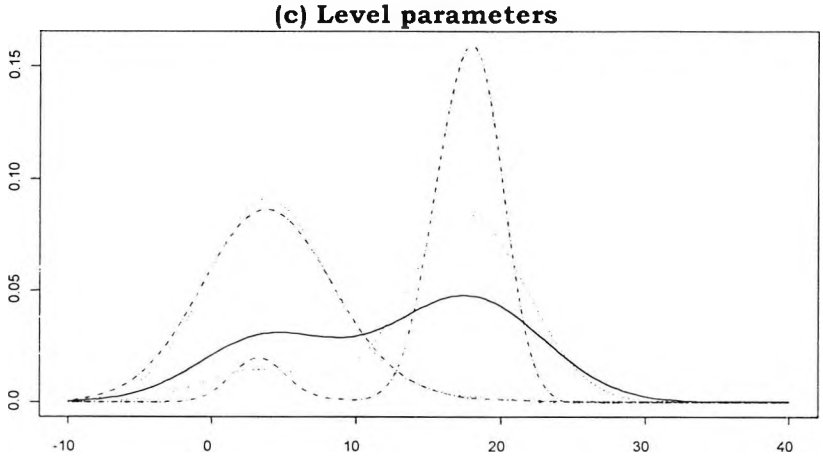


(b) Position of jumps



-·- s_i conditional jump=1; ··· s_i individually, — overall for all s_i i=1,...,6.

(c) Level parameters



-·- l_0, l_1 conditional jump=1; ··· l_0, l_1 individually, — overall for all l_i i= 0,...,6.

5.4 Risk premium

After performing the analysis for all eight models, and summarising their empirical posterior distributions (tables 5.5 and 5.6) the risk premium is calculated. By the analysis, considering the mixing and convergence properties, the models BIf, BIs, MDf and MDs were chosen. In this way all the values were calculated using these models, with the other four only serving to help confirm the results.

The first step is to define the groups for the risk premiums. The way chosen to perform this calculation is by ordering the modes of the position of the jumps for the chosen four models. In this way the following groups (table 5.7) are defined, with the groups also shown in the original scale:

Table 5.7 – Age groups for the risk premium

groups	1	2	3	4	5	6	7
Transf.	(0, 12.59)	(12.59,13.59)	(13.59,32.72)	(32.72,38.26)	(38.26,40.27)	(40.27,62.42)	(62.42,73.59)
Real	(19.39,31.98)	(31.98,32.98)	(32.98,52.11)	(52.11,57.65)	(57.65,59.66)	(59.66,81.81)	(81.81,92.98)

Thus, 7 groups were defined for the risk premium. Only one of the groups is fully dependent on a group derived from severity, while all the others are derived from frequency. These groups t' ($t'=1,\ldots,7$) define the risk premium following the formula 5.1 that is rewritten as:

$$\text{Risk premium}(t') = MD_{freq}(t')\times MD_{sev}(t') + BI_{freq}(t')\times BI_{sev}(t') \qquad (5.3)$$

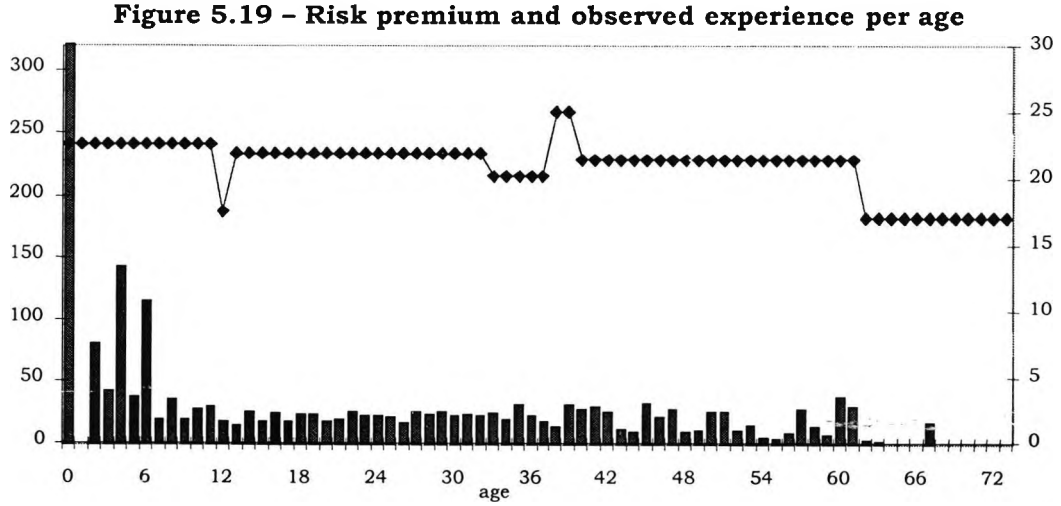
So, the risk premium is calculated by the formula 5.3 observing the change in values in each component depending on the group. The final risk premium is given in table 5.8 (observe that BI_{sev} had to be multiplied by 10 in order to be in 1,000's):

Table 5.8 – Risk premium per group in the automobile insurance problem

Group	Premium in 1,000s	Bodily injury		Motor damage	
		Frequency	Severity	Frequency	Severity
1	22.6281	0.01872	80.577	0.26668	28.293
2	17.5358	0.01240	80.577	0.26668	28.293
3	21.8737	0.01240	80.577	0.19588	60.665
4	20.1669	0.01028	80.577	0.19588	60.665
5	25.0049	0.01028	127.636	0.19588	60.665
6	21.4300	0.01028	127.636	0.13695	60.665
7	17.0020	0.01028	127.636	0.06396	60.665

In order to compare the chosen risk premium with the data a graph is plotted, with the observed risk premium versus the observed one when considering ages in full years as a discrete variable. The graph is in figure 5.19.

Figure 5.19 – Risk premium and observed experience per age



After deciding the risk premium values, the next question is the variability related to such an estimator. In this case it is difficult to define such a value, since again there is no obvious choice or defined rule. So in

order to perform this calculation the approach is to explore the risk premium formula and independence, calculating Var(Risk premium(t')) as:

$$Var(MD_{freq}(t'))\times Var(MD_{sev}(t')) + Var(BI_{freq}(t'))\times Var(BI_{sev}(t')) \qquad (5.4)$$

For $t'=1,\ldots,7$. $Var(MD_{freq}(t'))$, $Var(MD_{sev}(t'))$, $Var(BI_{freq}(t'))$ and $Var(BI_{sev}(t'))$ are calculated by the square of the sample standard deviation of each level l_j, $j=0,\ldots, k$, for each one of the following models: BIf, BIs, MDf and MDs. The results are shown in table 5.9 (where BI_{sev} had again to be multiplied by 10).

Table 5.9 – Standard deviation for the risk premium

Group	Risk premium	Bodily injury		Motor damage	
		Frequency	Severity	Frequency	Severity
1	0.6042	0.00257	21.528	0.01284	18.777
2	0.4772	0.00191	21.528	0.01284	18.777
3	0.4431	0.00191	21.528	0.01228	13.309
4	0.4731	0.00206	21.528	0.01228	13.309
5	0.8181	0.00206	38.870	0.01228	13.309
6	0.8690	0.00206	38.870	0.02521	13.309
7	0.9107	0.00206	38.870	0.03248	13.309

With all these values to hand, the result for the risk premium is complete. It is important to observe that in the calculation of the standard deviation only the variability related to the level parameters has been taken into consideration. In this way it could be argued that this value is underestimated.

But, for comparison, a second approach for calculating the risk premium is used. Now, instead of calculating $MD_{freq}(t')$, $MD_{sev}(t')$, $BI_{freq}(t')$ and $BI_{sev}(t')$ separately, the same sample is used to calculate the risk premium directly. So, in each simulation step the value of the risk premium is calculated directly from the sampled values for each type of data separately. At the end the risk premium has a chain of 10,000 values, that is used as the basis for the analysis.

In such an approach the choice of group positions and numbers of jumps is more difficult to decide since frequency and severity data have different trends over age (the first decreases, while the second increases with age). So, the choice of groups is dropped and the calculation of the premium is different. Now, some ages are chosen and the risk premium, together with the

standard deviations, are calculated for each of these ages. The results are given in table 5.10:

Table 5.10 – Risk premium based directly on the sample

Age	Mean	SD
0	22.63	6.62
2.45	22.49	5.18
4.91	22.56	4.58
7.36	22.30	3.66
9.81	22.16	3.39
12.27	20.62	3.83
14.72	22.20	1.91
17.17	22.20	1.86

Age	Mean	SD
19.62	22.32	1.81
22.08	22.15	1.70
24.53	22.22	1.69
26.98	22.24	1.74
29.44	22.36	1.81
31.89	22.66	2.21
34.34	23.55	3.15
36.80	22.07	3.87

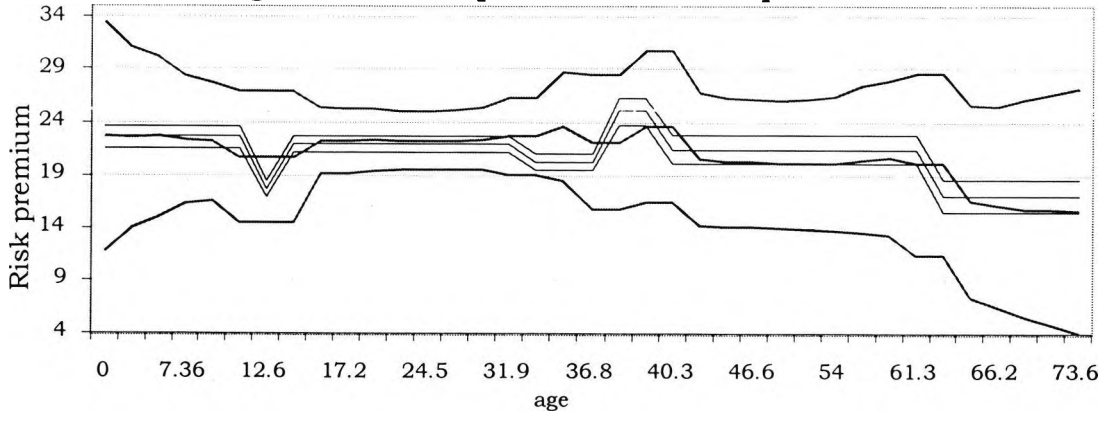
Age	Mean	SD
39.25	23.55	4.41
41.70	20.54	3.83
44.15	20.17	3.71
46.61	20.12	3.69
49.06	19.98	3.69
51.51	19.99	3.78
53.97	20.08	3.93
56.42	20.40	4.24

Age	Mean	SD
58.87	20.63	4.50
61.33	19.98	5.24
63.78	16.51	5.59
66.23	16.01	5.80
68.68	15.83	6.25
71.14	15.74	6.65
73.59	15.61	7.04

These results do not differ significantly from the previous ones in the mean values, but the standard deviation do differ. Such a difference is justified by the lack of use of groups, which leaves the values related to the levels more spread than when a group is defined. It is also true that the second approach takes into consideration also the deviation included in the position and number of jumps.

In order to compare the results a graph with a 90% confidence interval from the normal distribution is plotted in the figure 5.20 for both approaches, where the thicker line is the risk premium based on directly on the sample and the thinner one is the one supposing independence.

Figure 5.20 – Comparison of both risk premiums



5.5 Comparison with Verrall and Yakoubov(1998)

In the paper by Verrall and Yakoubov(1998), a fuzzy set approach was used to calculate the risk premium for the same data set used in this chapter. In their approach the data were discretized, and age was considered in years. Then the clusters were chosen based only on the frequency data for both

motor damage and bodily injury at the same time, where the severity parameters were derived from the sample mean on each specific cluster.

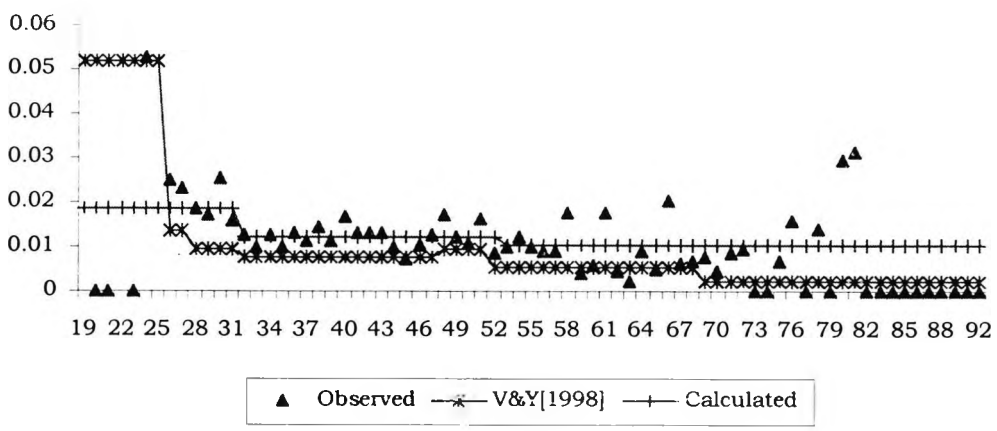
In order to use fuzzy set methodology, the number of clusters must be decided in advance and afterwards the probability of each age being in each group is derived. With this information available, it is then decided in which cluster the age should be included.

The results from Verrall and Yakoubov(1998) are given below in table 5.11. To help the comparison, in figure 5.21 the plot of the observed and estimated frequency is also shown. (Observe that the plots were truncated on younger ages to help presentation).

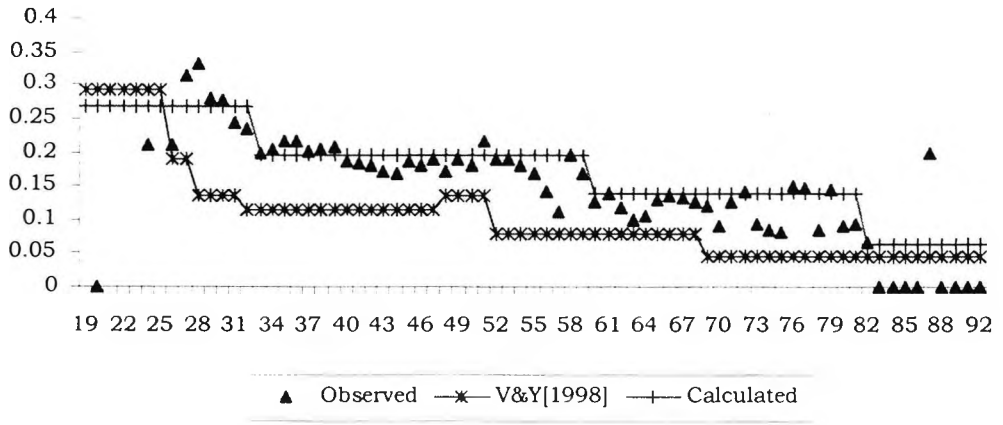
Table 5.11 – Risk premium for Verrall and Yakoubov(1998)

Group	Age	Cluster	BI frequency	MD frequency	Risk Premium
1	(<25,25)	1	0.051603	0.292859	406.55
2	(26,27)	2	0.013503	0.189539	139.18
3	(28,31)	3	0.009412	0.134440	116.77
4	(32,47)	4	0.007597	0.114335	91.97
5	(48,51)	3	0.009412	0.134440	116.77
6	(52,68)	5	0.005616	0.079628	66.65
7	(69,>69)	6	0.002293	0.045360	26.41

Figure 5.21 – Comparison of frequency results with data
(a) Bodily injury claims



(b) Motor damage claims



Comparing their results with those obtained in the previous section, it is possible to observe some differences. In table 5.11, for instance, there is a lower risk class (32,47), which is not a feasible result in the frequency analysis of motor insurance. It is also observed that the number of groups is higher than in the analysis in section 5.4, with only a few ages in one of them.

Apart from this, the range of frequency rate in motor damage is much higher than in bodily injury, but both have the same number of groups in table 5.11. The smoothness of the estimated parameters is also better in the model adopted in this thesis, where the high decrease from groups 1 to 2 in Verrall and Yakoubov(1998) in both motor damage and bodily injury data is not observed. By predefining a group for ages smaller than 26 to apply the model, they had created a higher risk group.

5.6 Chapter conclusion

In this chapter we have shown a Bayesian approach for deciding the number of groups, their positions and rates in each interval, as a unified model. The procedure was applied to severity and frequency of claims separately.

This kind of approach helps the decision making process, since it gives the posterior distribution, instead of a point estimate, and does not define the number of jumps in advance. The same ideas are the basis for the following chapter.

Chapter 6

Investigation of claims reserving

With the model in chapter 5 it was possible to use data at an individual level in order to define groups and summarise information. One of the features in that model was the ability to smooth the parameters over their whole range of values. Utilising both the smoothing property and the data structure, this model is now used in a different type of problem in this chapter: the prediction of claim reserves at an individual policy level.

A micro approach, using individual policy/claim information and continuous time, is used here to model the claim development. The idea is derived from Norberg (1993,1999), the model is similar to Arjas and Haastrup (1996) and the implementation is based on Green(1995). In some ways, the model applied in the following sections is an adaptation of the model of Arjas and Haastrup (1996) using the reversible jump Markov chain Monte Carlo technique.

Besides the change in the implementation technique, the model itself is modified. As before, severity and frequency of claims are modelled separately. The process related to the claims occurrence is assumed to be the same as in Arjas and Haastrup (1996): a Poisson process. But in the severity case the model changed, with the value of claims now being modelled parametrically.

The chapter is divided up in the following way. Section 6.1 gives the theoretical basis of the model. The next section explains the data structure, which is used further in section 6.3 where the distributional assumptions of the model are explained, based on chapters 3 and 4. The description of the full model is given in section 6.4, where the implementation technique is also reviewed. Section 6.5 gives the analysis of the results obtained after running

the implementation. Section 6.6 compares the results proposed in this chapter with those in Arjas and Haastrup(1996) and also to reserve techniques widely used by practical general insurance actuaries. Section 6.7 is the chapter conclusion.

6.1 Claim process description

Reviewing the results from chapter 3, the model is defined as a marked Poisson process. The occurrence of a claim is modelled as a non-homogeneous Poisson process with the corresponding development as position-dependent marks. A claim is a pair (T, Z) where T is the time of occurrence of the claim and Z is the mark describing its development. The model implemented in this chapter has a mark definition based on Arjas and Haastrup (1996) which was expressed in formula 3.20 and is rewritten here:

$$Z = (S, A, U, V, X, \{X(v'); 0 \le v' < V\}) \qquad (6.1)$$

where:

S is the sex of the policyholder;

A is the age of the policyholder;

U is the waiting time from occurrence until notification;

V is the waiting time from notification until final settlement;

X is the final claim amount ($X=X(V)$);

$X(v')$ is the indemnity paid in respect of the claim up to v' after its notification.

The target of the model is to predict the amount and number of claims which are due to be paid or reported after the present time τ. In the method used to model these quantities, not all the information in (6.1) is used. For instance, only the total of payments for each claim, settled or not, is used. In the same way, the waiting time from notification until settlement V is not used directly in the model, but it is utilised as an indicator of whether the claim was fully paid.

At a specific time τ, the mark Z could be classified into *settled*, *reported-not-settled*, *incurred-not-reported* or *covered-not-incurred*. In Norberg (1993) the intensity of occurrence in each these classes is dependent on the exposure for each class. Since the analysis is done at the present time, τ, it is assumed that no information is available for $t > \tau$, meaning that at any time after τ the exposure is equal to zero. In this way the *covered-not-incurred* possibility does not generate any payments or claims occurrence, and only the first three

classifications are used. Each class of claim has the following type of observation (superscript m stands for missing and o for observed):

$$\begin{aligned} &\text{Settled} - (T^o, S^o, A^o, U^o, V^o, X^o) \\ &\text{RBNS} \quad - (T^o, S^o, A^o, U^o, V^m, X^m) \\ &\text{IBNR} \quad - (T^m, S^m, A^m, U^m, V^m, X^m) \end{aligned} \tag{6.2}$$

Data augmentation is used in the model. The model is constructed as if all covariates and occurrence times had been observed for IBNR claims as well. In other words, as T, S and A have been observed for the IBNR claims. Then, when implementing the model using stochastic simulation, the missing data are sampled and used to update the parameters at each iteration.

In order to complete the description of the model, it is necessary to define some quantities. The total number of claims is a random variable that has a Poisson distribution with intensity as given in formula 3.18. This total number of claims n can be split into the three categories as expressed in the following formula:

$$\begin{aligned} &n = n^{settled} + n^{rbns} + n^{ibnr}, \text{ or} \\ &n = n^o + n^m \end{aligned} \tag{6.3}$$

Where $n^o = n^{settled} + n^{rbns}$ and $n^m = n^{ibnr}$. In the same way the total value of the claim reserve is an important quantity and can be expressed as follows:

$$R = R^{rbns} + R^{ibnr}$$

$$\text{with } R^{ibnr} = \sum_{i=1}^{n^m} X_i^m \text{, and} \tag{6.4}$$

$$R^{rbns} = \sum_{i=1}^{n^{rbns}} X_i^m - X(v')_{TOT}^o = X_{TOT}^m - X(v')_{TOT}^o \qquad v' < V$$

$X_{TOT}^m, X(v')_{TOT}^o$ are related only to the RBNS claims, where the first is the sum of the total value of the RBNS claims and the second is the sum of the observed part of these claims. The way chosen to calculate the R^{rbns} is by calculating X_{TOT}^m and then subtracting the observed part.

In the end, these are three main quantities that are predicted by the model: n^{ibnr}, R^{ibnr} and R^{rbns}. Since we will also obtain a large sample, the predictive empirical distribution for each one of these can be found. The graphs and analysis are presented in subsection 6.5.1.

Again, claim frequency and severity are modelled separately. As in Arjas and Haastrup (1996) the model for the occurrence of claims takes into account the age, sex, reporting delay and calendar time of occurrence. However, the

part of the model related to severity changes completely. Here it is not considered the partial payments are not considered, but rather the total value. As an improvement from the previous approach, sex and age are now included in the model and a parametric distribution is used.

Before describing the full model, with the distributional assumptions and implementation technique, the data is described in the next section.

6.2 Data

The data are for dental insurance, covering only the leisure accident claims. Claims which occurred between 1st January 1982 and 31st December 1990 and which were reported before 3rd March 1992 are in the data set. For comparison, the analysis in Arjas and Haastrup (1996) was followed and the present time is considered to be 31st of December 1987. So, only information reported in years 1982 to 1987, inclusive, was considered, with the rest used to check the model results.

The data set has two covariates: sex and age of the policyholder. In contrast to the application in chapter 5, age now is a discrete variable, since it is only recorded in full years. Ages were transposed by subtracting 16, and the final age range is (0,1,...,66). For simplicity, ages bigger than 66 were not considered since there was no information on exposure in these ages for some years. A factor is defined, taking the value one for females and zero for males.

The original data set has three basic files. The first file contains the exposure of the portfolio, which is given by the total number of policies in force at the end of the year split by sex and age. In order to use the continuous approach an interpolation for the years 1983 to 1987 is used, as is further explained in section 6.3.

The second file contains all the observed partial payments, including the date of payment and the original claim code. The way chosen to model the severity in this chapter (see subsection 5.8) does not require the partial payments, and they are aggregated for each claim. By checking with the third file, each claim is categorised as settled or not settled. The settled claims with total payment less than or equal to zero are discarded from the file.

The third file is related to the individual claim occurrence information, and shows the dates of occurrence, reporting and settlement (if settled), together with the covariates sex and age. In some cases the day of the occurrence of a claim is not know and in these cases a value was sampled

from a uniform distribution on the number of the days of the month, bounded above by the minimum of the final of the month and the reporting date. Again, the settled claims with no payment or negative total payment are not considered (there were 9 claims with this characteristic).

These three files were transformed into six files to be used by the program that implements the model. One gives the information on exposure by sex and age for the years 1982 to 1987. Two files give the extra information required by the severity model: the number of RBNS claims and the logarithm of the total payment for the settled claims, both by sex and age. The rest of the files are related to the frequency model and give the total number of claims by sex and age, by day of occurrence and observed delay.

Time was also rewritten as explained in table 6.1, where in the calendar time intensity the upper bound is given by $L = \tau = 2{,}190$.

Table 6.1 – Calendar time conversion

Calendar	*t*
1st January 1982	0
31st December 1982	364
31st December 1983	729
31st December 1984	1095
31st December 1985	1460
31st December 1986	1825
31st December 1987	2,190

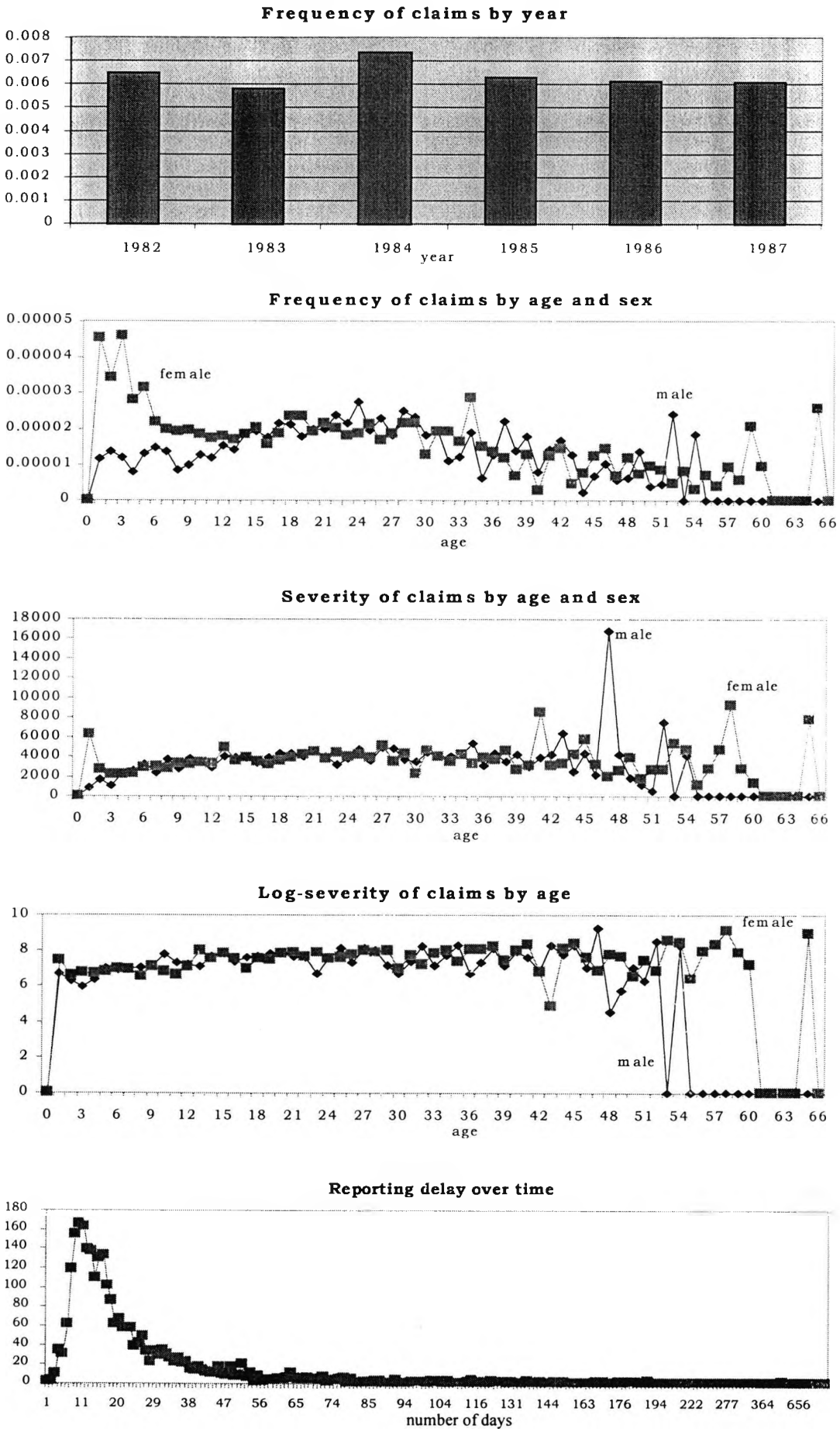
In order to illustrate the nature of the data, the summary plots are shown in figure 6.1. Some summaries are also given in table 6.2, where the total value of claims is 10,042,628, from 2,797 claims. In this amount there are 617 unsettled claims with partial payments amounting to 824,146 and 9 claims that were settled without any payment were not included at all. The average value of the logarithm of a settled claim is 7.98 and its standard deviation is 1.02.

Table 6.2 – Statistics of the dental insurance data

	Calendar daily frequency	Female daily frequency	Male daily frequency	Reporting delay in days	Female severity	Male severity
Mean	1.78E-05	1.19E-05	1.53E-05	31.45	4,186	4,243
St deviation	1.42E-06	8.170E-06	9.76E-06	62.05	1,226	1,544

These statistics, calculated over the data set used but over the settled claims in the severity case, help the choice of the hyper-parameters for the prior distributions. In the next section the full model is explained.

Figure 6.1 – Summary plots of dental insurance accident data



6.1 Distributional assumptions

In this section the distributions involved in the model are fully explained and defined. The basis of the model is explained in Norberg (1993), with the implementation following chapter 4 of this thesis. The construction of the likelihood is as in Arjas and Haastrup (1996), where a piecewise function is applied to all parameters in the model. Differently from their work, RJMCMC is used to find the estimates of the parameters.

The occurrence of a claim is a function of the calendar time (t), sex (s) and age (a) of the policyholder, and it follows a Poisson process with intensity given by $\varphi(t,s,a)$. By considering the calendar time to be independent of the covariates sex and age, the intensity is rewritten as follows:

$$\varphi(t,s,a) = \varphi_1(t)\ \varphi_2(s,a); \qquad (6.5)$$

$t \in [0, L]$, $s \in (0, 1)$ and $a \in (0,1,\ldots,66)$

The weight is given by $w(t,s,a)$, which is defined as the number of policies in force in each combination of t, s, a. Since the database only supplies the total policyholders at the end of the year for each combination of s, a an approximation had to be used. For t in the first year, 1982, the weight $w(t,s,a)$ is considered fixed for each combination of s, a. In the following years the number of policies is force was exponential interpolation over the year before and the actual year. In summary, the total number of claims from the portfolio is governed by the following intensity:

$$w(t,s,a)\ \varphi_1(t)\ \varphi_2(s,a). \qquad (6.6)$$

Males and females are considered independent and the intensities $\varphi_1(t)$, $\varphi_2(0,a)$ and $\varphi_2(1,a)$ are considered piecewise constant random variables. Now, it is necessary to define the model for the marks Z. The delay (u) is modelled via a piecewise intensity $\gamma(u)$, with the accumulated probability density given by:

$$P_U(x) = 1 - exp(-\int_0^x \gamma(t)dt). \qquad (6.7)$$

Finally the severity model uses a parametric approach, depending only on the sex and age of the policyholder. The logarithm of the total payment of a claim follows a normal distribution expressed by:

$$\text{Normal}(\mu(s, a), \sigma^2) \qquad (6.8)$$

where, again, the sexes are considered independent. Following the approach in chapter 4, the variance is considered known and equal for all combinations of sex and age. To sum up there are six unknown parameters expressed by:

$\varphi_1(t)$, $\varphi_2(0,a)$, $\varphi_2(1,a)$, $\gamma(u)$, $\mu(0,a)$ and $\mu(1,a)$. (6.9)

All these parameters are modelled as explained in chapter 4, where $\varphi_2(0,a)$, $\varphi_2(1,a)$, $\mu(0,a)$ and $\mu(1,a)$ are dependent on the covariate age which is considered discrete in this case. With values for these parameters, the missing data in the IBNR and RBNS claims can be simulated.

Using (3.19) the missing number of claims is distributed as a Poisson distribution expressed by:

$$n^{ibnr} \sim \text{Poisson}(W^{ibnr}) \tag{6.10}$$

where $W^{ibnr} = \int_0^{\tau} w(t,s,a)\varphi_1(t)\varphi_2(s,a)(1-P_U(\tau-t))dt$.

The times of occurrence of these claims T^m, together with the covariates S^m and A^m are iid with density

$$w(t,s,a)\varphi_1(t)\varphi_2(s,a)(1-P_U(\tau-t)) \,/\, W^{ibnr} \tag{6.11}$$

Now the results from Arjas and Haastrup (1996) are rewritten in order to show the likelihood. Apart from the severity, all the components are similar. The observed data are the reported claims, that can either be settled or RBNS. So the class of mark Z that is observed are those with reporting date earlier than the present time τ. In this way the process in (3.19) is fully defined by the following formulae for the reported claims case:

$$w^o(t,s,a)\,\varphi_1^o(t)\varphi_2^o(s,a) = w(t,s,a)\,\varphi_1(t)\,\varphi_2(s,a)P_U(\tau\text{-}t)$$

$$P^o{}_U(t+u<\tau) = \frac{P_U(T^o+U^o<\tau)}{P_U(\tau-t)} \tag{6.12}$$

In the same way the missing part, which are the IBNR claims, will have an occurrence intensity expressed by $w(t,s,a)\varphi_1(t)\varphi_2(s,a)(1 - P_U(\tau\text{-}t))$. Gathering together the missing and observed information, the total likelihood expression (following formula (4.15) page 153 in Arjas and Haastrup (1996)) is given as:

$$\prod_{i=1}^{n}\varphi_1(T_i)\varphi_2(S_i,A_i)\exp\{-\sum_{s,a}\int_0^t w(t,s,a)\varphi_1(t)\varphi_2(s,a)dt\}\times$$

$$\prod_{i=1}^{n^o}\gamma(U_i^o)\exp\{-\sum_{i=1}^{n^o}\int_0^{U_i^o}\gamma(u)du-\sum_{j=1}^{n^{ibnr}}\int_0^{-T_j^m}\gamma(u)du\}\times$$

$$\prod_{i=1}^{n^o}\frac{1}{\sqrt{2\sigma^2\pi}}\exp\{-\frac{(X_i^o-\mu(S_i^o,A_i^o))^2}{2\sigma^2}\} \tag{6.13}$$

The full definition of the model is completed by the prior distributions, and then the calculations of the complete conditional distributions can be performed. Each of the six parameters is modelled as piecewise constant

intensities. In this way, for each of them there is a set (k,l_0, l_1, l_2,..., l_k,s_1, s_2,..., s_k) defining the position, level and number of jumps.

In order to use the approach of chapter 4, the ranges t, a and u have to be restricted. Naturally $t \in [0, 2{,}190)$ and $a \in (0,1,...,66)$. In the delay case there is no obvious choice and since the highest observed delay was 1,175, an upper bound of 2,190 was chosen. From the results this limit appeared to be reasonable.

The piecewise intensities related to $\varphi_1(t)$, $\varphi_2(0,a)$, $\varphi_2(1,a)$ and $\gamma(u)$ have as prior the Gamma distribution, with hyper-parameter values fully defined in the next section. The prior distribution of $\mu(0, a)$ and $\mu(1, a)$ are normally distributed. All jump positions have as prior distributions the same distribution as in chapter 4, which is the even ordered uniform statistics on $[0,L)$, where L is 2,190 for $\varphi_1(t)$ and $\gamma(u)$ and 67 for $\varphi_2(0,a)$, $\varphi_2(1,a)$, $\mu(0, a)$ and $\mu(1, a)$.

Again, the prior distribution for k is defined to be a Poisson distribution with rate λ expressed by the following formula:

$$p(k) = \frac{\exp^{-\lambda}\lambda^k}{k!} \qquad k \le k_{max} \qquad (6.14)$$

where the hyper-parameter k_{max} is included in order to guarantee that only a finite numbers of models are considered. Having fully defined the model, its implementation can be explained and the description of the complete conditional distributions and implementation technique is given in the next section.

6.4 Description of the model

Since all parameters $\varphi_1(t)$, $\varphi_2(0,a)$, $\varphi_2(1,a)$, $\gamma(u)$, $\mu(0, a)$ and $\mu(1, a)$ are modelled as piecewise constant functions, the implementation is similar to that used in chapter 4. It is necessary to define the complete conditional distributions for each set $\{k, \theta^{(k)} = (l_0, l_1, l_2,..., l_k, s_1, s_2,..., s_k)\}$ related to each one of the six parameters $\theta_{(t)}^{(k(t))}$, $\theta_{(0,a)}^{(k(0,a))}$, $\theta_{(1,a)}^{(k(1,a))}$, $\theta_{(u)}^{(k(u))}$, $\theta_{\mu(0,a)}^{(k(0,a))}$ and $\theta_{\mu(1,a)}^{(k(1,a))}$, with the usual notation for k. In order to simplify notation, the subscript is not included when writing the distributions. All distributions are based in subsection 4.2.

The number of jumps, k, follows exactly the same approach as in subsection 4.2.2. The function $f(l_j)$ is as described in (4.19) for $\mu(0,a)$ and

$\mu(1,a)$, and as in (4.20) for the other parameters and the function $g(.)$ always assumes the format in (4.21), which is $g(s_j, s_{j+1}) = s_{j+1} - s_j$.

Subsection 4.2.1 explains the distributions related to the level and position of jumps. The parameters related to the severity problem, $\mu(0,a)$ and $\mu(1,a)$, follow exactly the procedures described in section 4.2.1 for the severity case, with the necessary adaptations for the discrete case (since age is discrete here). L is equal to 67.

The calendar occurrence time $\varphi_1(t)$ has the following complete conditional distributions for $\theta_{(t)}^{(k)}$:

$$\pi(l_j|.) = l_j^{x_j^{j+1}} \exp\{-l_j \sum_{s,a} \int_{s_j}^{s_{j+1}} w(t,s,a)\varphi_2(s,a)dt\}\, p(l_j)$$

$$\pi(s_j|.) = l_{j-1}^{x_{j-1}^{j}} l_j^{x_j^{j+1}} \times \qquad (6.15)$$

$$\exp\{-l_{j-1} \sum_{s,a} \int_{s_{j-1}}^{s_j} w(t,s,a)\varphi_2(s,a)dt - l_j \sum_{s,a} \int_{s_j}^{s_{j+1}} w(t,s,a)\varphi_2(s,a)dt\}\, p(s_j)$$

where x_j^{j+1} is the total number of observed or sampled claims in $[s_j, s_{j+1})$.

As explained in subsection 6.2, the weight $w(t,s,a)$ was assumed to be constant for t in the year 1982, but had to be exponentially interpolated for t in years 1983 to 1987. Since weight is an exponential function of t the expression $\pi(s_j|.)$ did not have an analytical solution in the calendar time intensity case. So, an approximation was used, based on the idea in Arjas and Gasbarra (1994) which was explained in subsection 4.2.1. The approach is that of dividing the interval (s_j, s_{j+1}) into intervals with the same weight:

$w(t',s,a)$ = *constant* for $t' \in [integer(t'), integer(t')+1)$

Because of this split, the update for the position of a jump in calendar occurrence time, $\varphi_1(t)$, is the most time demanding among all parameters. Now the (sex, age) intensity of occurrence of claims $\varphi_2(0,a)$ and $\varphi_2(1,a)$ have similar complete conditional distributions (expressed here for male $\theta_{(0,\alpha)}^{(k)}$, but being of the same form for female $\theta_{(1,\alpha)}^{(k)}$):

$$\pi(l_j|.) = l_j^{x_j^{j+1}} \exp\{-l_j \sum_{s_j \le s < s_{j+1}} \int_0^{2190} w(t,s,0)\varphi_1(t)dt\}\; p(l_j)$$

$$\pi(s_j|.) = l_{j-1}^{x_{j-1}^{j}} l_j^{x_j^{j+1}} \times p(s_j) \qquad (6.16)$$

$$\exp\{-l_{j-1} \sum_{s_{j-1} \le s < s_j} \int_0^{2190} w(t,s,0)\varphi_1(t)dt - l_j \sum_{s_j \le s < s_{j+1}} \int_0^{2190} w(t,s,0)\varphi_1(t)dt\}$$

Finally, the delay intensity $\gamma(u)$ has the following complete conditional distributions for $\theta_{(u)}^{(k)}$:

$$\pi(l_j|.) = l_j^{x_j^{j+1}} \exp\{-l_j W_u(s_j, s_{j+1})\}\ p(l_j) \qquad (6.17)$$

$$\pi(s_j|.) = l_{j-1}^{x_{j-1}^{j}} l_j^{x_j^{j+1}} \exp\{-l_{j-1} W_u(s_{j-1}, s_j) - l_j W_u(s_j, s_{j+1})\}\, p(s_j)$$

where $W_u(s_{j-1}, s_j) = \sum_{i=1}^{n^o} (d(u_i) - s_{j-1})$, and

$$d(u_i) \begin{cases} s_{j-1}, & u_i \le s_{j-1} \\ s_j, & u_i \ge s_j \\ u_i, & s_{j-1} < u_i < s_j \end{cases}$$

With all complete conditional distributions, the final algorithm is given by the scheme in figure 6.2. The prior parameter values and model constants necessary to implement the model are given in the next section.

Figure 6.2 – Algorithm for the dental insurance data

Initialise the vector of parameters for all six cases.
for j=1 to N^*
- Sample W^{ibnr}
- Sample n^{ibnr}
- Sample T_i^m, S_i^m, A_i^m for i=1,..., n^{ibnr}
- Calculate R^{ibnr} and R^{rbns}
- Update $\theta_{(t)}^{(k)}$ and $k_{(t)}$ using the scheme in figure 4.2
- Update $\theta_{(0,\alpha)}^{(k)}$ and $k_{(0,\alpha)}$ using the scheme in figure 4.2
- Update$\theta_{(1,\alpha)}^{(k)}$ and $k_{(1,\alpha)}$ using the scheme in figure 4.2
- Update $\theta_{(u)}^{(k)}$ and $k_{(u)}$ using the scheme in figure 4.2
- Update $\theta_{\mu(0,\alpha)}^{(k)}$ and $k_{\mu(0,\alpha)}$ using the scheme in figure 4.2
- Update $\theta_{\mu(1,\alpha)}^{(k)}$ and $k_{\mu(1,\alpha)}$ using the scheme in figure 4.2

Next j

6.5 Implementation of the model

In this section, the full definition of the prior distributions is given. The prior Poisson rate for the number of jumps k was chosen as λ=3, and,

controlling the number of jumps, k_{max} was chosen to be 30. This upper bound was not reached during the simulations of any of the parameters. The prior distribution for the position and level parameters for each of the parameters was chosen as expressed in the table 6.3:

Table 6.3 – Values of the hyper-parameters on the prior distributions

	Position	**Level**
$\varphi_1(t)$	$U^2(0,2,190)$	Gamma (2,0.2527)
$\varphi_2(0,a)$	$U^2\{0,1,...,66\}$	Gamma (2,126600)
$\varphi_2(1,a)$	$U^2\{0,1,...,66\}$	Gamma (2,126600)
$\gamma(u)$	$U^2(0,2,190)$	Gamma (2,421)
$\mu(0,a)$	$U^2\{0,1,...,66\}$	Normal (8,1.02)
$\mu(1,a)$	$U^2\{0,1,...,66\}$	Normal (8,1.02)

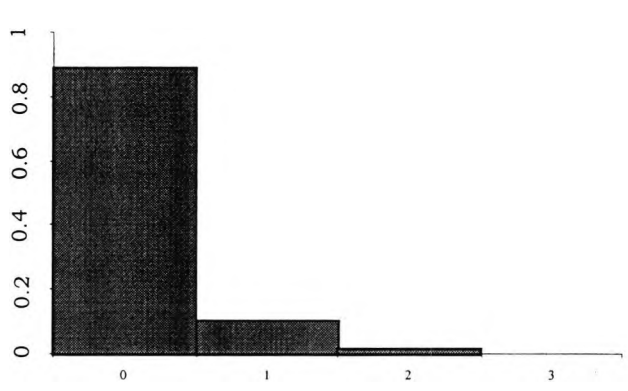
Where again U^2 is the even ordered uniform distribution over the respective interval. For the severity case, the standard deviation was fixed as σ=1.25 for both cases. This value was fixed after examining the data. The initial points were also defined, following the same approach as in chapter 5.

When implementing the model it was observed that the multiplicative structure of the intensity of occurrence $\varphi_1(t)\ \varphi_2(s,a)$ was a problem when using a piecewise constant structure without the number of groups being pre-defined. Allowing the groups to change created a problem of convergence, since the product converges, but not each intensity separately.

In order to solve this problem the following option was proposed. First the intensity $\varphi_1(t)$ would be updated, with $\varphi_2(s,a)$ fixed. Then, after 5,000 updates $\varphi_1(t)$ would be fixed at its posterior mean and $\varphi_2(s,a)$ updated. After 5,000 updates the same procedure would be used and $\varphi_1(t)$ would be updated while $\varphi_2(s,a)$ was kept fixed. The choice for the number of updates as 5,000 was completely arbitrary, but aimed to be large enough to achieve convergence, but not that large given the time constraint.

It was a quite demanding process since each run j in figure 6.2 would take 3 minutes to be completed. After some replication of this procedure it was observed that $\varphi_1(t)$ was not indicating any group, meaning that this intensity was constant over the calendar time. The histogram of the sample of number of jumps is shown in figure 6.3:

Figure 6.3 – $\varphi_1(t)$ histogram for the number of jumps



Reaffirming the result obtained, the analysis in Arjas and Haastrup (1996) did not shown any clear influence of the calendar time intensity. So it was chosen to keep this intensity fixed at the value one and perform the model with the intensity of occurrence dependent only on sex and age. Thus:

$\varphi_1(t)=1$, $t \in [0,L)$, zero otherwise.

The final values for $\varphi_2(0,a)$, $\varphi_2(1,a)$, γ (u), μ $(0, a)$ and μ $(1, a)$ in the update with calendar time intensity varying were used as initial values for the new modelling with $\varphi_1(t)$ constant. The simulation was updated for 15,000 steps on a Sparc ultra 1 140MHz with 64Mb memory. For each model, one step, *j*, in the algorithm in figure 4.8 took around three minutes, which is somewhat larger when compared to the previous application in chapter 5.

The acceptance rate for the birth and death moves ((b) and (c) in subsection 4.2.2) are shown in table 6.4.

Table 6.4 – Type of moves and acceptance levels

	Kept Move(a)	Birth Move(b)	Accepted	Rate	Death move(c)	Accepted	Rate
$\varphi_2(0,a)$	3,855	6,633	53	0.80%	4,512	53	1.17%
$\varphi_2(1,a)$	3,538	5,896	196	3.32%	5,566	195	3.50%
$\gamma(u)$	6,220	2,084	355	17.03%	6,696	354	5.29%
$\mu(0, a)$	10,779	5,930	253	4.27%	1,709	256	14.98%
$\mu(1, a)$	10,538	5,974	312	5.22%	1,512	317	20.97%

The rate of acceptance was not as high as would be expected and, in fact, it was very low in some cases. This is not a good result, but can be explained by one interesting feature in this sample. In most cases, including the ones with worst acceptance performance, the absolute number of accepted runs for both (b) and (c) moves are very close together. This could signify that convergence had already been achieved in the initial values derived from the

previous model with $\varphi_1(t)$ varying, and in this way that no change in the number of jumps in each parameter was actually necessary.

After obtaining the sample, we may perform the analysis. From the 15,000 samples, the initial 5,000 were considered as burn-in iterations (M=5,000) and discarded. Convergence was assessed by the visual evidence in the trace plots.

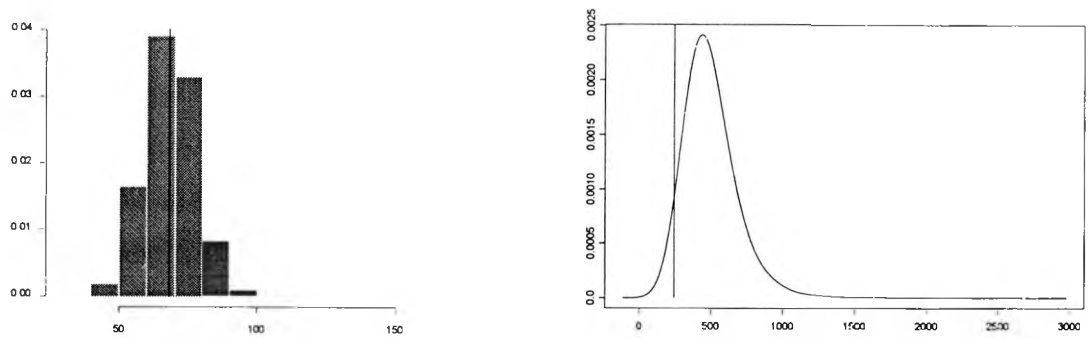
In the next section, the analysis of the sample is performed. It should be remembered that the main interest is the smoothness property of the model rather than the definition of the groups for each parameter. This is the reason why the analysis performed in the next subsections is different from that in chapter 5.

6.5.1 IBNR and RBNS analysis

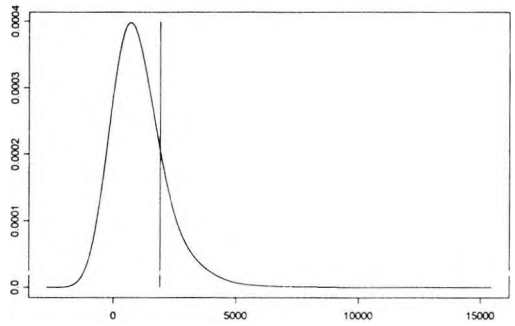
As stated previously before, the main interest of this analysis is the outstanding quantities for claims which occurred in $t \in (0,2,190)$. They are denoted by n^{ibnr}, R^{ibnr} and R^{rbns}, which are, respectively, the number of IBNR claims, the reserve for IBNR claims and reserve for RBNS claims. Their empirical predictive distributions are in figure 6.4 and they are estimated from the obtained sample, excluding the burn-in runs. For comparison the observed values derived from the complete data base are also included in each graph as a line.

Figure 6.4 – Outstanding quantities empirical predictive distributions

(a) Number of IBNR **(b) IBNR reserve**



(c) RBNS reserve



From this sample, some summaries are also presented in table 6.5, together with the observed values. All the currency values are shown in thousand.

Table 6.5 – Summaries of the sample for IBNR and RBNS claims

	n^{ibnr}	R^{ibnr}	R^{rbns}
Mean	68.83	499.09	1,066.78
St. deviation	9.42	180.62	1,099.22
90% confidence interval	(54,84)	(284.95,816.11)	(-247.83,3,064.40)
Actual observed values in $t \in$ (2,190,3,714)	68	246.40	1,910.64

These values are the amounts at the present time, defining the total liability with reference to the period $t \in (0,2,190)$. The comparison with the values in the data set is only an indicator of the development, since all the claims that were not notified or fully paid before 3/3/1992, or t=3,714, are not considered. $\pi_u(u > 3,714 - 2,190)$ is very low and there should be very few IBNR claims in t > 3,714 related to the analysed period. In this way it is supposed that by t=3,714 all IBNR claims for the period analysed would have been reported.

The empirical posterior distribution seems to model particularly well the number of IBNR claims, seeming a Poisson shaped distribution. This result is based on the frequency part of the model. The posterior results for the reserve amounts of IBNR and RBNS do not seem to predict the data so well. The observed amount of IBNR is a low quantile of the posterior distribution and the RBNS are in a high quantile.

This behaviour for the predicted reserve amount can be explained by the unusual claim value average observed for IBNR and RBNS: 3,623 and 7,519, compared to the 4,229 observed for settled claims in t <2,190. The

longest settlement delay observed in the data set is 3,494 days. In this way, while there are still some payments expected to occur for IBNR claims, that will increase the average (10 claims have not been settled by t=3,714), the RBNS value is quite high (with still 182 claims to be settled by t=3,714). The fact that the delay of payment is not taken into consideration on our analysis did not improve our model, since the data shows a positive correlation between total value and settlement delay.

Our model also proved to be quite sensitive to the choice of the variance σ^2. The chosen value was σ =1.25 and to show its sensitivity some scenarios were run for lower (1.02) and higher (2) values. The results are as follow in table 6.6:

Table 6.6 – Different scenarios for IBNR and RBNS claims

	σ	R^{ibnr}	R^{rbns}
Mean	1.02	384	149
	2	1,720	10,729
St. deviation	1.02	104	601
	2	1,974	15,626
90% confidence interval	1.02	(246, 570)	(-634, 1,236)
	2	(544, 4,135)	(2,364, 27,422)

These results show that not much improvement can be obtained by changing the value of σ, but that the posterior distribution is quite sensitive to its choice.

In the next three subsections the simulation output for each of the parameters is analysed. Traces and empirical posterior plots are shown.

6.5.2 Age, sex intensity of occurrence

The age effect is analysed for both the male and female cases. Good convergence is observed for both parameters, apart from the low acceptance level for the birth and death moves.

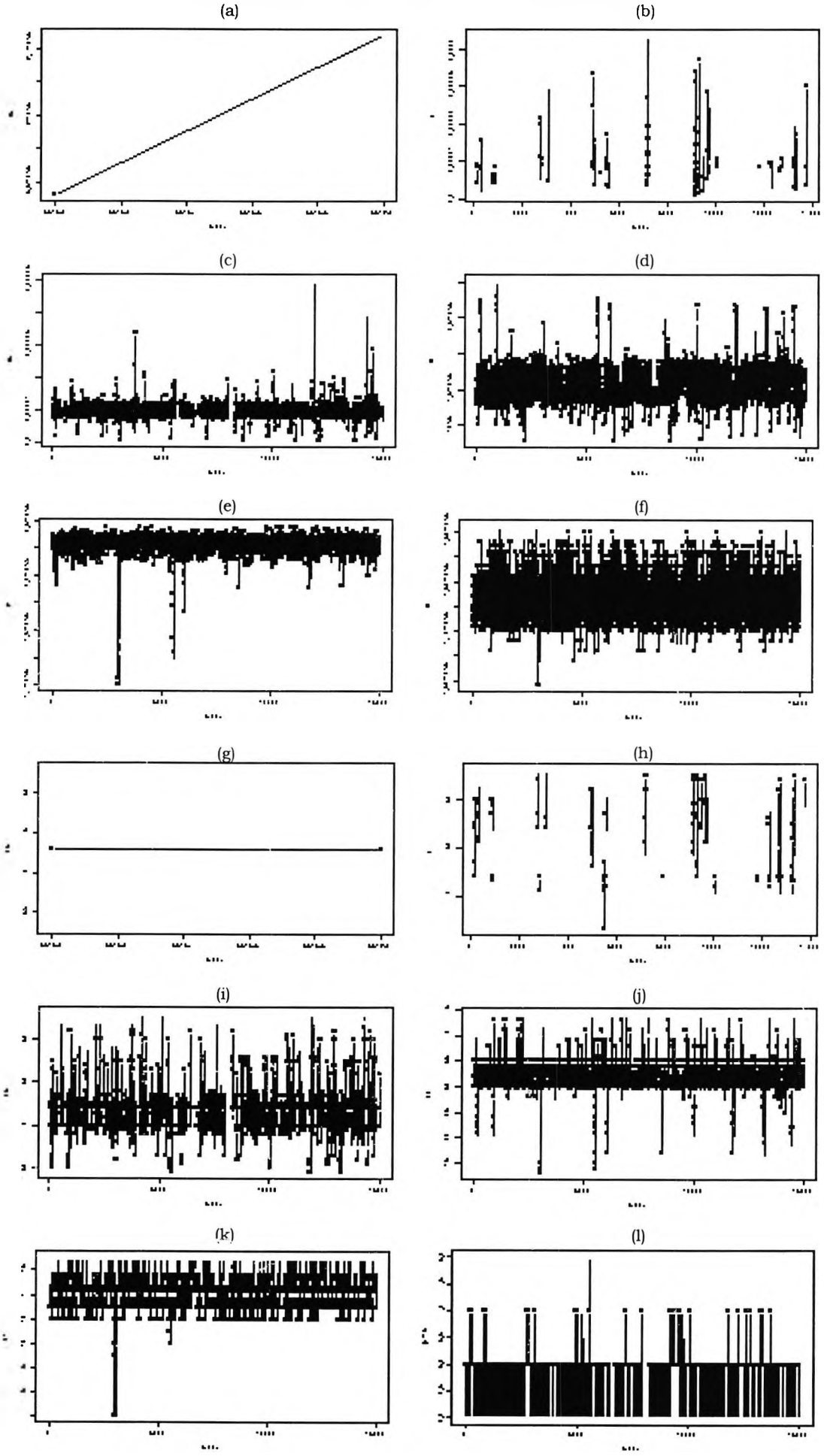
Female intensity:

Convergence (traces in figure 6.5):

Number of jumps (plot (l)) – Not a good mixing, varying between 2 and 3 jumps.

Position of jumps (plots (g) to (k)) – s_4 and s_5 are not frequent enough to draw any conclusion. s_3, s_2, and s_1 converge to values around 40, 30 and 14.

Figure 6.5 – Female occurrence trace plots



Level parameters (plots (a) to (f)) – l_4 and l_5 are not frequent enough to draw any conclusion. l_3, l_2, l_1 and l_0 converge to values around $1*10^{-4}$, $1.5*10^{-5}$, $2.4*10^{-5}$ and $1.5*10^{-5}$.

Analysis of empirical posterior distributions (Figure 6.6):

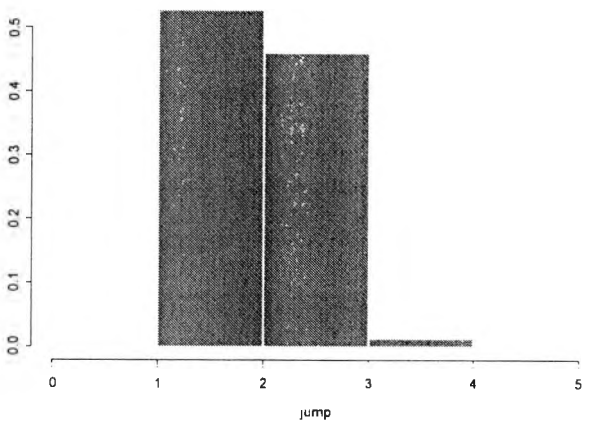
Number of jumps – The values 2 and 3 have the highest probability, with a very small difference between them.

Position of jumps – The graph indicates three well-defined modes (with approximate values 14, 30, 45). The highest value of 45 has the lowest probability.

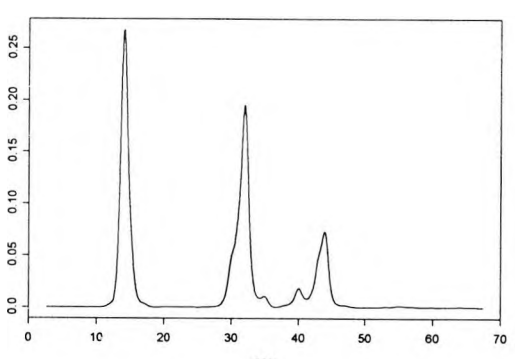
Level parameters – The overall graph indicates three main areas of higher density. The first one, related to l_3, has a value smaller than 0.00001, then l_2 and l_0 are in the same region around 0.000015. l_1 is around 0.000025.

Figure 6.6 – Female intensity empirical posterior distributions

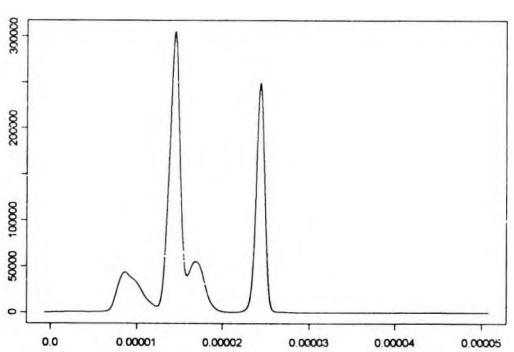
(a) Number of jumps



(b) Position of jumps



(c) level parameters



Male intensity:

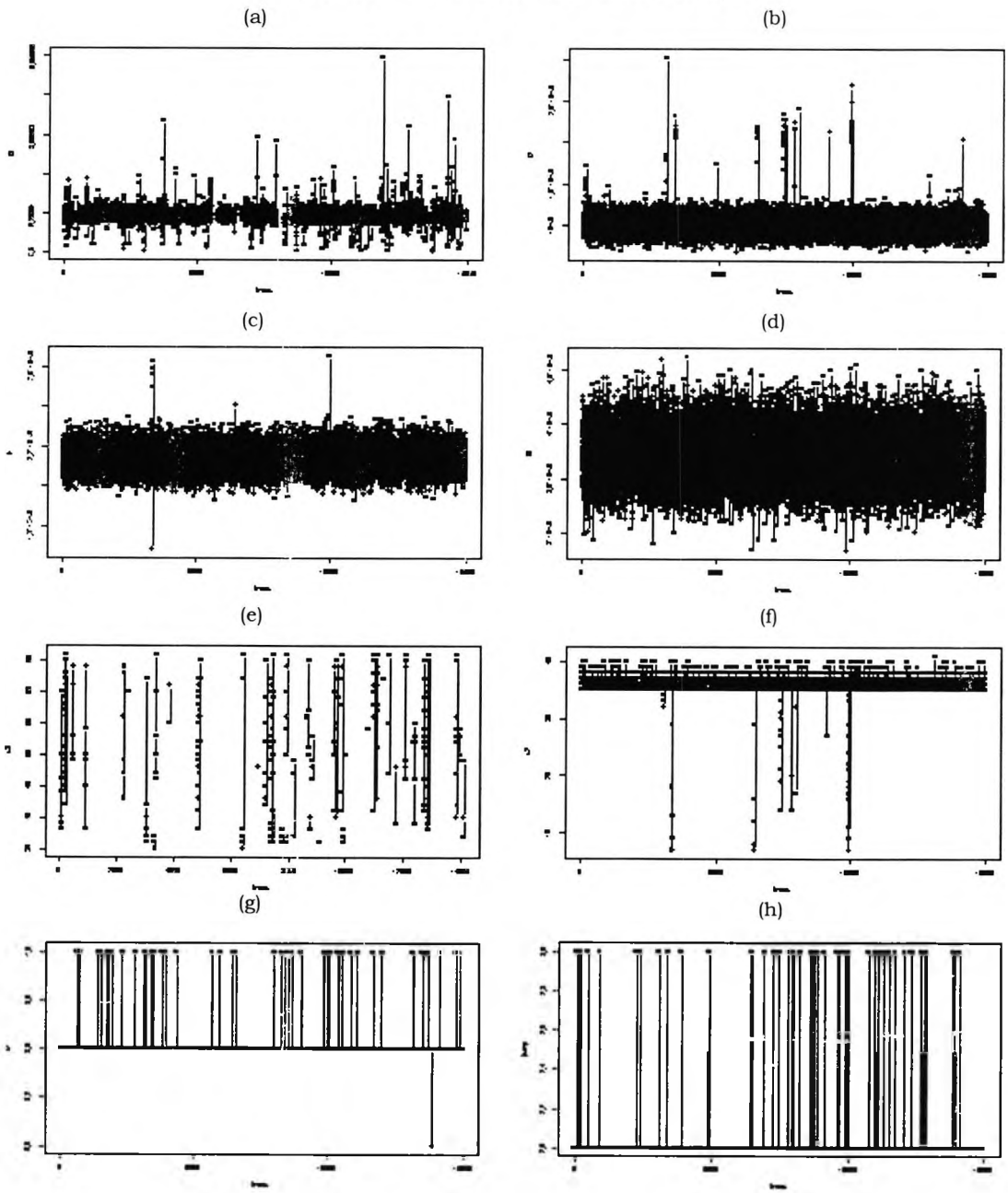
Convergence (traces in figure 6.7):

Number of jumps (plot (h)) – Not a good mixing, varying in between 2 and 3 jumps.

Position of jumps (plots (e) to (g)) – $s_3 > 35$ and $40 > s_2 > 35$. s_1 only assumes values 6 and 7 with the first as the most frequent one.

Level parameters (plots (a) to (d)) – l_3, l_2, l_1 and l_0 converges to values around $1.0*10^{-5}$, $1.0*10^{-5}$, $2.2*10^{-5}$ and $3.5*10^{-5}$. Both l_3 and l_2 go to the same region.

Figure 6.7 – Male occurrence trace plots



Analysis of empirical posterior distributions (Figure 6.8):

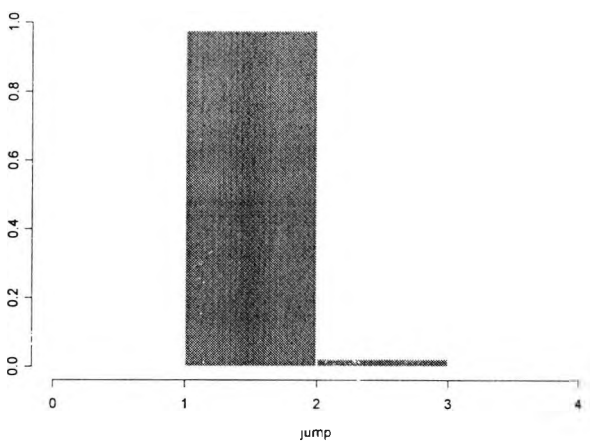
Number of jumps – The value 2 has the highest probability.

Position of jumps – The graph indicates two well-defined modes (approximated values of 6, 38).

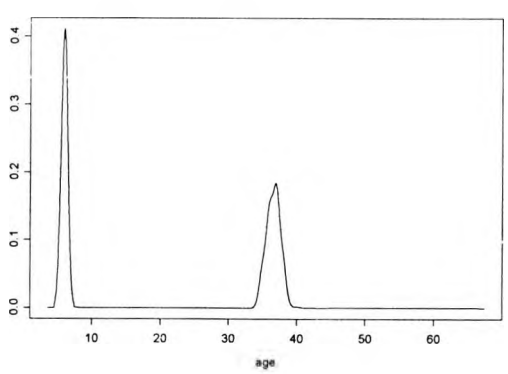
Level parameters – The overall graph indicates three main areas of higher density. The first one, related to l_2, has a range around 0.00001, then l_1 is around 0.00002 and l_0 is around 0.000035.

Figure 6.8 – Male intensity empirical posterior distributions

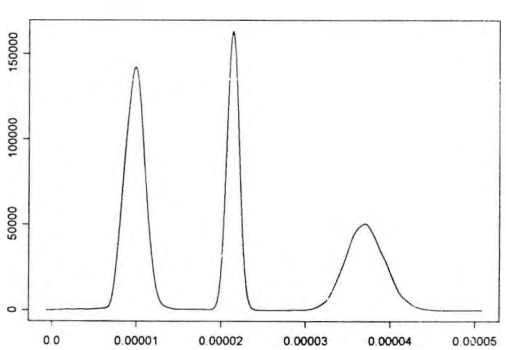
(a) Number of jumps



(b) Position of jumps



(c) level parameters



6.5.3 Intensity of delay

The updates of the intensity of delay had the best acceptance level among the parameters in this model, although the structure of its observed distribution (see figure 6.1) is closer to a polynomial than a piecewise constant function. This result is reaffirmed by the results for the number, position and

level parameters. The number of jumps k has mode around 7, while in the position and level parameters there is only one very high posterior mode.

The number of jumps goes up to a value of 16, but only the traces of the first to the eighth jump is plotted. The ones which are not included in here are either without an obvious convergence, either with very few observations.

Convergence (traces in figure 6.9):

Number of jumps (plot (r)) – A good mixing, but no obvious convergence. This is the highest range of values, which is from 6 to 16.

Position of jumps (plots (j) to (q)) – s_8 and s_7 do not have any convergence. The position of jumps s_6, s_5, s_4 and s_3 have some blocks, with the region with highest probability localised in (200,300), (70,80), (35,40) and 17 respectively. s_2 and s_1 have the highest convergence (values (7,8) and (4,5) respectively).

Level parameters (plots (a) to (i)) – l_8, l_7 and l_6 do no have an obvious convergence. Besides l_0 that converges to values in (0.01, 0.02) all other intensities l_5, l_4, l_3, l_2 and l_1 present some blocks. The most probable values for these intensities are respectively (<0.01), 0.01, 0.03, 0.05 and 0.1 showing a tendency of decreasing the value with the order of the level. In a way it proves that the level parameters decreases with u.

Analysis of empirical posterior distributions (Figure 6.10):

Number of jumps – The values around 9 are the most probable, but there is no obvious choice for the posterior number of jumps.

Position of jumps – The graph only shows two obvious choices for the position of jumps: values around 7 and then around 300.

Level parameters – The empirical posterior density shows a decreasing behaviour over u with only one clear mode in values smaller than 0.1.

Figure 6.9 – Delay of claim reporting trace plots

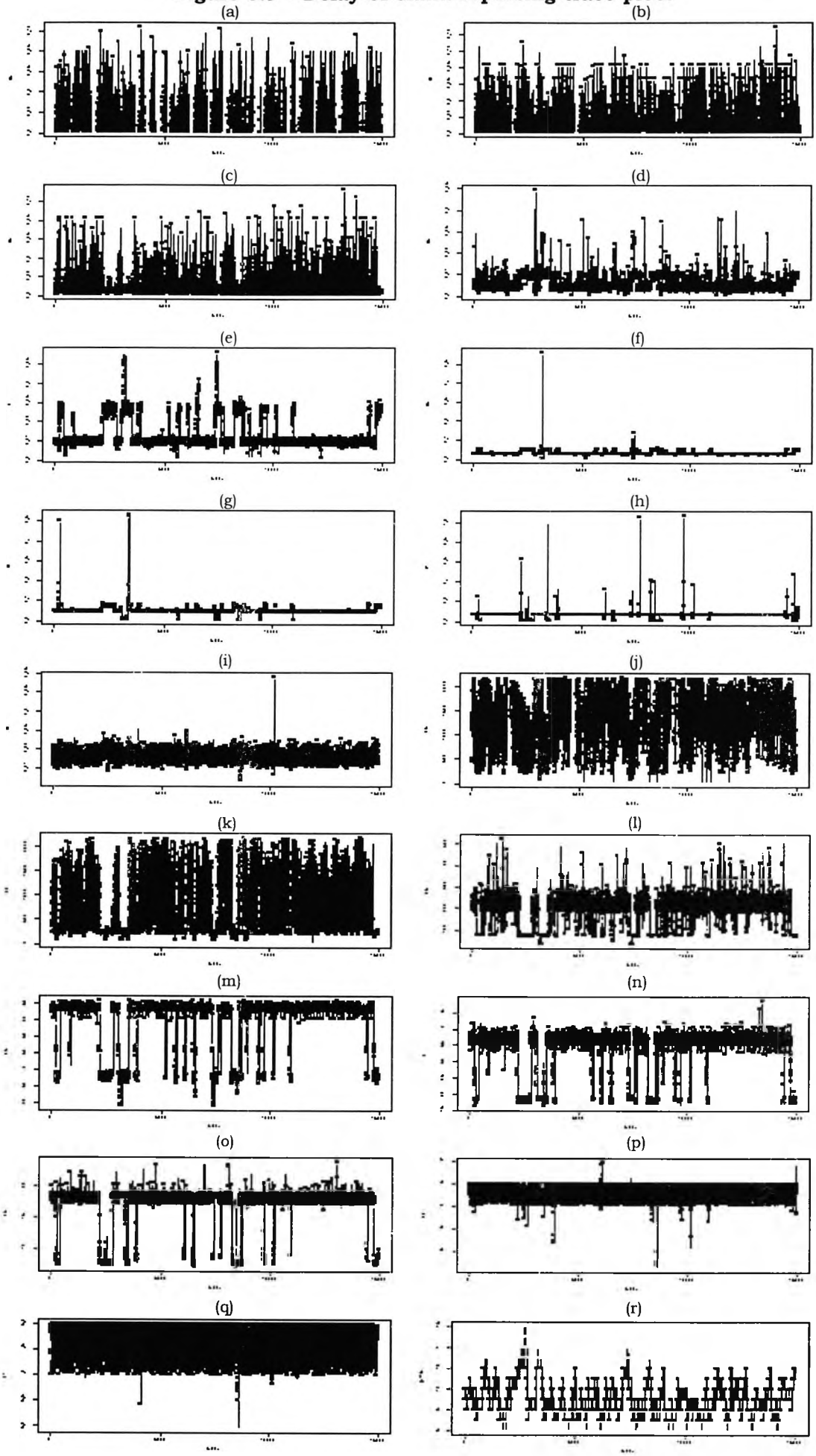
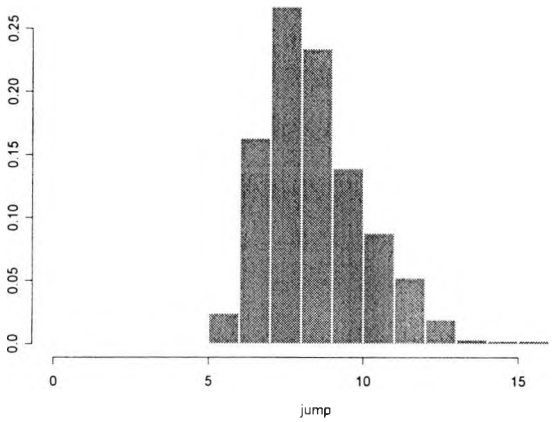
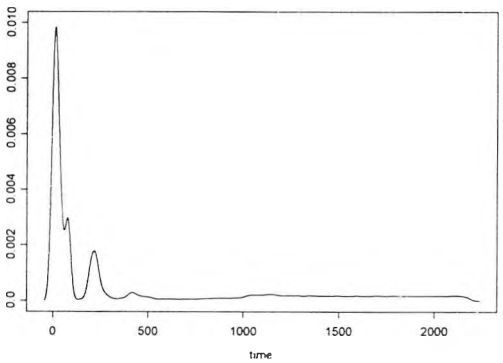


Figure 6.10 – Delay intensity empirical posterior distributions

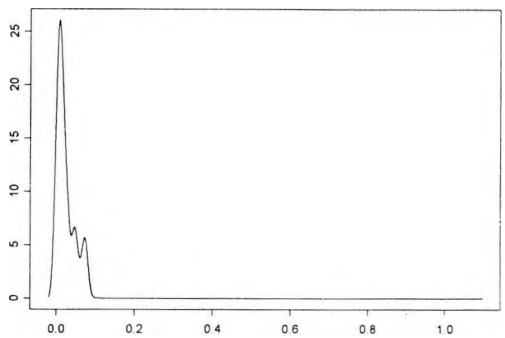
(a) Number of jumps



(b) Position of jumps



(c) level parameters



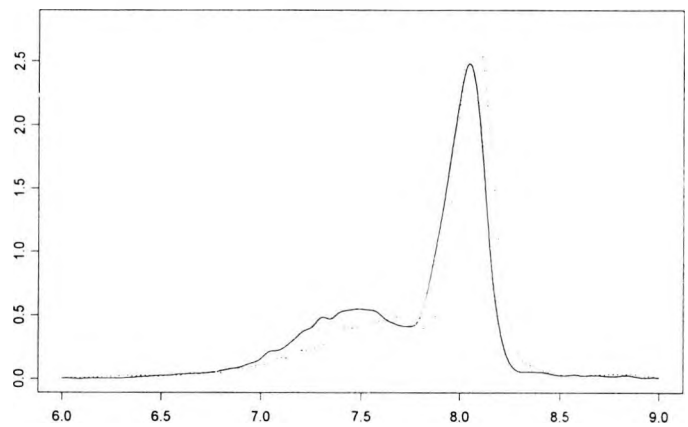
6.5.4 Age, sex parameter of severity

The severity of claim in the model presented here only depends on the sex and age of the policyholder and those are the parameters analysed in this subsection. The convergence is fairly good for both male and female mean parameters, showing for each two modes very close together. In contrast, the position parameter does not converge to any obvious value and some reasons for this behaviour will be discussed later in chapter 7. Generally speaking, the age effect does not seem to vary by sex covariate, as is observed from the similar results for both sets of parameter.

More specifically, male and female mean parameters show a very similar behaviour. Their posterior distribution is shown in figure 6.11 over the interval (6, 9). It is observed that female (dotted line) has a higher density on the higher mode around 8, while male (full line) has a more obvious second

peak. These results justify the higher density observed for one jump in the male case than in the female case.

Figure 6.11 – Female and male mean empirical posterior distributions



Female mean:

Convergence (traces in figure 6.12):

Number of jumps (plot (j)) – Good mixing, with higher probability values at 0 and 1.

Position of jumps (plots (f) to (i)) – s_4 and s_3 are not frequent enough to draw any conclusion. s_2 does not show any obvious convergence, except the fact of being bigger than 16. s_1 has no convergence either, mixing over the whole range of the jump position. There is a higher concentration on values below 16.

Level parameters (plots (a) to (e)) – l_4, l_3 and l_2 are not frequent enough to draw any conclusion. l_1 converges to values around 8 and l_0 is more sparse but still with some convergence to values 8 and below.

Analysis of empirical posterior distributions (Figure 6.13):

Number of jumps – Values 0 and 1 have the highest probability.

Position of jumps – The graph points to two regions of higher density. The first one is around 15 and the second one bigger than 60.

Level parameters – There is only one region with higher density (values around 8), where two very close modes are observed.

Figure 6.12 – Female severity trace plots

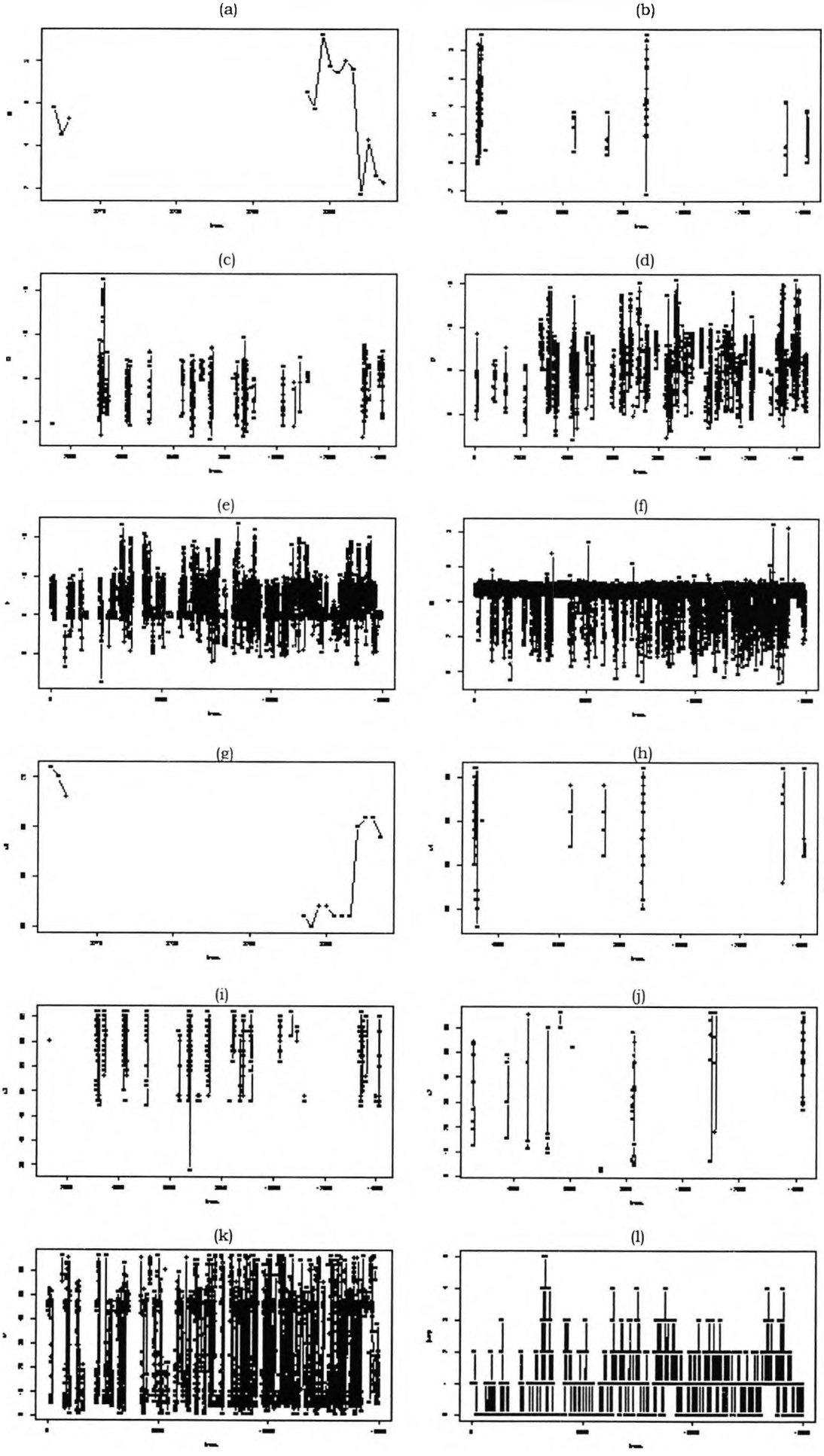
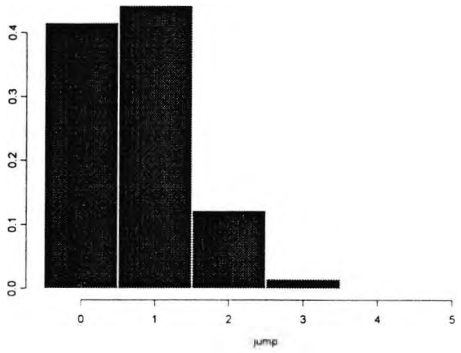
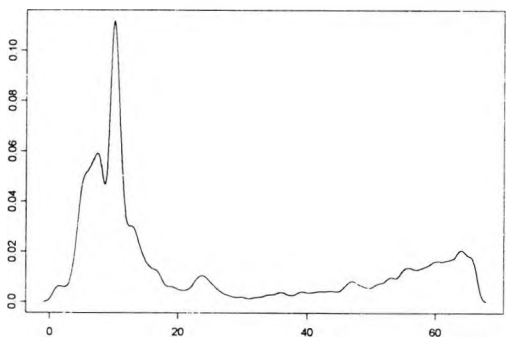


Figure 6.13 – Female mean empirical posterior distributions

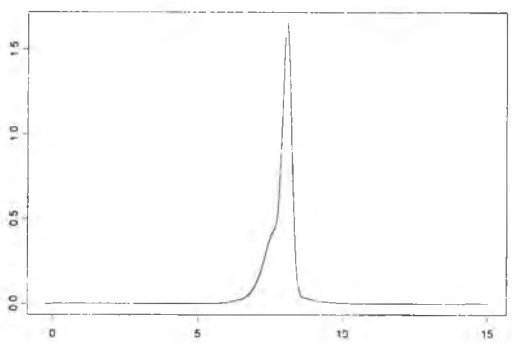
(a) Number of jumps



(b) Position of jumps



(c) level parameters



Male mean:

Convergence (traces in figure 6.14):

Number of jumps (plot (j)) – Good mixing, with higher probability values at 0 and 1.

Position of jumps (plots (f) to (i)) – s_4 and s_3 are not frequent enough to draw any conclusion. s_2 does not show any obvious convergence, except the fact of being bigger than 16. s_1 seems to converge to values below 16, but still has a high density over the whole range of the jump position.

Level parameters (plots (a) to (e)) – l_4 and l_3 are not frequent enough to draw any conclusion. l_2 has few values converging to values around 10 and l_1 to values around 8. l_0 is highly concentrated on values between 7 and 8.

Figure 6.14 – Male severity trace plots

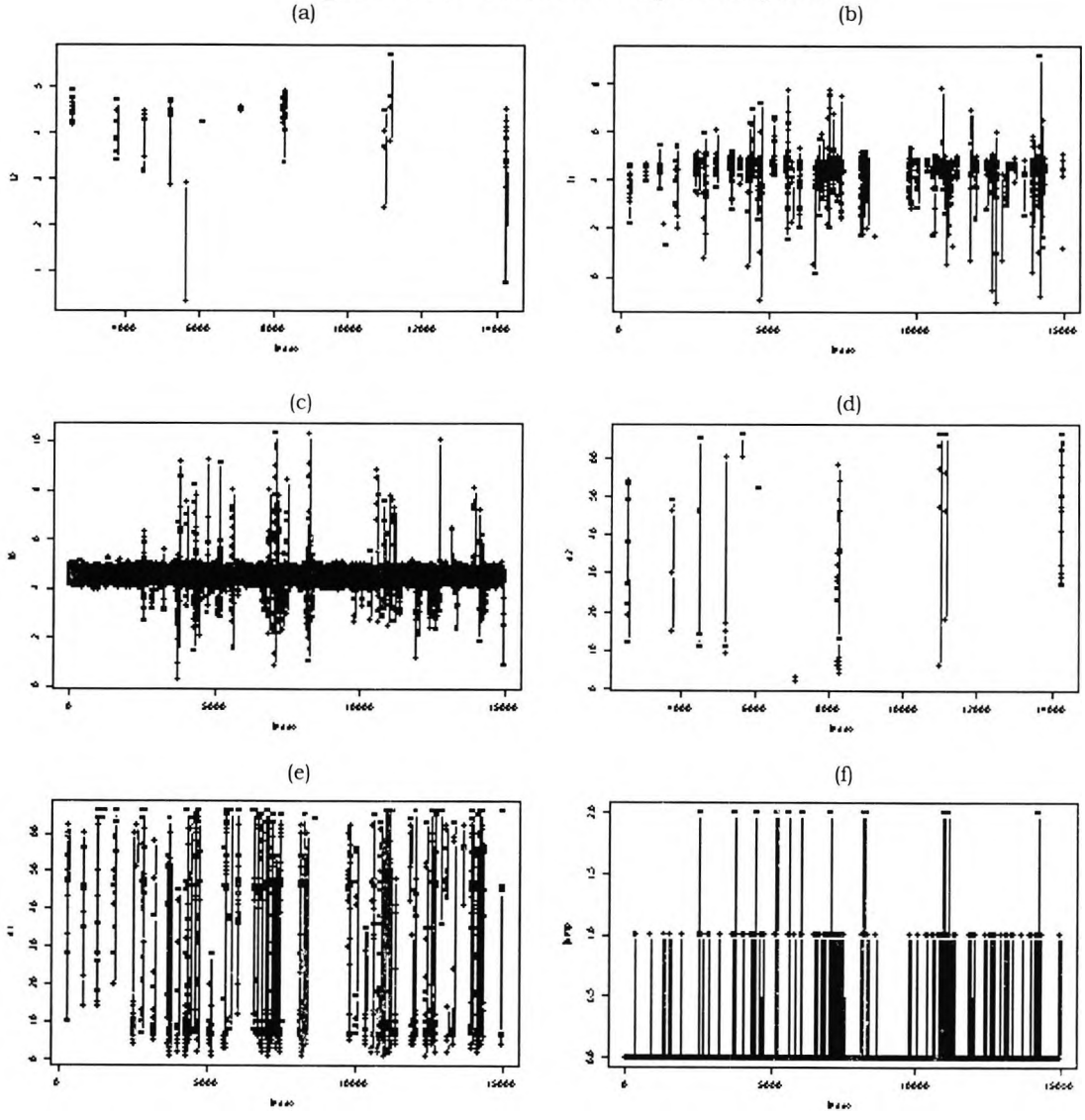
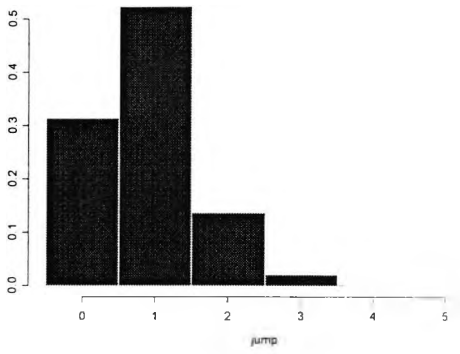
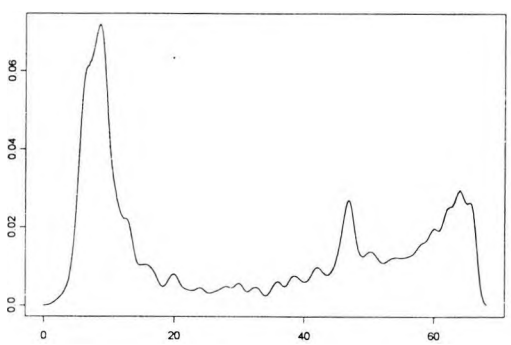


Figure 6.15 – Male mean empirical posterior distributions

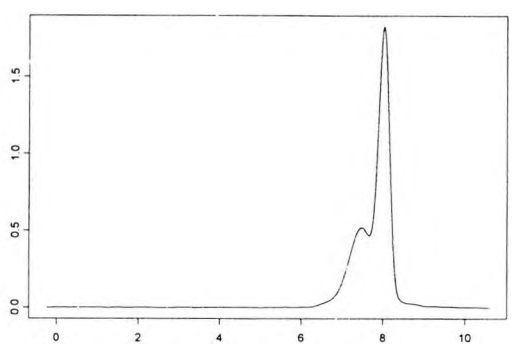
(a) Number of jumps



(b) Position of jumps



(c) level parameters



Analysis of empirical posterior distributions (Figure 6.15):

Number of jumps – Value 1 has the highest probability.

Position of jumps – The graph points to two regions of higher density. The first one is around 15 and the second one bigger than 40.

Level parameters – There are two modes very close together: one around 7.5 and the other one around 8.

6.6 Results comparison

Here the results presented in the previous section are compared to two methodologies of claims reserving. Firstly the results from Arjas and Haastrup (1996) are presented, which have a similar approach to the previous sections. Secondly some more commonly used techniques are shown.

Generally speaking the results are more similar on the number of IBNR claims analysis than on the amount of the reserve. This behaviour can be

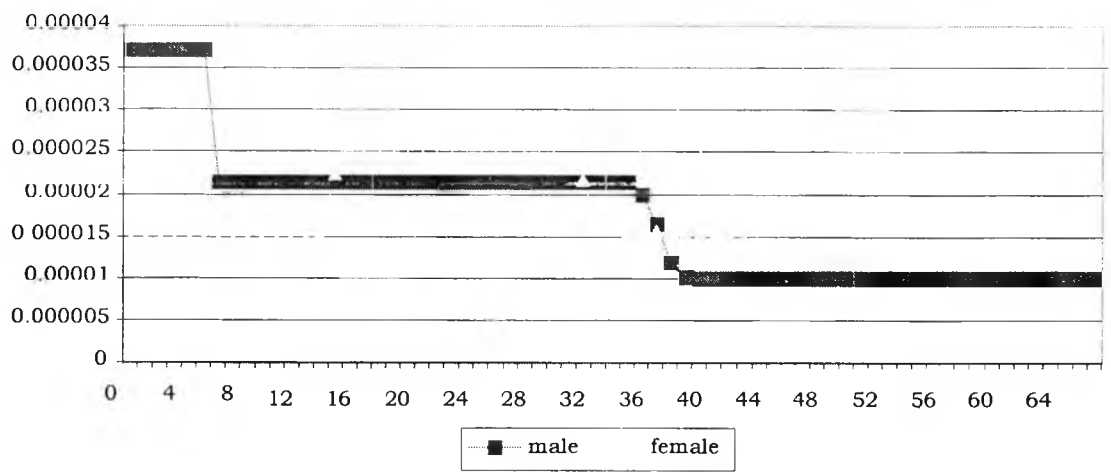
explained by the inherent higher uncertainty involved in the projection of claim amounts than on their reporting. After reporting a claim its settlement process is just starting.

6.6.1 Comparison with Arjas and Haastrup (1996)

The frequency of claims component of the model used in this thesis is very similar to the one in Arjas and Haastrup (1996), and so are the results. The observed value of IBNS claims is in the 39% quantile of the posterior distribution for Arjas and Haastrup (1996), and in 47% on the model presented here.

Returning to the individual parameter analysis it is possible to observe that again the intensity of reporting delay has the same behaviour: a high concentration of small values, being skewed to the left. In the age, sex occurrence parameters it is observed that by calculating the intensity level parameter for each age the posterior average per age is very similar for both models. Now, the model adopted here has the advantage of defining groups of ages. The posterior average is shown in table 16 below.

Figure 6.16 – Posterior mean of intensity of claim occurrence by sex and age



In the severity part of the model, the approach in here is completely different from the one in Arjas and Haastrup (1996). Here, age, sex and parametrical approach are used, while in Arjas and Haastrup (1996) a non-parametrical approach used the information about whther the payment was zero or not and if it was related to a settlement or not. The choice of modelling the payment based on age and sex does not seem to have enhanced the analysis, which could be improved by using both the settlement delay and the

claim type information. A further discussion on this point is presented in the conclusion of this chapter.

The projection from Arjas and Haastrup (1996) is shown in table 6.7 for both the reserves and number of IBNR claims. Values are in thousands and the main comparison factor is the quantile of the predictive distribution from Arjas and Haastrup (1996) and the model in this thesis.

Table 6.7 – Summaries of the projection in Arjas and Haastrup (1996)

	n^{ibnr}	R^{ibnr}	R^{rbns}
Mean	-	337	2090
90% confidence interval	-	(229,468)	(1780,2410)
Observed quantile by Arjas and Haastrup (1996)	39%	9%	18%
Observed quantile by proposed model	47%	2%	83%

Also, the model in this thesis had the observations in a higher quantile than in Arjas and Haastrup (1996). It also has a wider confidence interval, showing a greater variability of the posterior distribution.

The benefit of using the approach proposed in this thesis is the clarification of the level of mixture in the parameter level. Now it is possible to analyse for each parameter separately how many piecewise constant values are necessary for each of them, which seems an important feature on the age, sex intensity of occurrence. Again, the use of a higher number of samples (15000 compared with 1500 in Arjas and Haastrup (1996)) is also more reassuring, since it is known how important it is to have a large sample in order to analyse the results in a MCMC implementation.

6.6.2 Comparison with traditional methods

In order to have a wider idea of how the model presented in this work enhances the usual prediction of claims reserve, a comparison to two traditional reserving methods is shown in this section. The chain ladder and the Bornhuetter-Ferguson (Bornhuetter and Ferguson (1972)) methods are used to predict the total claim reserve and also the unreported number of claims (both methods can be found in Hart, Buchanan and Howe (1996)). The chosen method of performing the methods was to calculate the factors based on the theory and then apply these to calculate the reserves. In this process a

practical actuary usually selects the factor values instead of using the theoretical value, but here judgement is only used for the tail factor, if necessary.

The Bornhuetter-Ferguson method needs an exposure per accident year and in this example the measure available is the number of policies in force at the end of the year. These values are shown in figure 6.17, together with the development of the number of reported claims.

Figure 6.17 – Triangle of number of claims

Accident year	**1982**	**1983**	**1984**	**1985**	**1986**	**1987**
Exposure	15,511	17,906	22,006	28,586	34,866	39,596

Acc. Year	1	2	3	4	5	6
1982	254	277	278	279	279	279
1983	264	288	289	290	290	
1984	413	449	451	451		
1985	447	498	502			
1986	542	600				
1987	675					
Factors	1.100	1.005	1.002	1.000	1.000	

Factors are calculated as in the chain ladder model, which does not allow for the tail factor. In the frequency data no tail factor is needed, since the historical development shows that all claims had been reported by the third year. This is not true for the development of the claims payment which is shown below in figure 6.18. Values are in thousands.

Figure 6.18 – Triangle of paid claims

Acc. Year	1	2	3	4	5	6
1982	555	828	857	888	912	917
1983	722	1,117	1,140	1,139	1,139	
1984	1,199	1,701	1,767	1,802		
1985	1,253	1,869	1,947			
1986	1,561	2,191				
1987	2,047					
						Tail
Factors	1.456	1.036	1.017	1.012	1.005	**?**

It is possible to observe that the history of claims points to a longer development than only five years, but since no data is available for older years

neither the claim ladder nor the Bornhuetter-Ferguson methods give us a way to calculate its value via the data. So the choice will be fully based on the judgement of the analyst. In order to show how the result changes with the tail selected, the table 6.8 below shows the Claims reserve for various choices.

Table 6.8 – Summaries of the projection in traditional methods

	n° IBNR claims	Value of Total Reserve (IBNR + RBNS)				
		(a) 1.020	(b) 1.014	(c) 1.216	(d) 1.089	(e) 1.122
Chain ladder	78	1,637	1,565	3,880	2,430	2,800
Bornhuetter-Ferguson	71	1,160	1,105	2,591	1,725	1,964

The number of IBNR claims seems to be projected correctly, but the chain ladder method seems to slightly overestimate its value. In the reserve part the tail in (a) was chosen by exponentially smoothing the historical factors and applying some rounding. (b) and (c) are based on calculating back the tail that gives us the mean and 95% quantile as estimated by the model presented in this work. (d) and (e) are the respective for the Arjas and Haastrup (1996) analysis.

Recall that the observed payments for this reserve in the database is 2,157, and since this amount is still expected to increase, the choice in (a), which is the one that is the considered as the most probable analyst's choice, underestimates the final value. This also happens for (b) and for (d) and (e) on the Bornhuetter-Ferguson method. (c) is the only one that does not underestimate the value of the reserve, but its tail factor seems quite high and probably would not be selected by an analyst who only has the historical data from which to make a decision.

Apart from the probable low value of the reserve in this example, when using a traditional method there is no split into IBNR and RBNS reserves, which is a disadvantage for the analysis.

6.7 Chapter conclusion

In this chapter the smoothing property of the model presented in chapter 4 was explored. The micro approach uses individual information and continuous time, modelling the claim development. It is shown how the idea from Norberg (1993,1999) and used in Arjas and Haastrup (1996) can be

applied using RJMCMC. This technique gave a better understanding of the mixing process in the parameter level.

In the severity part of the model a modification from Arjas and Haastrup (1996) was also proposed, using a parametrical distribution dependent on age and sex. This is a nice modification since it imposes a structure that facilitates the prediction and a likely distributional behaviour. In order to predict the reserves better it would be interesting to use not only the parametrical approach, but also the dependency on the size of payment, which was shown to be quite influential in this data set, and also the settlement delay. The analysis for the settlement delay is shown is shown in figure 6.19 below.

Figure 6.19 – Claim average by settlement delay

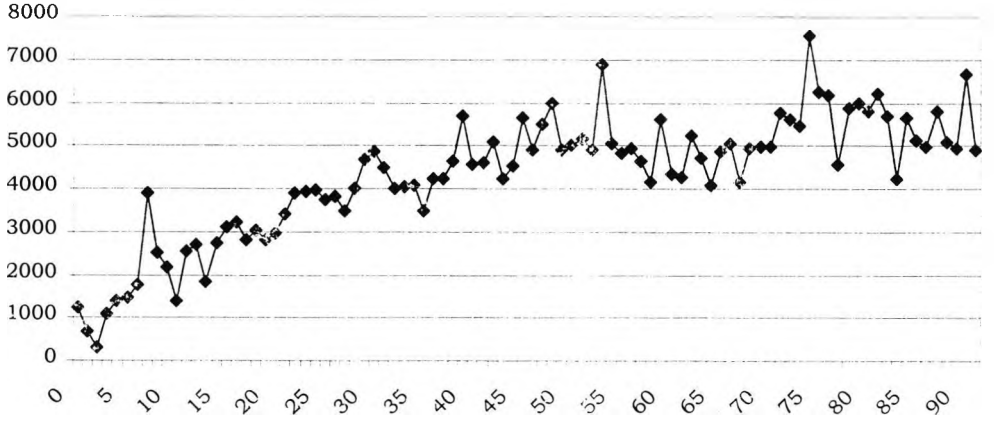
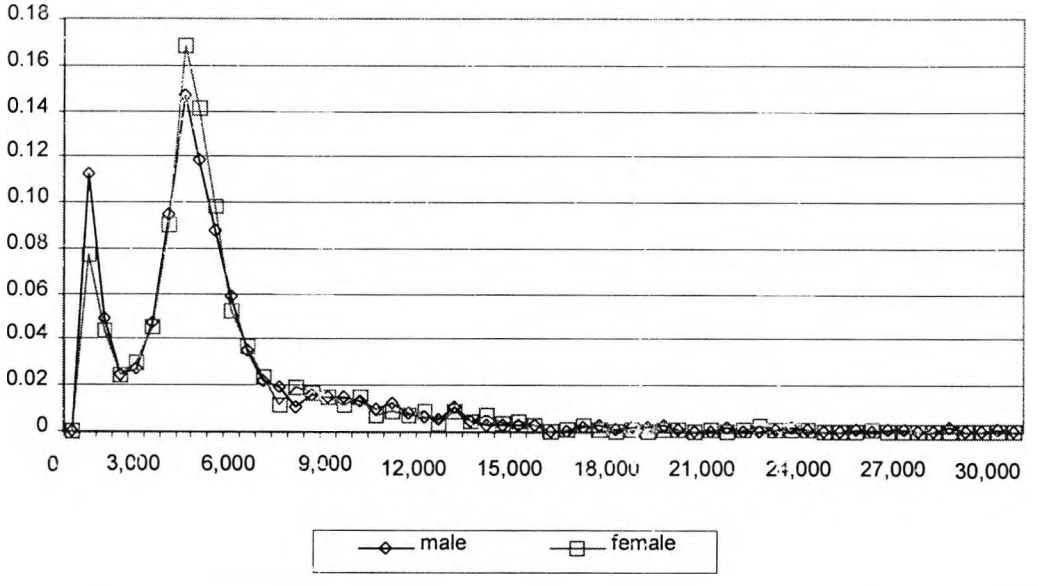


Figure 6.20 – Value of settled claim per sex



On the total value of settled claim the graph above in figure 6.20 shows the data histogram by sex. It is observed that no differentiation is observed by sex, confirming what has already been observed on the age, sex factor.

It is expected that the observed difference in settlement value is due to the type of claim: low values (for consultations) and high values (accidents). The problem with an approach bringing any of those characteristics into the model is the big computational burden that would be required.

The next chapter is the conclusion of the thesis. A summary of the work together with some proposals for new research is presented.

Chapter 7

Conclusion

In this final chapter an overview of the work in this thesis is presented. Some comments on the model limitations and future research issues are included in the following section.

7.1 Final remarks

In this thesis, two new applications of micro models were presented. In both risk assessment problems a process based approach, looking at individual information underlying behaviour, is used. It was observed that one of the nicest features of this type of model is that now there is no need to look for a model that simplifies the data before any analysis, as is the usual approach.

The implementation of such a model was possible by using the Bayesian philosophy and a MCMC class of simulation technique. RJMCMC methodology was the core of the applications in this work. To allow changes of the dimensionality of the parameter set is not a simple issue and RJMCMC extends the MCMC in a very straightforward way. MCMC simulation class has been proven to be quite powerful and via the results presented in here, so has RJMCMC.

Most of the RJMCMC problems observed in the applications presented here are related to practical issues. The fact that RJMCMC is a new technique means that there have not been enough studies of the convergence, which is a big concern for simulation techniques. Future research in this area is even more necessary since the usual MCMC convergence tests are not directly

applicable given that the interpretation of the parameter can change in every simulation. It is expected that the analysis of the level and position of jump parameters conditional on the number of jumps is the reasonable one.

As far as the results are concerned, most of the severity analysis had a worse result if compared to the frequency cases. It is understood that this is due to the fixing of the variance of the data distribution. This is a very limiting assumption, chosen due to the large extra computational burden when attaching a distribution to this parameter. In order to enhance the severity model, it is really desirable to use a stochastic variance instead of a fixed amount. It is not an easy question since on one hand it would probably bring some more uncertainty to the mean parameter if only one stochastic variance was defined. On another hand, choosing a stochastic variance that would also vary per group would increase enormously the computational burden.

It would also be interesting to check how much extra computational time is needed by including the Gibbs Sampling chain in the implementation. It is understood that this extra step is more reassuring for the results, than an ordinary RJMCMC chain, and was used, for instance, in Green (1995). But from another point of view it may considerably increase the computational burden.

An area where further research is necessary is how to choose more formally the groups. The definition of the best number of groups, together with the respective premium and variability is still an open question. How to decide on the optimal grouping for the premium, which is a function of separate analyses for the frequency and severity is not clear.

In chapter 6, the application of claims reserving could be enhanced for the severity analysis by using a bimodal distribution (or a mixture over claim values). It was also observed that the claim size is dependent on the settlement delay.

Also in the claims reserving model, it was observed that the multiplicative structure for the frequency of claim (calendar versus age, sex effect) could cause convergence problems. The choice of fixing one set of parameters and updating the other drastically increased the computational burden involved.

Although the models proposed in this work have been shown to be more adequate than some traditional analyses (the aggregation using age as discrete in chapter 5 and traditional reserving methods in chapter 6) it is not expected that the models presented here will be widely used on an everyday basis.

Among the limitations of applying Bayesian theory in practice already cited in chapter 3, the implementation time and the lack of portability are most likely to affect this model.

After applying these models based on individual data, when assumptions are made directly in the process itself rather than on the aggregated data, it is obvious that a great amount of time is needed for their implementation. But it was also clear that this approach does enhance the risk assessment process. It is expected that as the implementation time decreases, thesis class of models may became more widely used.

References and Bibliography

_____ (1986) Special issue on credibility theory; Insurance Abstract and Reviews vol. 2, Feb.1986, n° 3.

_____ (1998) WinBUGS Examples; MRC Biostatistcs Unit Cambridge.

_____ (1998) WinBUGS Manual; MRC Biostatistcs Unit Cambridge.

Albrecht (1985) An evolutionary credibility model for claim numbers; Astin Bulletin, vol. 15, n°1.

Arjas, E. and Gasbarra, D. (1994) Nonparametric Bayesian inference from right censored survival data, using the Gibbs sampler. Statistica Sinica 4, pp 505-524.

Arjas, E. and Haastrup, S. (1996) Claims reserving in continuous time; a nonparametric Bayesian Approach. ASTIN Bull. Vol 26, no.2, pp 139-164.

Bailey, A (1950) Credibility procedures, Laplace's generalisation of Bayes' rule and the combination of collateral knowledge with observed data; Proceedings of the Casualty Actuarial Society, vol 37.

Best, N.G., Gilks, W.R. and Tan, K.K.C. (1994) Adaptive Rejection Metropolis Sampling within Gibbs Sampling, Internal report.

Best,N., Spiegelhalter,D. and Thomas,A. (1998) WinBUGS User Manual version 1.1.1

Bornhuetter, R. and Ferguson, R. (1972), The actuary and IBNR, Proceedings of the Casualty Actuarial Society, Vol. LIX, Chicago.

Boskov, M. and Verrall, R.J. (1994) Premium rating by geographic area using spatial models. ASTIN Bull. Vol 24, no. 1, pp 134-143.

Bühlmann, H. (1967) Experience rating and credibility; Astin Bulletin, vol 4.

Bühlmann, H. and Straub (1970) Credibility for loss ratios; Bulletin of the Association of Swiss Actuaries, vol. 70.

Bühlmann, H. and Jewell (1987) Hierarchical credibility revisited; Bulletin of the Association of Swiss Actuaries.

Carlin, B.P. (1992) A simple Monte Carlo approach to Bayesian graduation; Transactions of Society of Actuaries, vol XLIV, pp 55-76.

Chib, S. and Greenberg, E. (1994) Understanding the Metropolis-Hastings Algorithm. University Report.

Chrissi, D. (1997) Claims reserving under a Bayesian approach using BUGS; Master dissertation; City University: London.

Dannenburg, D. (1993) Some results on the estimation of the credibility factor in the classical Bühlmann model; XXIV Astin Colloquium.

DeGroot, M.H. (1986) Probability and statistics: second edition; Addison-Wesley publishing company: USA.

Dellaportas, P. and Ntzoufras, I. (1997) Bayesian Prediction of Outstanding Claims. Athens University of Economics and Business. University Report.

Dellaportas, P., Karlis, D. and Xekalaki, E. (1997) Bayesian Prediction of Outstanding Claims. Athens University of Economics and Business. Working paper.

Denison, D. G. T., Mallick, B. K. and Smith, A. F. M. (1998) Automatic Bayesian curve fitting. J.R.Statistic Society B, 60, Part 2, 333-350

Gamerman, D and Migon, H. (1993) Inferência estatística: uma abordagem integrada; Mathematics Institute, Federal University of Rio de Janeiro - UFRJ.

Gamerman, D. (1997) Markov chain Monte Carlo: stochastic simulation for Bayesian inference. London: Chapman & Hall.

Gelfand, A.E. and Smith, A.F.M. (1990) Sampling-based approaches to calculating marginal densities. Journal American Statistical Society B, 55, 72-3.

Geman, S. and Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. I.E.E.E. Transactions on Pattern Analysis and Machine Intelligence, 6, 721-41.

Gilks, W.R., Richardson, S. and Spiegelhalter,D. J. (1996) Practical Markov Chain Monte Carlo. London: Chapman & Hall.

Goovaerts, M.J and Hoogstad, W.J. (1987) Credibility theory; Surveys of Actuarial Studies, n° 4, Nationale-Nederlanden N.V.

Goovaerts, M.J, Kaas, R., Van Heerwaarden, A.E. and Bauwelinckx, T. (1990) Effective actuarial methods; Elsevier Science Publishing Company, Holland.

Green, P. J. (1995) Reversible jump MCMC computation and Bayesian model determination. Biometrika, 82, 711-732.

Green, P. J. and Richardson, S. (1997) On Bayesian analysis of mixtures with an unknown number of components. J.R.Statistic Society B, 59, no.4, pp 000-000.

Haastrup, S. (1997) Some fully Bayesian micro models for claims reserving, PhD thesis, University of Copenhagen.

Haastrup, S. (1999) Maximum likelihood estimation in a marked point process with applications to non-life insurance. Internal Report.

Haberman, S. (1996) Landmarks in the history of actuarial science (up to 1919) ; Research report, City University: London/UK.

Hachemeister (1975) Credibility for regression models with application to trend; Credibility: theory and applications, Proceedings of the Berkeley Actuarial Research Conference on credibility, Academic Press.

Hart, Buchanan and Howe (1996), Actuarial Practice of General Insurance. Institute of Actuaries of Australia, Sydney/Australia.

Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika, vol 57, pp 97-109.

Jewell, W. S. (1974) Credible means are exact Bayesian for exponential families; Astin Bulletin,vol.8, n°1.

Jewell, W. S. (1975) The use of collateral data in credibility theory: a hierarchical model; Giornale dell'Istituto Italiano degli Atuari, vol 38.

Jewell, W. S. (1976) A survey of credibility theory; Operations Research Center, Research Report n° 76 - 3, Berkeley.

Jewell, W. S. (1989) Predicting IBNYR events and delays: I, Continuous time. Astin Bulletin, vol 19, 25-56.

Jewell, W. S. (1990) Predicting IBNYR events and delays: II, Discrete time. Astin Bulletin, vol 20, 93-111.

Jong and Zehnwirth (1983) Credibility theory and the Kalman filter; Insurance: Mathematics and Economics, vol 2.

Kimeldorf, G.S. and Jones, D.A. (1967) Bayesian graduation; Transactions of Society of Actuaries, vol XIX, pp 66.

Kling, B (1993) A note on iterative non-linear regression in credibility; XXIV Astin Colloquium.

Klugman, S. A. (1992) Bayesian statistics in actuarial science with emphasis on credibility theory; Boston: Kluwer.

Kouyoumoutzis, K (1998) Monitoring mortality over time; Master dissertation; City University: London.

Kremer (1982) Exponential smoothing and credibility theory; Insurance: Mathematics and Economics, vol 1, n° 3.

Ledolter, J. , Klugman, S. and Lee, C.S. (1990) Credibility models with time-varing trend components, Astin Bulletin, vol 21, n° 1.

Lemaire, J. (1977) Selection procedures of regression analysis applied to automobile insurance. Bullein of the Association of Swiss Actuaries, vol 77, no. 2, pp 143-160.

Liu,Y.-H, Makov, U.E. and Smith, A.F.M. (1996) Bayesian methods in actuarial science. The Statistician, 45, no.4, pp 503-515.

London, D. (1985) Graduation: the revision of estimates; ACTEX Publications: USA.

Longley-Cook (1962) An introduction to credibility theory; Proceedings of the Casualty Actuarial Society, vol 49.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953) Equations of state calculations by fast computing machines. J.Chem.Phys., vol 21, 1087-1091.

Mowbray, A. (1914) How extensive a payroll exposure is necessary to give a dependable pure premium?; Proceedings of the Casualty Actuarial Society, 1, pp 24-30.

Norberg, R (1992) Linear estimation and credibility in continuous time. Astin Bulletin, vol 22, 149-165.

Norberg, R (1993) Prediction of outstanding liabilities in non-life insurance. Astin Bulletin, vol 23, n° 1, 95-115.

Norberg, R (1999) Prediction of outstanding claims II: Model variations and extensions. Astin Bulletin, vol 29, n° 1, 5-25.

Pereira, F.C. (1998) Teoria da credibilidade: uma abordagem integrada. Caderno tese. Funenseg: Rio de Janeiro/BR.

Pereira, F.C. and Verrall, R.J. (1998) A Markov chain Monte Carlo approach to grouping premium rating factors. (to appear) .

Pereira, F.C. (1999) Practical "Modern" Bayesian Statistics in Actuarial Science. Proceedings of 1999 GIRO Convention. Brighton/UK.

Philips, D.B. and Smith, A.F.M. (1996) Bayesian model comparison via jump diffusions. In Practical MCMC (eds W.R. Gilks, S. Richardson and D.J. Spiegelhalter) , ch 13, pp. 215-239. London: Chapman and Hall.

Scollnik, D.P.M. (1996) An introduction to Markov chain Monte Carlo methods and their actuarial applications. Proceedings of the Casualty Actuarial Society, vol LXXXIII, n° 158.

Sundt (1982) Invariantly recursive credibility estimation; Insurance: Mathematics and Economics, vol 1, n° 3.

Sundt (1983) Finite credibility formulae in evolutionary models; Scandinavian Actuarial Journal, n°2

Sundt (1987) Credibility estimators with geometric weights; XX Astin Colloquium Scheveningen.

Taylor, G.C. (1989) Use of spline functions for premium rating by geographic area. Astin Bulletin, vol 19, n° 1, pp 91-122.

Taylor, G.C. and Ashe, F.R. (1983) Second moments of estimates of outstanding claims. Journal of Econometrics, vol 23, pp 37-61.

Tierne, L. (1994) Markov chain for exporing posterior distributions (with discussions) Annals of Statistics, Vol 22, no. 4, 1701-1762.

Verrall, R.J. (1990) Bayes and empirical Bayes estimation for the chain ladder model. Astin Bulletin, vol 20, n° 2, 217-243.

Verrall, R.J. and Yakoubov, Y. (1999) A fuzzy approach to grouping by policyholder age in General insurance. Journal of Actuarial Practice, vol 7, pp 181-203.

Waters, H.R. (1987) Special note: an introduction to credibility theory; Institute of Actuaries and Faculty of Actuaries.

Whitney, A. (1918) The theory of experience rating; Proceedings of the Casualty Actuarial Society, vol 4.

Whittaker, E.T. (1923) On a new method of Graduation. Proceedings Edinburg Mathematics Society, XLI, 63.

Wu, X. (1996) A Bayesian Discretizer for Real-Value Attributes. The Computer Journal, vol. 39, no. 8, pp 688-691.

Appendix A

A.1 Main Program

```
//      This program is to implement the calculation of the number of jumps
//      and related hazard rate for the frequency and severity of claims
//      using the reversible jump technique
//program beginning
#include "library/completo.h"
main(){
//name of files
char *Nome[] ={"/usr/home/fc/ages_new/results/completo/bif_L",
               "/usr/home/fc/ages_new/results/completo/bif_S",
               "/usr/home/fc/ages_new/results/completo/bifpes_L",
               "/usr/home/fc/ages_new/results/completo/bifpes_S",
               "/usr/home/fc/ages_new/results/completo/bis_L",
               "/usr/home/fc/ages_new/results/completo/bis_S",
               "/usr/home/fc/ages_new/results/completo/bislog_L",
               "/usr/home/fc/ages_new/results/completo/bislog_S"};
int i,j,z;
// defining parameter arrays
long double Lbif[kmax+1];              // history of levels
long double Sbif[kmax+2];              // history of jumps
Sbif[0]=0;
Lbif[0]=rgamma(Abif,Bbif,2);
int kbif=0;

long double Lbis[kmax+1];              // history of levels
long double Sbis[kmax+2];              // history of jumps
Sbis[0]=0;
Lbis[0]=rnorm(Abis,Bbis,1);
int kbis=0;

long double Lbislog[kmax+1];           // history of levels
long double Sbislog[kmax+2];  // history of jumps
Sbislog[0]=0;
Lbislog[0]=rnorm(Abis,Bbis,1);
int kbislog=0;

long double Lbifpes[kmax+1];           // history of levels
long double Sbifpes[kmax+2];  // history of jumps
Sbifpes[0]=0;
Lbifpes[0]=rgamma(Abif,Bbif,2);
int kbifpes=0;
//defining accumulation
int Tbif[3]={0,0,0};
int TbifA[2]={0,0};
int Tkbif[kmax+1];
for(i=0;i<=kmax;i++) Tkbif[i]=0;

int Tbifpes[3]={0,0,0};
int TbifpesA[2]={0,0};
int Tkbifpes[kmax+1];
for(i=0;i<=kmax;i++) Tkbifpes[i]=0;

int Tbis[3]={0,0,0};
int TbisA[2]={0,0};
int Tkbis[kmax+1];
for(i=0;i<=kmax;i++) Tkbis[i]=0;

int Tbislog[3]={0,0,0};
int TbislogA[2]={0,0};
int Tkbislog[kmax+1];
for(i=0;i<=kmax;i++) Tkbislog[i]=0;
//defining data arrays
double Xbi[sbipes];                    // observation - age
double Wbi[sbipes];                    // observation - weight
double WTbi[sbipes];          // observation - accumulate weight
```

```
double Xbivar[sbivar];                  // observation - number of claims
double Hbi[sbivar];                     // observation - number of claims
double HTbi[sbivar];            // observation - number of claims
double Hbilog[sbivar];                  // observation - number of claims
double HTbilog[sbivar];                 // observation - number of claims
int Nbi[sbivar];                        // observation - number of claims
int NTbi[sbivar];                       // observation - accumulate weight
Xbivar[0]=0;
Hbi[0]=0;
HTbi[0]=0;
Hbilog[0]=0;
HTbilog[0]=0;
Nbi[0]=0;
NTbi[0]=0;
//file data
ifstream fbivar("/usr/home/fc/ages_new/banco_de_dados/bivar.txt");
if(!fbivar) {cout << "Cannot open file for input\n"; exit(1);}
fbivar.seekg(0);

ifstream fbipes("/usr/home/fc/ages_new/banco_de_dados/pesobi.txt");
if(!fbipes) {cout << "Cannot open file for input\n"; exit(1);}
fbipes.seekg(0);

for(i=1;i<sbivar;i++) {
fbivar >> Xbivar[i] >>Hbi[i]>>Hbilog[i]>>Nbi[i];
NTbi[i]=NTbi[i-1]+Nbi[i];
HTbi[i]=HTbi[i-1]+Hbi[i];
HTbilog[i]=HTbilog[i-1]+Hbilog[i];
}

for(i=0;i<sbipes;i++) {
fbipes >> Xbi[i] >> WTbi[i];
if (i==0) Wbi[0]=0;
else Wbi[i]=WTbi[i]-WTbi[i-1];
}

fbivar.close();
fbipes.close();
//initiating output files
ofstream sail("/usr/home/fc/ages_new/results/completo/bif_L");
ofstream sais("/usr/home/fc/ages_new/results/completo/bif_S");
sail.close();
sais.close();
for(i=2;i<6;i++){
ofstream sail(Nome[i]);
ofstream sais(Nome[i+1]);
i++;
sail.close();
sais.close();
}
// calculating number of jumps
numero(kbif);
numero(kbis);
numero(kbifpes);
numero(kbislog);
// calculating the first set of jumps and hazards
if(kbis>0) {
long double uni[2*kbis+1];
for(i=0;i<(2*kbis+1);i++) uni[i]=un(4+15);
sort(uni,2*kbis+1);
for(i=1;i<=kbis;i++){
Lbis[i]=rnorm(Abis,Bbis,2);
Sbis[i]=uni[2*i-1]*Smax;}}
Sbis[kbis+1] = Smax;

if(kbislog>0) {
long double uni[2*kbislog+1];
for(i=0;i<(2*kbislog+1);i++) uni[i]=un(4+15);
sort(uni,2*kbislog+1);
for(i=1;i<=kbislog;i++){
Lbislog[i]=rnorm(Abis,Bbis,2);
Sbislog[i]=uni[2*i-1]*Smax;}}
Sbislog[kbislog+1] = Smax;

if(kbis>0) {
long double uni[2*kbis+1];
for(i=0;i<(2*kbis+1);i++) uni[i]=un(4+15);
sort(uni,2*kbis+1);
for(i=1;i<=kbis;i++){
Lbis[i]=rnorm(Abis,Bbis,2);
```

```
Sbis[i]=uni[2*i-1]*Smax;}}
Sbis[kbis+1] = Smax;

if(kbif>0) {
long double uni[2*kbif+1];
for(i=0;i<(2*kbif+1);i++) uni[i]=un(4+15);
sort(uni,2*kbif+1);
for(i=1;i<=kbif;i++){
Lbif[i]= rgamma(Abif,Bbif,2);
Sbif[i]=uni[2*i-1]*Smax;}}
Sbif[kbif+1] = Smax;

if(kbifpes>0) {
long double uni[2*kbifpes+1];
for(i=0;i<(2*kbifpes+1);i++) uni[i]=un(4+15);
sort(uni,2*kbifpes+1);
for(i=1;i<=kbifpes;i++){
Lbifpes[i]= rgamma(Abif,Bbif,2);
Sbifpes[i]=uni[2*i-1]*Smax;}}
Sbifpes[kbifpes+1] = Smax;
// calculating jumps and hazards for ini+ite times
for (z=1;z<=ini+ite;z++){
frequencia(Sbif,kmax+2,Lbif,kmax+1,kbif,Xbi,Wbi,WTbi,sbipes,Xbivar,Nbi,NTbi,sbivar,Abif,
Bbif,TbifA,2,Tbif,3,4);
sail.open(Nome[0],ios::app);
sais.open(Nome[1],ios::app);

for(i=0;i<=kbif;i++) {sail<<Lbif[i]<<" ";sais<<Sbif[i]<<" ";}
sais<<"\n";
sail<<"\n";
sais.close();
sail.close();

frequencia(Sbifpes,kmax+2,Lbifpes,kmax+1,kbifpes,Xbi,Wbi,WTbi,sbipes,Xbivar,Nbi,NTbi,sbi
var,Abif,Bbif,TbifpesA,2,Tbifpes,3,2);
sail.open(Nome[2],ios::app);
sais.open(Nome[3],ios::app);

for(i=0;i<=kbifpes;i++) {sail<<Lbifpes[i]<<" ";sais<<Sbifpes[i]<<" ";}
sais<<"\n";
sail<<"\n";
sais.close();
sail.close();

severidade(Sbis,kmax+2,Lbis,kmax+1,kbis,Hbi,HTbi,Xbivar,Nbi,NTbi,sbivar,Abis,Bbis,sigbi,
TbisA,2,Tbis,3);
sail.open(Nome[4],ios::app);
sais.open(Nome[5],ios::app);

for(i=0;i<=kbis;i++) {sail<<Lbis[i]<<" ";sais<<Sbis[i]<<" ";}
sais<<"\n";
sail<<"\n";
sais.close();
sail.close();

severidade(Sbislog,kmax+2,Lbislog,kmax+1,kbislog,Hbilog,HTbilog,Xbivar,Nbi,NTbi,sbivar,A
bis,Bbis,sigbilog,TbislogA,2,Tbislog,3);
sail.open(Nome[6],ios::app);
sais.open(Nome[7],ios::app);

for(i=0;i<=kbislog;i++) {sail<<Lbislog[i]<<" ";sais<<Sbislog[i]<<" ";}
sais<<"\n";
sail<<"\n";
sais.close();
sail.close();

Tkbif[kbif]++;
Tkbis[kbis]++;
Tkbislog[kbislog]++;
Tkbifpes[kbifpes]++;
}//End for in z

cout<<"Sample for **bi frequency** had the following results:\n";
for(j=0;j < 2;j++) cout<<"Tbif["<<j+1<<"]= "<<Tbif[j]<<" "<<"TbifA["<<j+1<<"]=
"<<TbifA[j];
cout<<"Tbif[3]= "<<Tbif[2]<<"\n";
for(j=0;j <=kbif;j++) cout<<" Lbif["<<j<<"]= "<<Lbif[j]<<";Sbif["<<j<<"]=
"<<Sbif[j]<<";";
cout<<"\n";
```

```
cout<<"Sample for **bipes frequency** had the following results:\n";
for(j=0;j < 2;j++) cout<<"Tbifpes["<<j+1<<"]= "<<Tbifpes[j]<<" "<<"TbifpesA["<<j+1<<"]=
"<<TbifpesA[j];
cout<<"Tbifpes[3]= "<<Tbifpes[2]<<"\n";
for(j=0;j <=kbifpes;j++) cout<<" Lbifpes["<<j<<"]= "<<Lbifpes[j]<<";Sbifpes["<<j<<"]=
"<<Sbifpes[j]<<";";
cout<<"\n";

cout<<"Sample for **bi severity** had the following results:\n";
for(j=0;j < 2;j++) cout<<"Tbis["<<j+1<<"]= "<<Tbis[j]<<" "<<"TbisA["<<j+1<<"]=
"<<TbisA[j];
cout<<"\n";
cout<<"Tbis[3]= "<<Tbis[2]<<"\n";
for(j=0;j <=kbis;j++) cout<<" Lbis["<<j<<"]= "<<Lbis[j]<<" Sbis["<<j<<"]= "<<Sbis[j];
cout<<"\n";

cout<<"Sample for **bilog severity** had the following results:\n";
for(j=0;j < 2;j++) cout<<"Tbislog["<<j+1<<"]= "<<Tbislog[j]<<" "<<"TbislogA["<<j+1<<"]=
"<<TbislogA[j];
cout<<"\n";
cout<<"Tbislog[3]= "<<Tbislog[2]<<"\n";
for(j=0;j <=kbislog;j++) cout<<" Lbislog["<<j<<"]= "<<Lbislog[j]<<" Sbislog["<<j<<"]=
"<<Sbislog[j];
cout<<"\n";

}//endmain
```

A.2 Library

```
#include <fstream.h>
#include <math.h>
#include <stdlib.h>
#include <iostream.h>
// general constants
const long double Smax = 73.59;       //age upper value
const int ini  = 0;           //Initial iterations
const int ite  = 10000;       //Total iterations
const int kmax = 30;          //Maximum number of jumps
const float c = 0.4;          //Acceptance constant
//size of arrays
const int sbivar = 566;       //bivar.txt
const int sbipes = 5257;      //pesobi.txt

const int lambda = 3;         //a priori parameter of k
const int Abif = 12;          // a priori parameter of frequency
const int Bbif = 1000;        // a priori parameter of frequency
const int Abis = 50;          // a priori parameter of severity
const int Bbis = 100;         // a priori parameter of severity
const int sigbi= 150;         // a priori parameter of severity
const int sigbilog= 20;       // a priori parameter of severity
//****************** calculation of Uniform(0,1) *************************
long double un(const int loc){
long double a,e,x,m;
static long double
I[21]={1561,1448,9879,67961,791,989,989,34274,3568,4679,7869,57807,57901,589,22457,26783
7,37909,37679,37577,3757,3579};
switch(loc){
case 1: a=65539;
        e=31;
       break;
case 2: a=pow(13,3);
        e=31;
       break;
case 3: a=pow(5,13);
        e=39;
       break;
case 4: a=pow(5,17);
        e=42;
       break;
case 5: a=pow(11,13);
        e=31;
       break;
case 6: a=pow(11,13);
        e=47;
       break;
case 7: a=pow(13,3);
```

```
        e=31;
       break;
case 8: a=pow(5,13);
        e=39;
       break;
case 9: a=pow(5,17);
        e=42;
       break;
case 10: a=pow(11,13);
        e=59;
       break;
case 11: a=65539;
        e=31;
       break;
case 12: a=pow(13,3);
        e=31;
       break;
case 13: a=pow(5,13);
        e=39;
       break;
case 14: a=pow(5,17);
        e=42;
       break;
case 15: a=pow(11,13);
        e=59;
       break;
case 16: a=65539;
        e=31;
       break;
case 17: a=pow(13,3);
        e=31;
       break;
case 18: a=pow(5,13);
        e=39;
       break;
case 19: a=pow(5,17);
        e=42;
       break;
case 20: a=65539;
        e=31;
       break;
}
m = pow(2,e);
x = fmod(a*I[loc],m);
I[loc]=x;
return x/m;
}
//********************** calculation of exp ***************************
long double check_exp(const long double x){
double upper=700;
double lower=-740;
long double value,temp;
if(x>=lower && x <=upper) value=exp(x);
else{
if(x>upper) {
temp=x;
value=1;
while(temp>upper){
temp=temp-upper;
value=value*exp(upper);}
value=exp(temp)*value;
}
else if(x<lower) {
temp=x;
value=1;
while(temp<lower){
temp=temp-lower;
value=value*exp(lower);}
value=exp(temp)*value;
}
}
return value;
}
//********************** calculation of log ***************************
long double check_log(const long double x){
double upper=700;
double lower=1e-300;
long double value,temp;
if(x<0) {cout<<"\n ****** PROBLEMA  ****** Log de valor negativo"<<x;value = log(1);}
else if(x>=lower && x <=upper) value=log(x);
```

```
else{
if(x>upper) {
temp=x;
value=0;
while(temp>upper){
temp=temp/upper;
value += log(upper);}
value += log(temp);
}
if(x<lower) {
temp=x;
value=0;
while(temp<lower){
temp=temp/lower;
value+=log(lower);}
value+=log(temp);
}
}
return value;
}
//************************ calculation of gamma ************************
long double rgamma(const int alfa,const long double beta,const int tp){
long double g=0;
for(int i=0;i<alfa;i++) g+=log(un(15+(tp-1)*5));
return -(g/beta);
}
//************************ calculation of normal ************************
long double rnorm(const double A1,const double A2,const int tp){
double N;
float PI = 3.14159265358979323846;
static long double u1 =0.959678;
static long double u2 =0.87675545;
u1 =un(1+(tp-1)*5);
u2 =un(2+(tp-1)*5);
N = pow((-2*log(u1)),0.5)*cos(2*PI*u2);
return N*A2+A1;
}
//************************ calculation of factorial ************************
int factorial(const int n){
int n1=n;
if(n1<0) return 0;
int f=1;
while(n1>1)
f*=n1--;
return f;
}
//************************ calculation of numero de pulos ************************
void numero(int& k1){
k1=kmax+1;
while(k1>kmax){
long double u=un(5);
long double u1= exp(-lambda);
for(int i=0;i<=kmax;i++){
if(u<u1) {k1=i;break;}
else u1+=pow(lambda,i+1)*exp(-lambda)/factorial(i+1);
}
}
}
//************************ calculation do sort ****************
void sort(long double a[],const int n){
long double temp;
int j;
for(int i=1;i<n;i++){
temp=a[i];
for(j=i;j>0 && a[j-1]>temp;j--) a[j]=a[j-1];
a[j]=temp;
}
}
//************************ calculation do movimento ****************
int movimento(const int k2){
double d,n,b;
long double u1;
static int j=1;
j++;
if(j==5) j=1;
if(k2==0){
d=0;
if (lambda<=1) b=c*lambda;
else b=c;
n=1-b;}
```

```
else {
if(k2==kmax) b=0;
else {
if(lambda<=k2+1) b=c*lambda/(k2+1);
else b=c;
}
if (lambda<k2) d=c;
else d=c*k2/lambda;
n=1-b-d;
}
u1=un(1+(j-1)*5);
if(u1<b) return 1;
else{
if(u1<(d+b)) return 2;
else return 3;
}
}
//********************** calculation of sample ************************
int sample(int k1){
long double u=un(5);
for(int i=1;i<=k1;i++){
if(u*k1<=i) {return i;break;}
}
}
//********************** calculation of inicio ************************
int inicio(const double T[],const int n ,const long double valor){
int i,ini;
ini=n;
for(i=0;i<n;i++){if(T[i] > valor) { ini=i;break;}}
return ini;
}
//********************** calculation of fim ************************
int fim(const double T[],const int n,const long double valor){
int i,ini;
ini=n-1;
for(i=1;i<n;i++){if(T[i] >= valor) { ini=i-1;break;}}
return ini;
}
//********************** calculation of peso ************************
long double peso(const double T[],const double T1[],const long double s1,const long
double s2,const int n,const int n1,const int n2){
long double valor;
if(n1<=n2) valor=(T[n2]-T[n1]+(T[n1]-T[n1-1])*(T1[n1]-s1)/(T1[n1]-T1[n1-1])+ (T[n2+1]-
T[n2])*(s2-T1[n2])/(T1[n2+1]-T1[n2]));
else valor=0;
return valor;
}
//********************** calculation of increase ************************
void increase(long double T[],long double value,int w,int n){
int i;
for(i=(n-1);i>w;i--) T[i]=T[i-1];
T[w] = value;
}
//********************** calculation of decrease ************************
void decrease(long double T[],int w,int n){
int i;
if((n-1)>w) for(i=w;i<(n-1);i++) T[i] = T[i+1];
}
//********************** calculation of S (jumps) ********************
long double gS(const long double s1,const long double l1,const long double l2,const long
double s2,const double X[],const double W[],const int n,const int beg,const int
fin,const double wei,const double X1[],const int N[],const int n1,const int beg1,const
int fin1,const int tp){
int num=fin-beg+2;
int hp,i,j;
long double jump,u,sum,u1,x,l3,l4,w1,w2,wtot,cons,cons1,wtemp;
double x1,c1;
long double* C = new long double[num];
l3=l2-l1;
l4=check_log(l1)-check_log(l2);
w1=W[beg]*(X[beg]-s1)/(X[beg]-X[beg-1]);
w2=W[fin+1]*(s2-X[fin])/(X[fin+1]-X[fin]);
wtot=wei+w1+w2;
if(s2<=X[fin] || s1>=X[beg]) {cout<<"problema1",num=num-1;}
if(s2<=X[fin] && s1>=X[beg]) {cout<<"problema2",num=num-1;}
cons=0;
cons1=-l2*(wtot-w1);
c1=check_exp(-l1*w1)-check_exp(-l2*w1);
if(c1<0) c1=-c1;
if(w1>0 && l1!=l2) C[0]=cons1-check_log(w1/(X[beg]-s1))+check_log(c1);
```

```
else {C[0]=X[beg]-s1;cout<<"blabla";}
if(fin > beg) for(i=beg;i<fin;i++) {
if(fin1>=beg1) for(j=beg1;j<=fin1;j++){if(X[i]==X1[j]) cons+=N[j]*l4;}
if(i!=beg) wtemp=W[i];
else wtemp=w1;
cons1 = cons1+l2*W[i+1] -l1*wtemp;
c1=check_exp(-l1*W[i+1])-check_exp(-l2*W[i+1]);
if(c1<0) c1=-c1;
if(W[i+1]>0 && l1!=l2) C[i-beg+1]=cons+cons1-check_log(W[i+1]/(X[i+1]-
X[i]))+check_log(c1);
else {C[i-beg+1]=X[i+1]-X[i];cout<<"blablabla";}
}
if(fin1>=beg1)for(j=beg1;j<=fin1;j++){if(X[fin]==X1[j]) cons+= N[j]*l4;}
if(fin!=beg) wtemp=W[fin];
else wtemp=w1;
cons1 = cons1+l2*w2 - l1*wtemp;
c1=check_exp(-l1*w2)-check_exp(-l2*w2);
if(c1<0) c1=-c1;
if(w2>0 && l1!=l2) C[num-1]=cons+cons1-check_log(w2/(s2-X[fin]))+check_log(c1);
else {C[num-1]=s2-X[fin];cout<<"blablablabla";}
if(l1!=l2){
sum=0;
for(i=0;i<num;i++) {sum+=C[i];}
sum=sum/num;
cons=C[0]-sum;
for(i=0;i<num;i++){
C[i]=C[i]-sum;
if(cons<C[i]) cons=C[i];
}
if(cons>700) {for(i=0;i<num;i++){C[i]=C[i]-(cons-700);}}
for(i=0;i<num;i++){C[i]=check_exp(C[i]);}
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1;i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=50000;
while(hp==50000) {
u=un(15+(tp-1)*5);
if (u< C[0]) hp=0;
else {if (C[num-2]< u && u<1) {hp=num-1;}
     else {if(fin>beg) {for(i=1;i< num-1;i++) if(C[i-1]<u && u<C[i]){hp=i;break;}}}}
}
if(l1!=l2){
if(hp==0) {
wtemp=w1*(l2-l1);
x1=s1;
x=X[beg]-s1;
}
else if(s2>X[fin] && hp==num-1) {
wtemp=w2*(l2-l1);
x1=X[fin];
x=s2-X[fin];
}
else {
wtemp=W[beg+hp]*(l2-l1);
x1=X[beg+hp-1];
x=X[beg+hp]-X[beg+hp-1];
}
u=un(14+(tp-1)*5);
jump=x1+x*check_log(u*check_exp(wtemp)-u+1)/wtemp;
}
else jump=x1+u*x;
delete []C;
return jump;
}
//********************** calculation of S2 (jumps) para severidade*******************
long double gS2(const long double s1,const long double l1,const long double l2,const
long double s2,const double X[],const double H[],const int N[],const int size,const int
beg,const int fin,const int sig,const int tp){
int num=fin-beg+2;
int hp,i,j;
long double jump,u,sum,u1,x,l3,l4,w1,w2,wtot,cons,cons1,wtemp;
double x1,c1;
long double* C = new long double[num];
l3=2*(l2-l1);
l4=pow(l1,2)-pow(l2,2);
C[0]=0;
c1=X[beg]-s1;
if(fin > beg) for(i=beg;i<fin;i++) {
```

```
C[i-beg+1]=C[i-beg]+log(X[i+1]-X[i])-check_log(c1)- (N[i]*l4+H[i]*l3)/(2*pow(sig,2));
c1=X[i+1]-X[i];
}
if(s2>X[fin]) C[num-1]=C[fin-beg]+check_log(s2-X[fin])-check_log(c1)-
(N[fin]*l4+H[fin]*l3)/(2*pow(sig,2));
else num=num-1;
sum=0;
for(i=0;i<num;i++)sum+=C[i];
sum=sum/num;
cons=C[0]-sum;
for(i=0;i<num;i++){
C[i]=(C[i]-sum);
if(cons<C[i]) cons=C[i];
}
if(cons>700) {for(i=0;i<num;i++){C[i]=C[i]-(cons-700);}}
for(i=0;i<num;i++){C[i]=check_exp(C[i]);}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1;i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=50000;
while(hp==50000) {
u=un(5+(tp-1)*5);
if (u< C[0]) hp=0;
else {if (C[num-2]< u && u<1) {hp=num-1;}
     else {if(fin>beg) {for(i=1;i< num-1;i++) if(C[i-1]<u && u<C[i]){hp=i;break;}}}}
}
if(hp==0) {x1=s1;x=X[beg]-s1;}
else if(hp==num-1 && s2>X[fin]) {x1=X[fin];x=s2-X[fin];}
else {x1=X[beg+hp-1];x=X[beg+hp]-X[beg+hp-1];}
u=un(4+(tp-1)*5);
jump=x1+u*x;
delete []C;
return jump;
}
//********************** calculation of aceitacao ************************
long double aceitacao(const double int1,const double int2,const int N1,const int
Nj,const long double L1,const long double Lj,const int k1,const long double h1,const
long double h2,const long double h3,const int A,const int B){
long double lr,pr,jac;
lr=N1*check_log(h1)+(Nj-N1)*check_log(h3)-Nj*check_log(h2)+h2*Lj-h1*L1-h3*(Lj-L1);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1 * factorial(A-1));
pr = check_log(pr) + A*check_log(B)- B * (h1 + h3 - h2) + (A-1)*check_log(h1*h3/h2);
jac = 2*check_log(h1 + h3)-check_log(h2);
return lr+pr+jac;
}
//********************** calculation of aceipeso ************************
long double aceipeso(const double int1,const double int2,const int N1,const int Nj,const
long double L1,const long double Lj,const int k1,const long double h1,const long double
h2,const long double h3,const int A,const int B){
long double lr,pr,jac;
lr=N1*check_log(h1)+(Nj-N1)*check_log(h3)-Nj*check_log(h2)+h2*Lj-h1*L1-h3*(Lj-L1);
pr = 2 * (2 * k1 + 3) * L1 * (Lj-L1) /(Smax * Lj * factorial(A-1));
pr = check_log(pr)+ A*check_log(B) - B * (h1 + h3 - h2) + (A-1)*check_log(h1*h3/h2);
jac = 2*check_log(h1 + h3)-check_log(h2);
return lr+pr+jac;
}
//******************* calculation of aceitacao para severidade ************************
long double aceitacao2(const double int1,const double int2,const int N1,const int
Nj,const long double L1,const long double Lj,const int k1,const long double h1,const
long double h2,const long double h3,const int sig,const int A,const int B){
long double lr,pr,jac;
long double temp,temp1;
float PI = 3.14159265358979323846;
lr = (2*h1*L1-N1*pow(h1,2)+2*h3*(Lj-L1)-(Nj-N1)*pow(h3,2)-
2*h2*Lj+Nj*pow(h2,2))/(2*sig*sig);
jac=2*PI;
jac=pow(jac,0.5);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1*sig*jac);
pr=check_log(pr)-(pow(A,2)-2*A*(h1+h2-h3)+pow(h1,2)+pow(h3,2)-pow(h2,2))/(2*pow(B,2));
if((h1+h3)<0) temp=-(h1+h3);
else temp=h1+h3;
if(h2<0) temp1=-h2;
else temp1=h2;
jac=-100;
if(temp!=0 && temp1!=0) jac= 2*check_log(temp)-check_log(temp1);
return lr+pr+jac;
}
//********************** calculation of frequencia ************************
```

```
void frequencia(long double S[],const int ns,long double L[],const int nl,int& k,const
double X[],const double W[],const double WT[],const int size,const double X1[],const int
N[],const int NT[],const int size1,const int A,const int B,int TOTA[],const int na,int
TOT[],const int n,int tipo_freq){
int i,j,beg1,beg,fin1,fin,num,u2,pos,mov,num1;
long double u,u1,a,s1,ac,cons,h1,h2,h3;
double wei,wei1;
static int tipo=1;
tipo++;
if(tipo==3) tipo=1;
beg1=inicio(X1,size1,S[0]);
fin1=fim(X1,size1,S[1]);
if (fin1>=beg1) num=NT[fin1]-NT[beg1-1];
else num=0;
beg=inicio(X,size,S[0]);
fin=fim(X,size,S[1]);
wei=peso(WT,X,S[0],S[1],size,beg,fin);
L[0]=rgamma(num+A,wei+B,tipo);
if(k>0) for(j=1;j <= k;j++){
fin=fim(X,size,S[j+1]);
fin1=fim(X1,size1,S[j+1]);
if(L[j-1]!=L[j]){
if(beg<=fin) {
S[j]=gS(S[j-1],L[j-1],L[j],S[j+1],X,W,size,beg,fin,(WT[fin]-WT[beg-
1]),X1,N,size1,beg1,fin1,tipo);
}
else {
u=un(15+(tipo-1)*5);
a= (L[j]-L[j-1])*W[fin+1];
S[j]=S[j-1]+check_log(u*check_exp(a)+1-u)*(S[j+1]-S[j-1])/a;}}
if(S[j]>S[j+1]) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
if(S[j]<S[j-1]) cout<<"erro2";
beg1=inicio(X1,size1,S[j]);
beg=inicio(X,size,S[j]);
if (fin1>=beg1) num=NT[fin1]-NT[beg1-1];
else num=0;
wei=peso(WT,X,S[j],S[j+1],size,beg,fin);
L[j]=rgamma(num+A,wei+B,tipo);
}
mov=movimento(k);
u2=0;
if(mov==1){
        s1 =un(14+(tipo-1)*5)*Smax;
        pos=0;
        while(u2 < k) {
          u2 = 0;
          s1=un(14+(tipo-1)*5)*Smax;
          pos = 0;
          for(i=1;i<=k;i++) {
            if(S[i] != s1) u2++;
            if(S[i] < s1) pos = i;
          }
        }
        beg=inicio(X,size,S[pos]);
        fin=fim(X,size,S[pos+1]);
        wei=peso(WT,X,S[pos],S[pos+1],size,beg,fin);
        fin=fim(X,size,s1);
        wei1=peso(WT,X,S[pos],s1,size,beg,fin);
        beg1=inicio(X1,size1,S[pos]);
        fin1=fim(X1,size1,S[pos+1]);
        if (fin1>=beg1) num=NT[fin1]-NT[beg1-1];
        else num=0;
        fin1=fim(X1,size1,s1);
        if (fin1>=beg1) num1=NT[fin1]-NT[beg1-1];
        else num1=0;
        if(wei1!=0 && wei1!=wei)
        {
        h2 = L[pos];
        cons = (1/un(11+(tipo-1)*5) - 1);
       if(tipo_freq==1 || tipo_freq==2) h1 = h2 * pow(ccns,((wei - wei1)/wei));
        else h1 = h2 * pow(cons,((s1-S[pos+1])/(S[pos+1]-S[pos])));

        h3 = cons * h1;
        if(tipo_freq==2 ||tipo_freq==4) ac=aceitacao(S[pos+1]-S[pos],s1-
S[pos],num1,num,wei1,wei,k,h1,h2,h3,A,B);
        else ac=aceipeso(S[pos+1]-S[pos],s1-S[pos],num1,num,wei1,wei,k,h1,h2,h3,A,B);
        if(ac>=0) ac=1;
        else ac=check_exp(ac);
        u1=un(14+(tipo-1)*5);
        if(u1 <= ac) {
```

```
          increase(S,sl,pos + 1,kmax +2);
          L[pos] = h1;
          increase(L,h3,pos + 1,kmax+1);
          k = k + 1;
          TOTA[mov-1]++ ;
}
}
}
if(mov==2){
        pos = sample(k);
        sl =S[pos];
        beg=inicio(X,size,S[pos-1]);
        fin=fim(X,size,S[pos+1]);
        wei=peso(WT,X,S[pos-1],S[pos+1],size,beg,fin);
        fin=fim(X,size,S[pos]);
        wei1=peso(WT,X,S[pos-1],sl,size,beg,fin);
        beg1=inicio(X1,size1,S[pos-1]);
        fin1=fim(X1,size1,S[pos+1]);
        if (fin1>=beg1) num=NT[fin1]-NT[beg1-1];
        else num=0;
        fin1=fim(X1,size1,sl);
        if (fin1>=beg1) num1=NT[fin1]-NT[beg1-1];
        else num1=0;
        if(wei1!=0 && wei1!=wei)
        {
        h1 = L[pos-1];
        h3 = L[pos];
        if(tipo_freq==1 || tipo_freq==2) h2 = check_exp((wei1 * check_log(h1) + (wei -
wei1) * check_log(h3))/wei);
        else h2 = check_exp(((S[pos]-S[pos-1]) * check_log(h1) + (S[pos+1]-S[pos]) *
check_log(h3))/(S[pos+1]-S[pos-1]));
        if(tipo_freq==2 ||tipo_freq==4) ac=aceitacao(S[pos+1]-S[pos-1],S[pos]-S[pos-
1],num1,num,wei1,wei,k,h1,h2,h3,A,B);
        else ac=aceipeso(S[pos+1]-S[pos-1],S[pos]-S[pos-1],num1,num,wei1,wei,k,h1,
h2,h3,A,B);
        if(ac<=0) ac=1;
        else ac = check_exp(-ac);
        u1 = un(15+(tipo-1)*5);
        if(u1 <= ac) {
          k = k - 1;
          decrease(S,pos,kmax +2);
          L[pos-1] = h2;
          decrease(L,pos + 1,kmax+1);
          TOTA[mov-1]++;
        }
        }
}
 TOT[mov-1]++;
}
//********************** calculation of severidade ************************
void severidade(long double S[],const int ns,long double L[],const int nl,int& k,const
double H[],const double HT[],const double X[],const int N[],const int NT[],const int
size,const int A,const int B,const int sig,int TOTA[],const int na,int TOT[],const int
n){
int i,j,beg,fin,num,u2,pos,mov,num1;
long double u,u1,a,sl,ac,cons,h1,h2,h3;
double wei,wei1;
static int tipo=1;

tipo++;
if(tipo==3) tipo=1;
beg=inicio(X,size,S[0]);
fin=fim(X,size,S[1]);
if (fin>=beg) {num=NT[fin]-NT[beg-1];wei=HT[fin]-HT[beg-1];}
else {num=0;wei=0;}
a=num/pow(sig,2)+1/pow(B,2);
cons=wei/pow(sig,2)+A/pow(B,2);
L[0]=rnorm((cons/a),pow(a,-0.5),tipo);
if(k>0) for(j=1;j <= k;j++){
fin=fim(X,size,S[j+1]);
if(L[j-1]!=L[j]){
if(beg<=fin) S[j]=gS2(S[j-1],L[j-1],L[j],S[j+1],X,H,N,size,beg,fin,sig,tipo);
else {
u=un(5+(tipo-1)*5);
S[j]=S[j-1]+u*(S[j+1]-S[j-1]);
}
}
if(S[j]>S[j+1]) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
if(S[j]<S[j-1]) cout<<"erro2";
beg=inicio(X,size,S[j]);
```

```
if (fin>=beg) {num=NT[fin]-NT[beg-1];wei=HT[fin]-HT[beg-1];}
else {num=0;wei=0;}
a=num/pow(sig,2)+1/pow(B,2);
cons=wei/pow(sig,2)+A/pow(B,2);
L[j]=rnorm(cons/a,pow(a,-0.5),tipo);
}
mov=movimento(k);
u2=0;
if(mov==1){
        sl =un(4+(tipo-1)*5)*Smax;
        pos=0;
        while(u2 < k) {
          u2 = 0;
          sl=un(4+(tipo-1)*5)*Smax;
          pos = 0;
          for(i=1;i<=k;i++) {
            if(S[i] != sl) u2++;
            if(S[i] < sl) pos = i;
          }
        }
        beg=inicio(X,size,S[pos]);
        fin=fim(X,size,S[pos+1]);
        if (fin>=beg) {num=NT[fin]-NT[beg-1];wei=HT[fin]-HT[beg-1];}
        else {num=0;wei=0;}
        fin=fim(X,size,sl);
        if (fin>=beg) {num1=NT[fin]-NT[beg-1];wei1=HT[fin]-HT[beg-1];}
        else {num1=0;wei1=0;}
        h2 = L[pos];
        if(h2!=0){
        cons = (1/un(1+(tipo-1)*5) - 1);
        h1 = (S[pos+1]-S[pos])*h2/((S[pos+1]-sl)/cons+sl-S[pos]);
        h3 = h1/cons;
        ac=aceitacao2(S[pos+1]-S[pos],sl-S[pos],num1,num,wei1,wei,k,h1,h2,h3,sig,A,B);
        if(ac>=0) ac=1;
        else ac=check_exp(ac);
        u1=un(4+(tipo-1)*5);
        if(u1 <= ac) {
          increase(S,sl,pos + 1,kmax +2);
          L[pos] = h1;
          increase(L,h3,pos + 1,kmax+1);
          k = k + 1;
          TOTA[mov-1]++ ;
}
}
}
if(mov==2){
        pos = sample(k);
        sl =S[pos];
        beg=inicio(X,size,S[pos]);
        fin=fim(X,size,S[pos+1]);
        if (fin>=beg) {num=NT[fin]-NT[beg-1];wei=HT[fin]-HT[beg-1];}
        else {num=0;wei=0;}
        fin=fim(X,size,sl);
        if (fin>=beg) {num1=NT[fin]-NT[beg-1];wei1=HT[fin]-HT[beg-1];}
        else {num1=0;wei1=0;}
        h1 = L[pos-1];
        h3 = L[pos];
        h2 = ((S[pos]-S[pos-1]) * h1 + (S[pos+1]-S[pos]) * h3)/(S[pos+1]-S[pos-1]);
        if(h2!=0){
        ac=aceitacao2(S[pos+1]-S[pos-1],S[pos]-S[pos-1],num1,num,wei1,wei,k,h1,h2,
h3,sig,A,B);
        if(ac<=0) ac=1;
        else ac = check_exp(-ac);
        u1 = un(5+(tipo-1)*5);
        if(u1 <= ac) {
          k = k - 1;
          decrease(S,pos,kmax +2);
          L[pos-1] = h2;
          decrease(L,pos + 1,kmax+1);
          TOTA[mov-1]++;
        }
}
}
 TOT[mov-1]++;
}
```

Appendix B

B.1 Main Program

```
//      This program is to implement the calculation of the number of jumps
//      and related hazard rate for the frequency and severity of claims
//      using the reversible jump technique for 3 different set of intensity
//       related to age&sex,calendar time and report delay. We are also
//       calculating the IBNR claims,defining their moment of occurrence and all
//       other information.

#include "library/den.h"
//************************program beginning

main(){
long double g1,f1,l1,a,b,u,tot;
double w,s1,s2,Res_RBNS,Res_IBNR;
int  i,j,z,z1,z2,temp,IBNR,temp1,temp2,temp3,temp4,contador,g1_index,g1_itemp,tam;
int kf[6]={0,0,0,0,0,0};
//declaracao dos dynamic arrays
int* N1;
double* ST;
long double* uniforme;
double *data_IBNR_time;
int* N;
long double *Prob[2*Amax];
//working arrays
int data_tot_number[2][Amax];
double data_tot_v[2][Amax];
long double acumu[2][Amax];
int IND[2][2*kmax+7];
double T_S[2*kmax+7];

for(j=0;j<Amax;j++) for(z=0;z<2;z++) acumu[z][j]=0;
for(i=0;i<(2*kmax+7);i++) T_S[i]=0;
for(j=0;j<2;j++) for(i=0;i<(2*kmax+7);i++) IND[j][i]=0;
double W[6][Amax][2];
for(i=0;i<6;i++) for(j=0;j<Amax;j++) for(z=0;z<2;z++) W[i][j][z]=0;

double data_t_time[tam_t];
int data_t_number[tam_t];
double data_u_time[tam_u];
int data_u_number[tam_u];
int data_a_number[2][Amax];
double data_value[2][Amax];
int number_RBNS[2][Amax];

ifstream f_t("/usr/home/fc/den/severity/dados/rep_t");
 if(!f_t) {cout << "Cannot open file for input\n";  exit(1);}
f_t.seekg(0);

ifstream f_u("/usr/home/fc/den/severity/dados/rep_u");
 if(!f_u) {cout << "Cannot open file for input\n";  exit(1);}
f_u.seekg(0);

ifstream f_a("/usr/home/fc/den/severity/dados/rep_a");
 if(!f_a) {cout << "Cannot open file for input\n";  exit(1);}
f_a.seekg(0);

ifstream fexp("/usr/home/fc/den/severity/dados/exp.txt");
 if(!fexp) {cout << "Cannot open file for input\n";  exit(1);}
fexp.seekg(0);

ifstream fvalue("/usr/home/fc/den/severity/dados/paid_log.txt");
//ifstream fvalue("/usr/home/fc/den/severity/dados/paid_nonlog.txt");
 if(!fvalue) {cout << "Cannot open file for input\n";  exit(1);}
fvalue.seekg(0);
```

```
ifstream frbns("/usr/home/fc/den/severity/dados/rbns_num.txt");
 if(!frbns) {cout << "Cannot open file for input\n";  exit(1);}
frbns.seekg(0);

ifstream frsev("/usr/home/fc/den/severity/dados/rep_a_sev.txt");
 if(!frsev) {cout << "Cannot open file for input\n";  exit(1);}
frsev.seekg(0);

ofstream sail;
ofstream fibnr;

for(i=0;i<tam_t; i++){ f_t >> data_t_time[i] >>data_t_number[i];}

for(i=0;i<tam_u; i++){ f_u >> data_u_time[i] >>data_u_number[i];}

for(i=0;i<(2*Amax); i++){
 f_a >> temp1 >>temp2>>temp3;
 data_a_number[temp1][temp2]=temp3;
}
for(i=0; i<Exp; i++){
 fexp >> temp1 >>temp2>>temp3>> w;
 W[temp1-1][temp3][temp2]=w/365;
}
for(i=0;i<(2*Amax); i++){
 fvalue >> temp1 >>temp2>>s2;
 data_value[temp1][temp2]=double(s2);
}
for(i=0;i<(2*Amax-2); i++){
 frbns >> temp1 >>temp2>>temp3;
 number_RBNS[temp1][temp2]=temp3;
}
for(i=0;i<(2*Amax-2); i++){
 frsev >> temp1 >>temp2>>temp3;
 data_a_number1[temp1][temp2]=temp3;
}
number_RBNS[0][0]=0;
number_RBNS[1][0]=0;

fexp.close();
f_t.close();
f_u.close();
f_a.close();
frbns.close();
fvalue.close();

//defining accumulation of acceptation rate
int T[6][4]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
//parameter arrays
long double L[4][kmax+1];              // history of levels
double S_t[kmax+2];            // history of jumps
double S_u[kmax+2];            // history of jumps
int S_a[2][kmax+2];                    // history of jumps

long double Mi[2][kmax+1];             // history of levels
int S_val[2][kmax+2];                          // history of jumps

kf[4]=5;
kf[5]=5;
S_t[0]=0;
S_u[0]=0;
S_a[0][0]=0;
S_a[1][0]=0;
S_val[0][0]=0;
S_val[1][0]=0;

for(i=0;i<2;i++){
Mi[i][0]=rnorm(A[i+4],B[i+4],i+3);
}
for(i=0;i<2;i++){
for(j=1;j<=kf[i+4];j++){
Mi[i][j]=rnorm(A[i+4],B[i+4],i+3);
S_val[i][j]=j*72/(kf[i+4]+1);
}}

kf[0]=0;
S_t[1]=Smax;
L[0][0]=1;

kf[1]=2;
S_a[0][0]= 0;
```

```
L[2][0]= 3.58985e-05;
S_a[0][1]= 6;
L[2][1]= 1.9966e-05;
S_a[0][2]= 36;
L[2][2]= 9.60227e-06;
S_a[0][3]= 67;

kf[3]=2;
S_a[1][0]= 0;
L[3][0]= 1.47824e-05;
S_a[1][1]= 15;
L[3][1]= 2.464e-05;
S_a[1][2]= 32;
L[3][2]= 1.3668e-05;
S_a[1][3]= 67;

kf[1]=5;
S_u[0]= 0;
L[1][0]= 0.00131357;
S_u[1]= 4.95386;
L[1][1]= 0.0148287;
S_u[2]= 7.99944;
L[1][2]= 0.0745206;
S_u[3]= 18.5306;
L[1][3]= 0.0489292;
S_u[4]= 36.2348;
L[1][4]= 0.0242072;
S_u[5]= 78.7191;
L[1][5]= 0.00745073;
S_u[6]= 2190;

for (contador=0; contador<ini+ite; contador++){
T_S[0]=0; IND[0][0]=0; IND[1][0]=kf[1];
tam=1;i=0;j=0;z=kf[1];
for(z1=1;;z1++){
 tam=z1;
 s1=S_t[j+1];
 if(s1 > (Smax-S_u[z])) s1 = Smax-S_u[z];
 if(s1 > W1[i+1]) s1 = W1[i+1];
 if(s1 == (Smax-S_u[z])) if(z!=0) z--; else break;
 if(s1 == W1[i+1]) if(i!=6) i++; else break;
 if(s1 == S_t[j+1]) if(j!=kf[0]) j++; else break;
 T_S[z1]=s1;
 IND[0][z1]=j;
 IND[1][z1]=z;
}
T_S[tam]=Smax;
for(i=0;i<(2*Amax);i++){Prob[i]=new long double[tam];for(j=0;j<tam;j++){Prob[i][j]=0;}}
tot=0;
a=0;
for(i=0;i<=kf[1];i++){a-=L[1][i]*(S_u[i+1]-S_u[i]);}
for(j=0;j<tam;j++){
temp = transf(T_S[j]);
 z1=IND[1][j];
 g1 = L[1][z1];
 if(j>0) {z2=IND[1][j-1];if(z1!=z2) a+=(L[1][z2]-L[1][z1])*(Smax-S_u[z2]);}
 z1=IND[0][j];
 f1 =L[0][z1];
for(i=0;i<(Amax*2);i++){
   z=0;
   if(i>=Amax) z=1;
   l1=qual_a(L[z+2],S_a[z],kf[z+2],i-z*Amax);
   if(temp>0 && W[temp-1][i-z*Amax][z]!=W[temp][i-z*Amax][z]){
    b=check_log(W[temp][i-z*Amax][z]/W[temp-1][i-z*Amax][z])/365;
    Prob[i][j]=f1*l1*W[temp-1][i-z*Amax][z]*check_exp(a-
b*W1[temp]+(b+g1)*T_S[j])*(check_exp((b+g1)*(T_S[j+1]-T_S[j]))-1)/(b+g1);
   }
   else{Prob[i][j]=f1*l1*W[temp][i-z*Amax][z]*check_exp(a+g1*T_S[j+1])*(1-
check_exp(g1*(T_S[j]-T_S[j+1])))/g1;}
  if(Prob[i][j]<0) cout<<"Probability smaller than zero!!!";
  tot+=Prob[i][j];
  Prob[i][j]=tot;
}
}
a=tot;
b=pow(tot,0.5);
if(tot>50) u=rnorm(a,b,1);
else u=dpoi(a);
IBNR = int(u);
```

```
uniforme= new long double [IBNR];
for(i=0;i<IBNR;i++) {uniforme[i]=un(1)*tot;}
sort(uniforme,IBNR);
data_IBNR_time=new double[IBNR];
for(j=0;j<Amax;j++) for(z=0;z<2;z++) data_tot_number[z][j]=0;
for(i=0;i<Amax;i++)for(z=0;z<2;z++) data_tot_number[z][i]=data_a_number[z][i];
i=0;
j=0;
for(z1=0;z1<IBNR;){
  temp = transf(T_S[j]);
  z=0;
  if(i>=Amax) z=1;
  if(uniforme[z1]<=Prob[i][j]){
    data_tot_number[z][i-z*Amax]++;
if(temp>0 && W[temp][i-z*Amax][z]!=W[temp-1][i-z*Amax][z]){
      b=check_log(W[temp][i-z*Amax][z]/W[temp-1][i-z*Amax][z])/365;
      u=un(2);
      u=0.5 +100*check_log((1-u)*check_exp(b*(T_S[j+1]-T_S[j]))+u)/b;
      temp1=int(u);
      data_IBNR_time[z1]=T_S[j]+double(temp1/100);
       }
    else {
      u=0.5+100*un(2)*(T_S[j+1]-T_S[j]);
      temp1=int(u);
      data_IBNR_time[z1]=T_S[j]+double(temp1/100);
      }
  if(data_IBNR_time[z1]>T_S[j+1]) data_IBNR_time[z1]=T_S[j+1];
  z1++;
  }
  else{
    i++;
    if(i==(2*Amax)){i=0; j++;}
    if(j==tam) break;
  }
}
delete[] uniforme;
delete[] Prob;
sort_d(data_IBNR_time,IBNR);
//**************************updating all parameters
//*****************************delay
for (i=0;i<IBNR;i++){data_IBNR_time[i]= Smax-data_IBNR_time[i];}
tam=IBNR+tam_u+2;
ST = new double[tam];
for(i=0;i<tam_u;i++){ST[i]=data_u_time[i];}
if(IBNR>0) for(i=0;i<IBNR;i++){ST[i+tam_u]=data_IBNR_time[i];}
ST[IBNR+tam_u]=0;
ST[IBNR+tam_u+1]=Smax;
sort_d(ST,tam);
temp=tam;
for(i=1;i<temp;i++){
  if(i<tam) {
   if(ST[i-1]==ST[i]) {
    decrease_d(ST,i,temp);
    i--;
    tam=tam-1;
   }
  }
  else break;
}
N = new int[tam];
N1 = new int[tam];
for(i=0;i<tam;i++){N[i]=0;N1[i]=0;}
sort_d(data_IBNR_time,IBNR);
N[0]=data_u_number[0];
N1[0]=N[0]+IBNR;
z=1;
j=0;
for(i=1;i<tam;i++){
  N[i]=N[i-1];
  N1[i]=N1[i-1];
  if(ST[i]==data_u_time[z]){N[i]=data_u_number[z];z++;N1[i]+=(N[i]-N[i-1]);}
  if(ST[i]==data_IBNR_time[j]){
    N1[i]--;
for(temp=0;temp<IBNR;temp++) {
      j++;
      if(data_IBNR_time[j-1]==data_IBNR_time[j]) N1[i]--; else break;
    }
  }
}
fr_gu(S_u,L[1],kf[1],ST,N,N1,tam,A[1],B[1],T[1]);
```

```
delete []N;
delete []N1;
delete []ST;
delete []data_IBNR_time;
//**************************age and sex
for(j=0;j<Amax;j++) for(z=0;z<2;z++) data_tot_v[z][j]=0;
for(i=0;i<=kf[0];i++){
s1=S_t[i];
  s2=S_t[i+1];
  if(s2<=W1[1]) {for(j=0;j<Amax;j++) {for(z=0;z<2;z++) {
  data_tot_v[z][j]=L[0][i]*W[0][j][z]*(s2-s1);}}}
  else
  {
   if(s1<W1[1]) {s1=W1[1];temp=1;for(j=0;j<Amax;j++) {for(z=0;z<2;z++)
{data_tot_v[z][j]=L[0][i]*W[0][j][z]*(s1-S_t[i]);}}}
   temp1=int(s2/365);
   if(temp1=6) temp1--;
   temp=int(s1/365);
   for(z1=temp;z1<=temp1;z1++){
     s2=S_t[i+1];
     if(s2>W1[z1+1])s2=W1[z1+1];
     for(j=0;j<Amax;j++)  for(z=0;z<2;z++) {
       b=check_log(W[z1][j][z]/W[z1-1][j][z])/365;
       if(b!=0) data_tot_v[z][j]+=L[0][i]*W[z1-1][j][z]*check_exp(b*(s1-
W1[z1]))*(check_exp(b*(s2-s1))-1)/b;
       else data_tot_v[z][j]+=L[0][i]*W[z1][j][z]*(s2-s1);
      }
    s1=s2;
    }
  }
}
fr_sa(S_a[0],L[2],kf[2],data_tot_number[0],data_tot_v[0],A[2],B[2],T[2]);
fr_sa(S_a[1],L[3],kf[3],data_tot_number[1],data_tot_v[1],A[3],B[3],T[3]);
Res_IBNR=0;
for(j=0;j<Amax;j++) for(z=0;z<2;z++)
{data_tot_v[z][j]=data_value[z][j];Res_IBNR+=data_value[z][j];}
Res_IBNR=0;
Res_RBNS=0;
temp1=0;
temp2=0;
temp3=0;
for(j=0;j<Amax;j++){
  for(z=0;z<2;z++){
    l1=qual_a(Mi[z],S_val[z],kf[z+4],j);
    a=rnorm(l1,val_sig,1);
    temp=data_tot_number[z][j]-data_a_number1[z][j];
    if(number_RBNS[z][j]>0){
      data_tot_v[z][j]+=a*number_RBNS[z][j];
      Res_RBNS+=number_RBNS[z][j]*exp(a);
}
    if(temp>0){
      data_tot_v[z][j]+=a*temp;
      Res_IBNR+=temp*exp(a);
}
}
}
fibnr.open("/usr/home/fc/den/severity/results/new/high/IBNR_log1",ios::app);
fibnr<<tot<<" "<<IBNR<<" "<<Res_RBNS<<" "<<Res_IBNR<<"\n";
fibnr.close();
severidade_sa(S_val[0],Mi[0],kf[4],data_tot_number[0],data_tot_v[0],A[4],B[4],val_sig,T[4
]);
severidade_sa(S_val[1],Mi[1],kf[5],data_tot_number[1],data_tot_v[1],A[5],B[5],val_sig,T[5
]);
sail.open("/usr/home/fc/den/severity/results/new/high/male_log1",ios::app);
sail<<"male"<<" 1 "<<kf[4]<<" ";
for (i=0;i<=kf[4];i++) sail<<S_val[0][i]<<" ";
sail<<"\n"<<"male"<<" 2 ";
for (i=0;i<=kf[4];i++) sail<<Mi[0][i]<<" ";
sail<<"\n";
sail<<"female"<<" 1 "<<kf[5]<<" ";
for (i=0;i<=kf[5];i++) sail<<S_val[1][i]<<" ";
sail<<"\n"<<"female"<<" 2 ";
for (i=0;i<=kf[5];i++) sail<<Mi[1][i]<<" ";
sail<<"\n";
sail.close();

}//end for
sail.open("/usr/home/fc/den/severity/results/new/high/result_log1",ios::app);
sail<<"calendar mov 1 "<<T[0][0]<<" "<<T[0][2]<<"\n";
sail<<"calendar mov 2 "<<T[0][1]<<" "<<T[0][3]<<"\n";
```

```
sail<<"delay mov 1 "<<T[1][0]<<" "<<T[1][2]<<"\n";
sail<<"delay mov 2 "<<T[1][1]<<" "<<T[1][3]<<"\n";
sail<<"Age_mas mov 1 "<<T[2][0]<<" "<<T[2][2]<<"\n";
sail<<"Age_mas mov 2 "<<T[2][1]<<" "<<T[2][3]<<"\n";
sail<<"Age_fem mov 1 "<<T[3][0]<<" "<<T[3][2]<<"\n";
sail<<"Age_fem mov 2 "<<T[3][1]<<" "<<T[3][3]<<"\n";
sail<<"Val_mas mov 1 "<<T[4][0]<<" "<<T[4][2]<<"\n";
sail<<"Val_mas mov 2 "<<T[4][1]<<" "<<T[4][3]<<"\n";
sail<<"Val_fem mov 1 "<<T[5][0]<<" "<<T[5][2]<<"\n";
sail<<"Val_fem mov 2 "<<T[5][1]<<" "<<T[5][3]<<"\n";
sail.close();

}//end program
```

B.2 Library

```
#include <fstream.h>
#include <math.h>
#include <stdlib.h>
#include <iostream.h>
// general constants
const double Smax = 2190;
const int Amax = 67;
const int ini  = 1;
const int ite  =5000;
//size of arrays
const int tam_u = 197;
const int tam_t = 1488;
const int Exp = 804;
const int W1[7]={0,365,730,1095,1460,1825,2190};
int divisor_age=1000000;
int divisor_g=100;
const double val_sig=2;
const double Val_RENS = 824146;
//************prior values*************************
const int lambda = 3;
const int A[6]={2,2,2,2,8,8};
const double B[6]={0.2527,0.0421,0.1266,0.1266,1.02,1.02};
const int kmax = 30;
const float c = 0.4;
//******************* calculation of Uniform(0,1) *************************
long double un(const int loc) {
long double a,e,x,m;
static long double
I[21]={1561,1448,9879,67961,791,989,989,34274,3568,4679,7869,57807,57901,589,22457,267837
,37909,37679,37577,3757,3579};
switch(loc){
case 1: a=65539;
   e=31;
  break;
case 2: a=pow(13,3);
   e=31;
  break;
case 3: a=pow(5,13);
   e=39;
  break;
case 4: a=pow(5,17);
   e=42;
  break;
case 5: a=pow(11,13);
   e=31;
  break;
case 6: a=pow(11,13);
   e=47;
  break;
case 7: a=pow(13,3);
   e=31;
  break;
case 8: a=pow(5,13);
   e=39;
  break;
case 9: a=pow(5,17);
   e=42;
  break;
case 10: a=pow(11,13);
   e=59;
```

```
  break;
case 11: a=65539;
   e=31;
  break;
case 12: a=pow(13,3);
   e=31;
  break;
case 13: a=pow(5,13);
   e=39;
  break;
case 14: a=pow(5,17);
   e=42;
  break;
case 15: a=pow(11,13);
   e=59;
  break;
case 16: a=65539;
   e=31;
  break;
case 17: a=pow(13,3);
   e=31;
  break;
case 18: a=pow(5,13);
   e=39;
  break;
case 19: a=pow(5,17);
   e=42;
  break;
case 20: a=65539;
   e=31;
  break;
}
m = pow(2,e);
x = fmod(a*I[loc],m);
I[loc]=x;
return x/m;
}
//********************** calculation of transf ***************************
int transf(long double S){
int temp;
if(S<W1[1])temp=0;
else if(S<W1[2])temp=1;
else if(S<W1[3])temp=2;
else if(S<W1[4])temp=3;
else if(S<W1[5])temp=4;
else temp=5;
return temp;
}
//************************ calculation of exp ***************************
long double check_exp(const long double x){
double upper=700;
double lower=-740;
long double value,temp;
if(x>=lower && x <=upper) value=exp(x);
else{
  if(x>upper) {
    temp=x;
    value=1;
    while(temp>upper){
      temp=temp-upper;
      value=value*exp(upper);}
    value=exp(temp)*value;
  }
  else if(x<lower) {
    temp=x;
    value=1;
    while(temp<lower){
      temp=temp-lower;
      value=value*exp(lower);}
    value=exp(temp)*value;
  }
}
return value;
}
//************************ calculation of log ***************************
long double check_log(const long double x) {
double upper=700;
double lower=1e-300;
long double value,temp;
if(x<0) {cout<<"\n ****** PROBLEMA  ****** Log de valor negativo"<<x; value = log(1);}
```

```
else if(x>=lower && x <=upper) value=log(x);
  else{
    if(x>upper) {
      temp=x;
      value=0;
      while(temp>upper){
  temp=temp/upper;
  value += log(upper);}
      value += log(temp);
    }
    if(x<lower) {
      temp=x;
      value=0;
      while(temp<lower){
  temp=temp/lower;
  value+=log(lower);}
      value+=log(temp);
    }
  }
return value;
}
//********************** calculation of gamma ************************
long double rgamma(const int alfa,const double beta,const int tp) {
long double g=0;
for(int i=0;i<alfa;i++) g+=log(un(15+(tp-1)*5));
return -(g/beta);
}
//********************** calculation of normal ************************
long double rnorm(const double A1,const double A2,const int tp) {
double N;
float PI = 3.14159265358979323846;
static long double u1 =0.959678;
static long double u2 =0.87675545;
u1 =un(1+(tp-1)*5);
u2 =un(2+(tp-1)*5);
N = pow((-2*log(u1)),0.5)*cos(2*PI*u2);
return N*A2+A1;
}
//********************** calculation of factorial ************************
int factorial(const int n) {
int n1=n;
if(n1<0) return 0;
int f=1;
while(n1>1) f*=n1--;
return f;
}
//********************** calculation of numero de pulos ******************
int dpoi(long double k1) {
int temp,i;
long double u=un(5);
long double temp1= check_exp(-k1);
long double temp2=k1;
long double temp3=1;
long double u1;
for(i=0;;i++){
  if(u1<=u) {
    temp2=temp2*k1;
    temp3=(i+1)*temp3;
    u1+=temp2*temp1/temp3;}
  else break;
}
return i;
}
//********************** calculation of numero de pulos ******************
void numero(int& k1) {
k1=kmax+1;
while(k1>kmax) {
  long double u=un(5);
  long double u1= exp(-lambda);
  for(int i=0;i<=kmax;i++) {
    if(u<u1) {k1=i; break;}
    else u1+=pow(lambda,i+1)*exp(-lambda)/factorial(i+1);
  }
}
}
//********************** calculation do sort ****************
void sort(long double a[],const int n){
long double temp;
int j;
for(int i=1;i<n;i++){
```

```
    temp=a[i];
    for(j=i;j>0 && a[j-1]>temp;j--) a[j]=a[j-1];
    a[j]=temp;
  }
  }
  //************************ calculation do sort p/ double *******
  void sort_d(double a[],const int n){
  long double temp;
  int j;
  for(int i=1;i<n;i++){
    temp=a[i];
    for(j=i;j>0 && a[j-1]>temp;j--) a[j]=a[j-1];
    a[j]=temp;
  }
  }
  //************************ calculation do qual *****************
  long double qual(long double a[],double a1[],int size,double value) {
  int i,temp;
  temp=size;
  for(i=1;i<=size;i++)if(value<a1[i]){temp=(i-1);break;}
  return a[temp];
  }
  //************************ calculation do qual *****************
  long double qualg(long double a[],double a1[],int size,double value) {
  int i,temp;
  for(i=(size+1);i>=0;i--)if(value>a1[i]){temp=i;break;}
  return a[temp];
  }
  //************************ calculation do qual ******************
  int qual_ig(long double a[],double a1[],int size,double value) {
  int i,temp;
  for(i=(size+1);i>=0;i--)if(value>a1[i]){temp=i;break;}
  return temp;
  }
  //************************ calculation do qual *****************
  int qual_index(long double a[],double a1[],int size,double value) {
  int i,temp;
  temp=size;
  for(i=1;i<=size;i++)if(value<a1[i]){temp=(i-1);break;}
  return (temp+1);
  }
  //************************ calculation do qual_int *****************
  long double qual_a(long double a[],int a1[],int size,int value) {
  int i,temp;
  temp=size;
  for(i=1;i<=size;i++)if(value<a1[i]){temp=(i-1);break;}
  return a[temp];
  }
  //************************ calculation do movimento *****************
  int movimento(const int k2){
  double d,n,b;
  long double u1;
  static int j=1;
  j++;
  if(j==5) j=1;
  if(k2==0){
    d=0;
    if (lambda<=1) b=c*lambda;
    else b=c;
    n=1-b;
  }
  else {
    if(k2==kmax) b=0;
    else {
      if(lambda<=k2+1) b=c*lambda/(k2+1);
      else b=c;
    }
    if (lambda<k2) d=c;
    else d=c*k2/lambda;
    n=1-b-d;
  }
  u1=un(1+(j-1)*5);
  if(u1<b) return 1;
  else{
    if(u1<(d+b)) return 2;
    else return 3;
  }
  }
  //************************ calculation of sample ************************
  int sample(int k1) {
```

```
long double u=un(5);
for(int i=1;i<=k1;i++) if(u*k1<=i) {return i;break;}
}
//********************** calculation of inicio ************************
int inicio(const double T[],const int n ,const long double valor) {
int i,ini;
ini=n;
for(i=0;i<n;i++) if(T[i] > valor) {ini=i; break;}
return ini;
}
//********************** calculation of fim ************************
int fim(const double T[],const int n,const long double valor) {
int i,ini;
ini=n-1;
for(i=1;i<n;i++) if(T[i] >= valor) {ini=i-1; break;}
return ini;
}
//********************** calculation of inicio ************************
int inicio_d(const double T[],const int n ,const double valor) {
int i,ini;
ini=n;
for(i=0;i<n;i++) if(T[i] > valor) {ini=i; break;}
return ini;
}
//********************** calculation of fim ************************
int fim_d(const double T[],const int n,const double valor) {
int i,ini;
ini=n-1;
for(i=1;i<n;i++) if(T[i] >= valor) {ini=i-1; break;}
return ini;
}
//********************** calculation of peso ************************
long double peso(const double T[],const double T1[],const long double s1,const long
double s2,const int n,const int n1,const int n2) {
long double valor;
if(n1<=n2) valor=(T[n2]-T[n1]+(T[n1]-T[n1-1])*(T1[n1]-s1)/(T1[n1]-T1[n1-1])+(T[n2+1]-
T[n2])*(s2-T1[n2])/(T1[n2+1]-T1[n2]));
else valor=0;
return valor;
}
//********************** calculation of increase ************************
void increase(long double T[],long double value,int w,int n) {
int i;
for(i=(n-1);i>w;i--) T[i]=T[i-1];
T[w] = value;
}
//********************** calculation of increase/ double ************************
void increase_d(double T[], double value,int w,int n) {
int i;
for(i=(n-1);i>w;i--) T[i]=T[i-1];
T[w] = value;
}
//********************** calculation of increase/ int ************************
void increase_i(int T[], int value,int w,int n) {
int i;
for(i=(n-1);i>w;i--) T[i]=T[i-1];
T[w] = value;
}
//********************** calculation of decrease ************************
void decrease(long double T[],int w,int n) {
int i;
if((n-1)>w) for(i=w;i<(n-1);i++) T[i] = T[i+1];
}
//********************** calculation of decrease p/ double *************
void decrease_d(double T[],int w,int n) {
int i;
if((n-1)>w) for(i=w;i<(n-1);i++) T[i] = T[i+1];
}
//********************** calculation of decrease p/ int *************
void decrease_i(int T[],int w,int n) {
int i;
if((n-1)>w) for(i=w;i<(n-1);i++) T[i] = T[i+1];
}
//********************** calculation of S (jumps) for fr_sa ******************
int fsaS(int s1,long double l1,long double l2,int s2,int N[],double W[],int tp) {
int np,i,jump,num;
long double u,sum,l3,l4,cons,cons1;
num= s2-s1-1;
long double* C;
C = new long double[num];
```

```
l3=(l2-l1);
l4=check_log(l1)-check_log(l2);
C[0]=0;
cons=0;
cons1=0;
if(num>2) for(i=1;i<num;i++){
  cons=cons+N[s1+i]*l4;
  cons1=cons1+l3*W[s1+i];
  C[i]=cons + cons1;
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
sum=sum/num;
cons=C[0]-sum;
for(i=0;i<num;i++){
  C[i]=C[i]-sum;
  if(cons<C[i]) cons=C[i];
}
if(cons>700) for(i=0;i<num;i++) C[i]=C[i]-(cons-700);
for(i=0;i<num;i++) C[i]=check_exp(C[i]);
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(15+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(u>C[i-1] && u<=C[i]) {hp=i;break;}
delete[] C;
jump=s1+hp+1;
return jump;
}
//********************** calculation of S (jumps) for Mi *******************
int fsaS2(int s1,long double l1,long double l2,int s2,int N[],double W[],double sig,int
tp){
int hp,i,jump,num;
long double  u,sum,l3,l4,cons,cons1;
num= s2-s1-1;
long double* C;
C = new long double[num];
l3=(l1-l2)/(pow(sig,2));
l4=(pow(l2,2)-pow(l1,2))/(2*pow(sig,2));
C[0]=0;
if(num>1) for(i=1;i<num;i++) {
C[i]=C[i-1] + N[s1+i]*l4 + W[s1+i]*l3;
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
sum=sum/num;
cons=C[0]-sum;
for(i=0;i<num;i++){
  C[i]=C[i]-sum;
  if(cons<C[i]) cons=C[i];
}
if(cons>700) for(i=0;i<num;i++) C[i]=C[i]-(cons-700);
for(i=0;i<num;i++) if(C[i]< -700) C[i]=0; else C[i]=check_exp(C[i]);
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(15+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(u>C[i-1] && u<=C[i]) {hp=i;break;}
delete[] C;
jump=s1+hp+1;
return jump;
}
//********************** calculation of S (jumps) for ft *******************
double ftS(double W[][Amax][2],double s1,long double l1,long double l2,double s2,double
X[],int N[],int S_a[][32],long double L[][31],int k[],int size,int fin,int beg,int tp) {
int num=fin-beg+2;
int hp,i,j,temp,z,u2;
long double  u,sum,u1,x,l3,l4,cons,cons1,cons2,b,l12,n_wei;
double jump,x1,c1;
long double* C;
C = new long double[num];
l3=(l2-l1);
l4=check_log(l1)-check_log(l2);
if(s2<=X[fin] || s1>=X[beg]) {cout<<"problema1",num=num-1;}
if(s2<=X[fin] && s1>=X[beg]) {cout<<"problema2",num=num-1;}
cons=0;
```

```
cons1=0;
temp=transf(s1);
n_wei=0;
if(temp>0) {
  for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
    l12=qual_a(L[z+2],S_a[z],k[z+2],u2);
    b=check_log(W[temp][u2][z]/W[temp-1][u2][z])/365;
    n_wei+=l12*W[temp-1][u2][z]*check_exp(b*(int(s1)-W1[temp]));
  }
}
else{
  for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
    l12=qual_a(L[z+2],S_a[z],k[z+2],u2);
    n_wei+=l12*W[0][u2][z];
  }
}
cons2=l3*(X[beg]-s1)*n_wei;
if(cons2>700|| cons2<-740) {c1= cons2;}
else{
  c1=check_exp(cons2)-1;
  if(c1<0) c1=-c1;
  c1=check_log(c1);
}
if(l1!=l2) C[0]=cons + cons1-check_log(n_wei)+c1;
else {C[0]=x1-s1;cout<<"blabla";}
if(fin>=beg) for(i=beg; i<=fin;i++) {
  temp=transf(X[i]);
  cons=cons+cons2;
  if(temp!=0){
    n_wei=0;
    for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
      l12=qual_a(L[z+2],S_a[z],k[z+2],u2);
      b=check_log(W[temp][u2][z]/W[temp-1][u2][z])/365;
      n_wei+=l12*W[temp-1][u2][z]*check_exp(b*(int(X[i])-W1[temp]));
    }
  }
  if(i!=fin) x1=X[i+1]; else x1=s2;
  cons2=l3*(x1-X[i])*n_wei;
  cons1+=l4*N[i];
  if(cons2>700 || cons2<-740) c1= cons2;
  else{
    c1=check_exp(cons2)-1;
    if(c1<0) c1=-c1;
    c1=check_log(c1);
  }
  if(l1!=l2) C[i-beg+1]=cons + cons1-check_log(n_wei)+c1;
  else  C[i-beg+1]=x1-X[i];
}
if(l1!=l2){
  sum=0;
  for(i=0;i<num;i++) sum+=C[i];
  sum=sum/num;
  cons=C[0]-sum;
  for(i=0;i<num;i++){
    C[i]=C[i]-sum;
    if(cons<C[i]) cons=C[i];
  }
  if(cons>700) for(i=0;i<num;i++) C[i]=C[i]-(cons-700);
  for(i=0;i<num;i++) C[i]=check_exp(C[i]);
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(15+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(C[i-1]<u && u<=C[i]){hp=i;break;}
if(hp==0) {
  x1=s1;
  x=X[beg]-s1;
}
else if(hp==num-1) {
  x1=X[fin];
  x=s2-X[fin];
}
else {
  x1=X[beg+hp-1];
  x=X[beg+hp]-X[beg+hp-1];
}
u=un(14+(tp-1)*5);
```

```
jump=double(x1+u*x);
delete[] C;
return jump;
}
//********************** calculation of S (jumps) for gu ********************
double guS(const double s1,const long double l1,const long double l2,const double
s2,const double X[],const int N[],const int N1[],const int size,const int beg,const int
fin,const int tp) {
int num=fin-beg+2;
int hp,i,j;
long double  u,sum,u1,x,l3,l4,cons,cons1,cons2;
double jump,x1,c1;
long double* C;
C = new long double[num];
l3=(l2-l1);
l4=check_log(l1)-check_log(l2);
if(s2<=X[fin] || s1>=X[beg]) {cout<<"problema1",num=num-1;}
if(s2<=X[fin] && s1>=X[beg]) {cout<<"problema2",num=num-1;}
cons1=(N1[beg-1]-N1[fin])*check_log(l2);
cons2=l3*(X[beg]-s1)*N1[beg-1];
if(cons2>700 || cons2<-740) c1= cons2;
else{
  c1=check_exp(cons2)-1;
  if(c1<0) c1=-c1;
  c1=check_log(c1);
}
cons=0; /
if(l1!=l2) C[0]=cons + cons1-check_log(N1[beg-1])+c1;
else {C[0]=X[beg]-s1;cout<<"blabla";}
if(fin >= beg) for(i=beg; i<=fin;i++) {
  if(i==fin && N1[i]==0) {
    if(l3<0) l3=-l3;
    C[i-beg+1]= cons1 + cons +log(l3*(s2-X[fin]));
  }
  else{
    cons1+=(N[i-1]-N[i])*l4;
    cons += cons2;
    if(i!=fin) cons2=l3*(X[i+1]-X[i])*N1[i];
    else cons2=l3*(s2-X[i])*N1[i];
    if(cons2>700 || cons2<-740) c1= cons2;
    else{
      c1=check_exp(cons2)-1;
      if(c1<0) c1=-c1;
      c1=check_log(c1);
    }
    if(N1[i]==0) C[i-beg+1]=cons+cons1;
    else if(l1!=l2) C[i-beg+1]=cons+cons1-check_log(N1[i])+c1;
    else if(i!=fin) C[i-beg+1]=X[i+1]-X[i]; else C[i-beg+1]=s2-X[i];
  }
}
if(l1!=l2){
  sum=0;
  for(i=0;i<num;i++) sum+=C[i];
  sum=sum/num;
  cons=C[0]-sum;
  for(i=0;i<num;i++){
    C[i]=C[i]-sum;
    if(cons<C[i]) cons=C[i];
  }
  if(cons>700) for(i=0;i<num;i++)C[i]=C[i]-(cons-700);
  for(i=0;i<num;i++) C[i]=check_exp(C[i]);
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(15+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(C[i-1]<u && u<=C[i]){hp=i;break;}
if(hp==0) {
  x1=s1;
  x=X[beg]-s1;
}
else if(s2>X[fin] && hp==num-1) {
  x1=X[fin];
  x=s2-X[fin];
}
else {
  x1=X[beg+hp-1];
  x=X[beg+hp]-X[beg+hp-1];
```

```
}
u=un(14+(tp-1)*5);
jump=double(x1+u*x);
delete[] C;
return jump;
}
//************************ calculation of S (jumps) ********************
long double gS(const long double s1,const long double l1,const long double l2,const long
double s2,const double X[],const double W[],const int n,const int beg,const int fin,const
double wei,const double X1[],const int N[],const int n1,const int beg1,const int
fin1,const int tp) {
int num=fin-beg+2;
int hp,i,j;
long double jump,u,sum,u1,x,l3,l4,w1,w2,wtot,cons,cons1,wtemp;
double x1,c1;
long double* C;
C = new long double[num];
l3=l2-l1;
l4=check_log(l1)-check_log(l2);
w1=W[beg]*(X[beg]-s1)/(X[beg]-X[beg-1]);
w2=W[fin+1]*(s2-X[fin])/(X[fin+1]-X[fin]);
wtot=wei+w1+w2;
if(s2<=X[fin] || s1>=X[beg]) {cout<<"problema1",num=num-1;}
if(s2<=X[fin] && s1>=X[beg]) {cout<<"problema2",num=num-1;}
cons=0;
cons1=-l2*(wtot-w1);
c1=check_exp(-l1*w1)-check_exp(-l2*w1);
if(c1<0) c1=-c1;
if(w1>0 && l1!=l2) C[0]=cons1-check_log(w1/(X[beg]-s1))+check_log(c1);
else {C[0]=X[beg]-s1;cout<<"blabla";}
if(fin > beg) for(i=beg; i<fin;i++) {
  if(fin1>=beg1) for(j=beg1;j<=fin1;j++) if(X[i]==X1[j]) cons+=N[j]*l4;
  if(i!=beg) wtemp=W[i];
  else wtemp=w1;
  cons1 = cons1+l2*W[i+1] -l1*wtemp;
  c1=check_exp(-l1*W[i+1])-check_exp(-l2*W[i+1]);
  if(c1<0) c1=-c1;
  if(W[i+1]>0 && l1!=l2) C[i-beg+1]=cons+cons1-check_log(W[i+1]/(X[i+1]-
X[i]))+check_log(c1);
  else C[i-beg+1]=X[i+1]-X[i];
}
if(fin1>=beg1) for(j=beg1;j<=fin1;j++) if(X[fin]==X1[j]) cons+= N[j]*l4;
if(fin!=beg) wtemp=W[fin];
else wtemp=w1;
cons1 = cons1+l2*w2 - l1*wtemp;
c1=check_exp(-l1*w2)-check_exp(-l2*w2);
if(c1<0) c1=-c1;
if(w2>0 && l1!=l2) C[num-1]=cons+cons1-check_log(w2/(s2-X[fin]))+check_log(c1);
else C[num-1]=s2-X[fin];
if(l1!=l2){
  sum=0;
  for(i=0;i<num;i++) sum+=C[i];
  sum=sum/num;
  cons=C[0]-sum;
  for(i=0;i<num;i++){
    C[i]=C[i]-sum;
    if(cons<C[i]) cons=C[i];
  }
  if(cons>700) for(i=0;i<num;i++) C[i]=C[i]-(cons-700);
  for(i=0;i<num;i++) C[i]=check_exp(C[i]);
}
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(15+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(C[i-1]<u && u<=C[i]){hp=i;break;}
if(l1!=l2){
  if(hp==0) {
    wtemp=w1*(l2-l1);
    x1=s1;
    x=X[beg]-s1;
  }
  else if(s2>X[fin] && hp==num-1) {
    wtemp=w2*(l2-l1);
    x1=X[fin];
    x=s2-X[fin];
  }
  else {
```

```
    wtemp=W[beg+hp]*(l2-l1);
    x1=X[beg+hp-1];
    x=X[beg+hp]-X[beg+hp-1];
   }
u=un(14+(tp-1)*5);
jump=x1+x*check_log(u*check_exp(wtemp)-u+1)/wtemp;
}
else jump=x1+u*x;
delete[] C;
return jump;
}
//********************** calculation of S2 (jumps) para severidade*******************
long double gS2(const long double s1,const long double l1,const long double l2,const long
double s2,const double X[],const double H[],const int N[],const int size,const int
beg,const int fin,const int sig,const int tp) {
int num=fin-beg+2;
int hp,i,j;
long double jump,u,sum,u1,x,l3,l4,w1,w2,wtot,cons,cons1,wtemp;
double x1,c1;
long double* C = new long double[num];
l3=2*(l2-l1);
l4=pow(l1,2)-pow(l2,2);
C[0]=0;
c1=X[beg]-s1;
if(fin > beg) for(i=beg; i<fin;i++) {
  C[i-beg+1]=C[i-beg]+log(X[i+1]-X[i])-check_log(c1)- (N[i]*l4+H[i]*l3)/(2*pow(sig,2));
  c1=X[i+1]-X[i];
}
if(s2>X[fin]) C[num-1]=C[fin-beg]+check_log(s2-X[fin])-check_log(c1)-
(N[fin]*l4+H[fin]*l3)/(2*pow(sig,2));
else num=num-1;
sum=0;
for(i=0;i<num;i++)sum+=C[i];
sum=sum/num;
cons=C[0]-sum;
for(i=0;i<num;i++){
  C[i]=(C[i]-sum);
  if(cons<C[i]) cons=C[i];
}
if(cons>700) for(i=0;i<num;i++)C[i]=C[i]-(cons-700);
for(i=0;i<num;i++)C[i]=check_exp(C[i]);
sum=0;
for(i=0;i<num;i++) sum+=C[i];
C[0]=C[0]/sum;
for(i=1; i<num;i++) C[i]=C[i-1]+C[i]/sum;
hp=0;
u=un(5+(tp-1)*5);
if(num>1) for(i=1; i< num;i++) if(C[i-1]<u && u<=C[i]){hp=i;break;}
if(hp==0) {
  x1=s1;
  x=X[beg]-s1;
}
else if(hp==num-1 && s2>X[fin]) {
    x1=X[fin];
    x=s2-X[fin];
    }
else {
  x1=X[beg+hp-1];
  x=X[beg+hp]-X[beg+hp-1];
}
u=un(4+(tp-1)*5);
jump=x1+u*x;
delete []C;
return jump;
}
//********************** calculation of aceii2 ************************
long double acei_i2(const int int1,const int int2,const int N1,const int Nj,const double
L1,const double Lj,const int k1,const long double h1,const long double h2,const long
double h3,const double sig,const int alpha,const double beta){
long double lr,pr,jac;
long double temp,temp1;
float PI = 3.14159265358979323846;
lr = (2*h1*L1-N1*pow(h1,2)+2*h3*(Lj-L1)-(Nj-N1)*pow(h3,2)-
2*h2*Lj+Nj*pow(h2,2))/(2*sig*sig);
jac=2*PI;
jac=pow(jac,0.5);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1*sig*jac);
pr=check_log(pr)-(pow(alpha,2)-2*alpha*(h1+h2-h3)+pow(h1,2)+pow(h3,2)-
pow(h2,2))/(2*pow(beta,2));
if((h1+h3)<0) temp=-(h1+h3);
```

```
else temp=h1+h3;
if(h2<0) temp1=-h2;
else temp1=h2;
jac=-100;
if(temp!=0 && temp1!=0) jac= 2*check_log(temp)-check_log(temp1);
return lr+pr+jac;
}
//********************** calculation of acei fot severidade ***********
long double acei2(const double int1,const double int2,const int N1,const int Nj,const
long double L1,const long double Lj,const int k1,const long double h1,const long double
h2,const long double h3,const int sig,const int alpha,const double beta){
long double lr,pr,jac;
long double temp,temp1;
float PI = 3.14159265358979323846;
lr = (2*h1*L1-N1*pow(h1,2)+2*h3*(Lj-L1)-(Nj-N1)*pow(h3,2)-
2*h2*Lj+Nj*pow(h2,2))/(2*sig*sig);
jac=2*PI;
jac=pow(jac,0.5);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1*sig*jac);
pr = check_log(pr) - (pow(alpha,2)-2*alpha*(h1+h2-h3)+pow(h1,2)+pow(h3,2)-
pow(h2,2))/(2*pow(beta,2));
if((h1+h3)<0) temp=-(h1+h3);
else temp=h1+h3;
if(h2<0) temp1=-h2;
else temp1=h2;
jac=-100;
if(temp!=0 && temp1!=0) jac= 2*check_log(temp)-check_log(temp1);
return lr+pr+jac;
}
//********************** calculation of aceitacao ************************
long double aceitacao_i(const int int1,const int int2,const int N1,const int Nj,const
double L1,const double Lj,const int k1,const long double h1,const long double h2,const
long double h3,const int A,const double B) {
long double lr,pr,jac;
lr = N1 * check_log(h1) + (Nj - N1) * check_log(h3) - Nj * check_log(h2) + h2 * Lj - h1 *
L1 - h3 * (Lj - L1);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1 * factorial(A-1));
pr = check_log(pr) + A*check_log(B)- B * (h1 + h3 - h2) + (A-1)*check_log(h1*h3/h2);
jac = 2*check_log(h1 + h3)-check_log(h2);
return lr+pr+jac;
}
//********************** calculation of aceitacao ************************
long double aceitacao_d(const double int1,const double int2,const int N1,const int
Nj,const long double L1,const long double Lj,const int k1,const long double h1,const long
double h2,const long double h3,const int A,const double B) {
long double lr,pr,jac;
lr = N1 * check_log(h1) + (Nj - N1) * check_log(h3) - Nj * check_log(h2) + h2 * Lj - h1 *
L1 - h3 * (Lj - L1);
pr = 2 * (2 * k1 + 3) * int2 * (int1-int2)/(Smax * int1 * factorial(A-1));
pr = check_log(pr) + A*check_log(B)- B * (h1 + h3 - h2) + (A-1)*check_log(h1*h3/h2);
jac = 2*check_log(h1 + h3)-check_log(h2);
return lr+pr+jac;
}
//********************** updation of gu ************************
void fr_gu(double S[],long double L[],int& k,const double X[],const int N[],const int
N1[],const int size, const int A,const double B,int TOT[]){
int i,j,beg1,beg,fin1,fin,num,u2,pos,mov,num1;
long double u,u1,a,s1,ac,cons,h1,h2,h3;
double wei,wei1;
static int tipo=1;
tipo++;
if(tipo==3) tipo=1;
beg=inicio_d(X,size,S[0]);
fin=fim_d(X,size,S[1]);
num=0;
wei=0;
if (fin>=beg) {
  wei=(X[beg]-S[0])*double(N1[beg-1])+(S[1]-X[fin])*double(N1[fin]);
  num=N[beg-1]-N[fin];
  for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*N1[i];
}
else wei=N1[beg-1]*(S[1]-S[0]);
wei=wei/divisor_g;
L[0]=rgamma(num+A,wei+B,tipo);
L[0]=L[0]/divisor_g;
if(k>0) for(j=1; j <= k; j++) {
  fin=fim_d(X,size,S[j+1]);
  if(L[j-1]!=L[j] && beg<=fin && N1[beg-1]!=0) S[j]=guS(S[j-1],L[j-
1],L[j],S[j+1],X,N,N1,size,beg,fin,tipo);
  else {
```

```
    u=un(15+(tipo-1)*5);
    S[j]=S[j-1]+u*(S[j+1]-S[j-1]);
  }
  if(S[j]>S[j+1]) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
  if(S[j]<S[j-1]) cout<<"erro2";
  beg=inicio_d(X,size,S[j]);
  num=0;
  wei=0;
  if (fin>=beg) {
    wei=(X[beg]-S[j])*double(N[beg-1])+(S[j+1]-X[fin])*double(N1[fin]);
    num=N[beg-1]-N[fin];
    for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*double(N1[i]);
  }
  else wei= N1[beg-1]*(S[j+1]-S[j]);
  wei=wei/divisor_g;
  L[j]=rgamma(num+A,wei+B,tipo);
  L[j]=L[j]/divisor_g;
}
beg=inicio_d(X,size,S[0]);
fin=fim_d(X,size,S[1]);
num=0;
wei=0;
if (fin>=beg) {
  wei=(X[beg]-S[0])*double(N1[beg-1])+(S[1]-X[fin])*double(N1[fin]);
  num=N[beg-1]-N[fin];
  for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*N1[i];
}
else wei=N1[beg-1]*(S[1]-S[0]);
wei=wei/divisor_g;
L[0]=rgamma(num+A,wei+B,tipo);
L[0]=L[0]/divisor_g;
if(k>0) for(j=1; j <= k; j++){
  beg=inicio_d(X,size,S[j]);
  fin=fim_d(X,size,S[j+1]);
  num=0;
  wei=0;
  if (fin>=beg) {
    wei=(X[beg]-S[j])*double(N[beg-1])+(S[j+1]-X[fin])*double(N1[fin]);
    num=N[beg-1]-N[fin];
    for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*double(N1[i]);
  }
  else wei= N1[beg-1]*(S[j+1]-S[j]);
  wei=wei/divisor_g;
  L[j]=rgamma(num+A,wei+B,tipo);
  L[j]=L[j]/divisor_g;
}
mov=movimento(k);
u2=0;
num=0;
wei=0;
num1=0;
wei1=0;
if(mov!=3) TOT[1+mov]++;
if(mov==1){
  sl =double(un(14+(tipo-1)*5)*Smax);
  pos=0;
  while(u2 < k) {
    u2 = 0;
    sl =double(un(14+(tipo-1)*5)*Smax);
    pos = 0;
    for(i=1;i<=k;i++) {
      if(S[i] != sl) u2++;
      if(S[i] < sl) pos = i;
    }
  }
  beg=inicio_d(X,size,S[pos]);
  fin=fim_d(X,size,S[pos+1]);
  if (fin>=beg){
    num=N[beg-1]-N[fin];
    wei=double(N1[beg-1])*(X[beg]-S[pos])+double(N1[fin])*(S[pos+1]-X[fin]);
    for(i=beg;i<fin;i++) wei+=double(N1[i])*(X[i+1]-X[i]);
  }
  else wei= N1[beg-1]*(S[pos+1]-S[pos]);
  fin=fim_d(X,size,sl);
  if (fin>=beg){
    num1=N[beg-1]-N[fin];
    wei1=double(N1[beg-1])*(X[beg]-S[pos])+double(N1[fin])*(sl-X[fin]);
    for(i=beg;i<fin;i++) wei1+=double(N1[i])*(X[i+1]-X[i]);
  }
  else wei1=N1[beg-1]*(sl-S[pos]);
```

```
    if(wei1!=0 && wei1!=wei){
      h2 = L[pos];
      cons = (1/un(11+(tipo-1)*5) - 1);
      h1 = h2 * pow(cons,((sl-S[pos+1])/(S[pos+1]-S[pos])));
      h3 = cons * h1;
      ac=aceitacao_d(S[pos+1]-S[pos],sl-S[pos],num1,num,wei1,wei,k,h1,h2,h3,A,B*divisor_g);
      if(ac>=0) ac=1;
      else ac=check_exp(ac);
      u1=un(14+(tipo-1)*5);
      if(u1 <= ac) {
        increase_d(S,sl,pos + 1,kmax +2);
        L[pos] = h1;
        increase(L,h3,pos + 1,kmax+1);
        k = k + 1;
        TOT[mov-1]++;
      }
    }
  }
  if(mov==2){
    pos = sample(k);
    sl =S[pos];
    beg=inicio(X,size,S[pos-1]);
    fin=fim(X,size,S[pos+1]);
    if (fin>=beg) {
      num=N[beg-1]-N[fin];
      wei=N1[beg-1]*(X[beg]-S[pos-1])+N1[fin]*(S[pos+1]-X[fin]);
      for(i=beg;i<fin;i++) wei+=N1[i]*(X[i+1]-X[i]);
    }
    else wei=N1[beg-1]*(S[pos+1]-S[pos-1]);
    fin=fim(X,size,S[pos]);
    if (fin>=beg){
      num1=N[beg-1]-N[fin];
      wei1=N1[beg-1]*(X[beg]-S[pos-1])+N1[fin]*(S[pos]-X[fin]);
      for(i=beg;i<fin;i++) wei1+=N1[i]*(X[i+1]-X[i]);
    }
    else wei1=N1[beg-1]*(S[pos]-S[pos-1]);
    if(wei1!=0 && wei1!=wei){
      h1 = L[pos-1];
      h3 = L[pos];
      h2 = check_exp(((S[pos]-S[pos-1]) * check_log(h1) + (S[pos+1]-S[pos]) *
check_log(h3))/(S[pos+1]-S[pos-1]));
      ac=aceitacao_d(S[pos+1]-S[pos-1],S[pos]-S[pos-
1],num1,num,wei1,wei,k,h1,h2,h3,A,B*divisor_g);
      if(ac<=0) ac=1;
      else ac = check_exp(-ac);
      u1 = un(15+(tipo-1)*5);
      if(u1 <= ac) {
        k = k - 1;
        decrease_d(S,pos,kmax +2);
        L[pos-1] = h2;
        decrease(L,pos + 1,kmax+1);
        TOT[mov-1]++;
      }
    }
  }
  beg=inicio_d(X,size,S[0]);
  fin=fim_d(X,size,S[1]);
  num=0;
  wei=0;
  if (fin>=beg) {
    wei=(X[beg]-S[0])*double(N1[beg-1])+(S[1]-X[fin])*double(N1[fin]);
    num=N[beg-1]-N[fin];
    for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*N1[i];
  }
  else wei=N1[beg-1]*(S[1]-S[0]);
  wei=wei/divisor_g;
  L[0]=rgamma(num+A,wei+B,tipo);
  L[0]=L[0]/divisor_g;
  if(k>0) for(j=1; j <= k; j++){
    fin=fim_d(X,size,S[j+1]);
    beg=inicio_d(X,size,S[j]);
    num=0;
    wei=0;
    if(fin>=beg) {
      wei=(X[beg]-S[j])*double(N[beg-1])+(S[j+1]-X[fin])*double(N1[fin]);
      num=N[beg-1]-N[fin];
      for(i=beg;i<fin;i++) wei+=(X[i+1]-X[i])*double(N1[i]);
    }
    else wei= N1[beg-1]*(S[j+1]-S[j]);
    wei=wei/divisor_g;
```

```
  L[j]=rgamma(num+A,wei+B,tipo);
  L[j]=L[j]/divisor_g;
}
}
//********************** updation of ft ************************
void fr_ft(double S[],long double L[][31],int k[],double X[],int N[],int size,double
W[][Amax][2],int S_a[][32],int A,double B,int TOT[]) {
int i,j,beg1,beg,fin1,fin,num,u2,pos,mov,num1,temp,temp1,z;
long double u,u1,a,sl,ac,cons,h1,h2,h3,l1,b;
double wei,wei1;
static int tipo=1;
tipo++;
if(tipo==3) tipo=1;
beg=inicio_d(X,size,S[0]);
fin=fim_d(X,size,S[1]);
wei=0;
temp=transf(S[0]);
temp1=transf(S[1]);
for(i=temp; i<=temp1;i++){
  if(i==temp) u1=S[0]; else u1=W1[i];
  if(i==temp1) u=S[1]; else u= W1[i+1];
  for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
    l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
    if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
      b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
      b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
      }
    else b=W[i][u2][z]*(u-u1);
  wei+=l1*b;
  }
}
num=0;
if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
L[0][0]=rgamma(num+A,wei+B,tipo);
if(k[0]>0) for(j=1; j <= k[0]; j++) {
  fin=fim_d(X,size,S[j+1]);
  if(L[0][j-1]!=L[0][j]){
  if(fin>=beg)S[j]=ftS(W,S[j-1],L[0][j-1],L[0][j],S[j+1],X,N,S_a,L,k,size,fin,beg,tipo);
    else {
      u=un(15+(tipo-1)*5);
      S[j]=S[j-1]+u*(S[j+1]-S[j-1]);
    }
  }
  if(S[j]>S[j+1]) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
  if(S[j]<S[j-1]) cout<<"erro2";
  beg=inicio_d(X,size,S[j]);
  num=0;
  wei=0;
  temp=transf(S[j]);
  temp1=transf(S[j+1]);
  for(i=temp; i<=temp1;i++){
    if(i==temp) u1=S[j]; else u1=W1[i];
    if(i==temp1) u=S[j+1]; else u= W1[i+1];
    for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
      l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
      if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
  b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
  b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
  }
        else b=W[i][u2][z]*(u-u1);
    wei+=l1*b;
    }
  }
  if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
  L[0][j]=rgamma(num+A,wei+B,tipo);
}
beg=inicio_d(X,size,S[0]);
fin=fim_d(X,size,S[1]);
wei=0;
temp=transf(S[0]);
temp1=transf(S[1]);
for(i=temp; i<=temp1;i++){
  if(i==temp) u1=S[0]; else u1=W1[i];
  if(i==temp1) u=S[1]; else u= W1[i+1];
  for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
    l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
    if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
      b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
      b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
      }
```

```
      else b=W[i][u2][z]*(u-u1);
    wei+=l1*b;
    }
  }
  num=0;
  if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
  L[0][0]=rgamma(num+A,wei+B,tipo);
  if(k[0]>0) for(j=1; j <= k[0]; j++) {
    fin=fim_d(X,size,S[j+1]);
    beg=inicio_d(X,size,S[j]);
    num=0;
    wei=0;
    temp=transf(S[j]);
    temp1=transf(S[j+1]);
    for(i=temp; i<=temp1;i++){
      if(i==temp) u1=S[j]; else u1=W1[i];
      if(i==temp1) u=S[j+1]; else u= W1[i+1];
      for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
        l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
        if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
    b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
    b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
    }
        else b=W[i][u2][z]*(u-u1);
      wei+=l1*b;
      }
    }
    if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
    L[0][j]=rgamma(num+A,wei+B,tipo);
  }
  mov=movimento(k[0]);
  u2=0;
  if(mov!=3) TOT[1+mov]++;
  if(mov==1){
    s1 =un(14+(tipo-1)*5)*Smax;
    pos=0;
    if(k[0]>0) while(u2 < k[0]) {
      u2 = 0;
      s1=un(14+(tipo-1)*5)*Smax;
      pos = 0;
      for(i=1;i<=k[0];i++) {
        if(S[i] != s1) u2++;
        if(S[i] < s1) pos = i;
      }
    }
    beg=inicio_d(X,size,S[pos]);
    fin=fim_d(X,size,S[pos+1]);
    num=0;
    wei=0;
    temp=transf(S[pos]);
    temp1=transf(S[pos+1]);
    for(i=temp; i<=temp1;i++){
      if(i==temp) u1=S[pos]; else u1=W1[i];
      if(i==temp1) u=S[pos+1]; else u= W1[i+1];
      for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
        l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
        if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
          b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
          b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
          }
        else b=W[i][u2][z]*(u-u1);
      wei+=l1*b;
      }
    }
    if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
    fin=fim_d(X,size,s1);
    num1=0;
    wei1=0;
    temp=transf(S[pos]);
    temp1=transf(s1);
    for(i=temp; i<=temp1;i++){
      if(i==temp) u1=S[pos]; else u1=W1[i];
      if(i==temp1) u=s1; else u= W1[i+1];
      for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
        l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
        if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
          b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
          b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
          }
        else b=W[i][u2][z]*(u-u1);
```

```
      wei1+=l1*b;
      }
    }
    if (fin>=beg) for(i=beg;i<=fin;i++) num1+=N[i];
    if(wei1!=0 && wei1!=wei){
      h2 = L[0][pos];
      cons = (1/un(11+(tipo-1)*5) - 1);
      h1 = h2 * pow(cons,((s1-S[pos+1])/(S[pos+1]-S[pos])));
      h3 = cons * h1;
      ac=aceitacao_d(S[pos+1]-S[pos],s1-S[pos],num1,num,wei1,wei,k[0],h1,h2,h3,A,B);
      if(ac>=0) ac=1;
      else ac=check_exp(ac);
      u1=un(14+(tipo-1)*5);
      if(u1 <= ac) {
        increase_d(S,s1,pos + 1,kmax +2);
        L[0][pos] = h1;
        increase(L[0],h3,pos + 1,kmax+1);
        k[0] = k[0] + 1;
        TOT[mov-1]++;
      }
    }
  }
  if(mov==2){
    pos = sample(k[0]);
    s1 =S[pos];
    beg=inicio(X,size,S[pos-1]);
    fin=fim(X,size,S[pos+1]);
    num=0;
    wei=0;
    temp=transf(S[pos-1]);
    temp1=transf(S[pos+1]);
    for(i=temp; i<=temp1;i++){
      if(i==temp) u1=S[pos-1]; else u1=W1[i];
      if(i==temp1) u=S[pos+1]; else u= W1[i+1];
      for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
        l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
        if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
          b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
          b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
          }
        else b=W[i][u2][z]*(u-u1);
      wei+=l1*b;
      }
    }
    if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
    fin=fim(X,size,S[pos]);
    num1=0;
    wei1=0;
    temp=transf(S[pos]);
    temp1=transf(s1);
    for(i=temp; i<=temp1;i++){
      if(i==temp) u1=S[j]; else u1=W1[i];
      if(i==temp1) u=s1; else u= W1[i+1];
      for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
        l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
        if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
          b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
          b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
          }
        else b=W[i][u2][z]*(u-u1);
      wei1+=l1*b;
      }
    }
    if (fin>=beg) for(i=beg;i<=fin;i++) num1+=N[i];
    if(wei1!=0 && wei1!=wei){
      h1 = L[0][pos-1];
      h3 = L[0][pos];
      h2 = check_exp(((S[pos]-S[pos-1]) * check_log(h1) + (S[pos+1]-S[pos]) *
check_log(h3))/(S[pos+1]-S[pos-1]));
      ac=aceitacao_d(S[pos+1]-S[pos-1],S[pos]-S[pos-
1],num1,num,wei1,wei,k[0],h1,h2,h3,A,B);
      if(ac<=0) ac=1;
      else ac = check_exp(-ac);
      u1 = un(15+(tipo-1)*5);
      if(u1 <= ac) {
        k[0] = k[0] - 1;
        decrease_d(S,pos,kmax +2);
        L[0][pos-1] = h2;
        decrease(L[0],pos + 1,kmax+1);
        TOT[mov-1]++;
```

```
    }
  }
}
beg=inicio_d(X,size,S[0]);
fin=fim_d(X,size,S[1]);
wei=0;
temp=transf(S[0]);
temp1=transf(S[1]);
for(i=temp; i<=temp1;i++){
  if(i==temp) u1=S[0]; else u1=W1[i];
  if(i==temp1) u=S[1]; else u= W1[i+1];
  for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
    l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
    if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
      b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
      b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
      }
    else b=W[i][u2][z]*(u-u1);
  wei+=l1*b;
  }
}
num=0;
if (fin>=beg) {for(i=beg;i<=fin;i++) num+=N[i];}
L[0][0]=rgamma(num+A,wei+B,tipo);
if(k[0]>0) for(j=1; j <= k[0]; j++) {
  fin=fim_d(X,size,S[j+1]);
  beg=inicio_d(X,size,S[j]);
  num=0;
  wei=0;
  temp=transf(S[j]);
  temp1=transf(S[j+1]);
  for(i=temp; i<=temp1;i++){
    if(i==temp) u1=S[j]; else u1=W1[i];
    if(i==temp1) u=S[j+1]; else u= W1[i+1];
    for(u2=0;u2<Amax;u2++) for(z=0;z<2;z++) {
      l1=qual_a(L[z+2],S_a[z],k[z+2],u2);
      if(i>0 && W[i][u2][z]!=W[i-1][u2][z]) {
        b=check_log(W[i][u2][z]/W[i-1][u2][z])/365;
        b=W[i-1][u2][z]*(check_exp(b*(u-W1[i]))-check_exp(b*(u1-W1[i])))/b;
      }
      else b=W[i][u2][z]*(u-u1);
    wei+=l1*b;
    }
  }
  if (fin>=beg) for(i=beg;i<=fin;i++) num+=N[i];
  L[0][j]=rgamma(num+A,wei+B,tipo);
}
}
//********************** updation of fsa ************************
void fr_sa(int S[],long double L[],int& k,int N[],double W[],int A,double B,int TOT[]){
int i,j,beg1,beg,fin1,fin,num,u2,pos,mov,num1,temp,temp1,s1,z;
long double u,u1,a,ac,cons,h1,h2,h3,l1,b;
double wei,wei1;
static int tipo=1;
tipo++;
if(tipo==3) tipo=1;
num=0;
wei=0;
if(k==0) s1=Amax; else s1=S[1];
for(i=0; i<s1;i++){
  num+=N[i];
  wei+=W[i];
}
wei=wei/divisor_age;
L[0]=rgamma(num+A,wei+B,tipo);
L[0]=L[0]/divisor_age;
if(k>0) for(j=1; j <= k; j++){
  if(j==k) s1=Amax; else s1=S[j+1];
  num=s1-S[j-1];
  if(num>2){
    if(L[j-1]!=L[j]) S[j]=fsaS(S[j-1],L[j-1],L[j],s1,N,W,tipo);
    else S[j]= S[j-1]+ sample(s1-S[j-1]-1);
    if(S[j]>=s1) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
    if(S[j]<=S[j-1]) cout<<"erro2";
  }
  num=0;
  wei=0;
  for(i=S[j]; i<s1;i++){
    num+=N[i];
    wei+=W[i];
```

```
    }
    wei=wei/divisor_age;
    L[j]=rgamma(num+A,wei+B,tipo);
    L[j]=L[j]/divisor_age;
  }
  num=0;
  wei=0;
  if(k==0) sl=Amax; else sl=S[1];
  for(i=0; i<sl;i++){
    num+=N[i];
    wei+=W[i];
  }
  wei=wei/divisor_age;
  L[0]=rgamma(num+A,wei+B,tipo);
  L[0]=L[0]/divisor_age;
  if(k>0) for(j=1; j <= k; j++){
    if(j==k) sl=Amax; else sl=S[j+1];
    num=0;
    wei=0;
    for(i=S[j]; i<sl;i++){
      num+=N[i];
      wei+=W[i];
    }
    wei=wei/divisor_age;
    L[j]=rgamma(num+A,wei+B,tipo);
    L[j]=L[j]/divisor_age;
  }
  mov=movimento(k);
  u2=0;
  num=0;
  wei=0;
  num1=0;
  wei1=0;
  if(mov!=3) TOT[1+mov]++;
  if(mov==1){
    sl =sample((Amax-1));
    pos=0;
    while(u2 < k) {
      u2 = 0;
      sl =sample((Amax-1));
      pos = 0;
      for(i=1;i<=k;i++) {
        if(S[i] != sl) u2++;
        if(S[i] < sl) pos = i;
      }
    }
    for(i=S[pos]; i<S[pos+1];i++){num+=N[i];wei+=W[i];}
    for(i=S[pos]; i<sl;i++){num1+=N[i];wei1+=W[i];}
    if(wei1!=0 && wei1!=wei){
      h2 = L[pos];
      cons = (1/un(11+(tipo-1)*5) - 1);
      h1 = h2 * pow(cons,((sl-S[pos+1])/(S[pos+1]-S[pos])));
      h3 = cons * h1;
      ac=aceitacao_i(S[pos+1]-S[pos],sl-
S[pos],num1,num,wei1,wei,k,h1,h2,h3,A,B*divisor_age);
      if(ac>=0) ac=1;
      else ac=check_exp(ac);
      u1=un(14+(tipo-1)*5);
      if(u1 <= ac) {
        increase_i(S,sl,pos + 1,kmax +2);
        L[pos] = h1;
        increase(L,h3,pos + 1,kmax+1);
        k = k + 1;
        TOT[mov-1]++;
      }
    }
  }
  if(mov==2){
    pos = sample(k);
    sl =S[pos];
    for(i=S[pos-1]; i<S[pos+1];i++){num+=N[i];wei+=W[i];}
    for(i=S[pos-1]; i<S[pos];i++){num1+=N[i];wei1+=W[i];}
    if(wei1!=0 && wei1!=wei){
      h1 = L[pos-1];
      h3 = L[pos];
      h2 = check_exp(((S[pos]-S[pos-1]) * check_log(h1) + (S[pos+1]-S[pos]) *
check_log(h3))/(S[pos+1]-S[pos-1]));
      ac=aceitacao_i(S[pos+1]-S[pos-1],S[pos]-S[pos-
1],num1,num,wei1,wei,k,h1,h2,h3,A,B*divisor_age);
      if(ac<=0) ac=1;
```

```
      else ac = check_exp(-ac);
      u1 = un(15+(tipo-1)*5);
      if(u1 <= ac) {
        k = k - 1;
        decrease_i(S,pos,kmax +2);
        L[pos-1] = h2;
        decrease(L,pos + 1,kmax+1);
        TOT[mov-1]++;
      }
    }
  }
  num=0;
  wei=0;
  if(k==0) sl=Amax; else sl=S[1];
  for(i=0; i<sl;i++){
    num+=N[i];
    wei+=W[i];
  }
  wei=wei/divisor_age;
  L[0]=rgamma(num+A,wei+B,tipo);
  L[0]=L[0]/divisor_age;
  if(k>0) for(j=1; j <= k; j++){
    if(j==k) sl=Amax; else sl=S[j+1];
    num=0;
    wei=0;
    for(i=S[j]; i<sl;i++){
      num+=N[i];
      wei+=W[i];
    }
    wei=wei/divisor_age;
    L[j]=rgamma(num+A,wei+B,tipo);
    L[j]=L[j]/divisor_age;
  }
}
//********************** updation of severidade_sa ************************
void severidade_sa(int S[],long double L[],int& k,int N[],double W[],int alpha,double
beta,double sig,int TOT[]){
int i,j,beg,fin,num,u2,pos,mov,num1,temp,temp1,sl,z;
long double u,u1,a,ac,cons,h1,h2,h3,l1,b;
double wei,wei1;
static int tipo=1;
tipo++;
if(tipo==3) tipo=1;
for(j=0; j <= k; j++){
  num=0;
  wei=0;
  for(i=S[j]; i<S[j+1];i++){num+=N[i]; wei+=W[i];}
  a=num/pow(sig,2)+1/pow(beta,2);
  cons=wei/pow(sig,2)+alpha/pow(beta,2);
  L[j]=rnorm(cons/a,pow(a,-0.5),tipo);
}
if(k>0) for(j=1; j <= k; j++){
  num=S[j+1]-S[j-1];
 if(num>2){
    if(L[j-1]!=L[j]){S[j]=fsaS2(S[j-1],L[j-1],L[j],S[j+1],N,W,sig,tipo);}
    else S[j]= S[j-1]+ sample(S[j+1]-S[j-1]-1);
    if(S[j]>=S[j+1]) cout<<"erro1"<<"S["<<j<<"]="<<S[j]<<" S["<<j+1<<"]="<<S[j+1]<<"\n";
    if(S[j]<=S[j-1]) cout<<"erro2";
  }
  num=0;
  wei=0;
  for(i=S[j]; i<S[j+1];i++){num+=N[i]; wei+=W[i];}
  a=num/pow(sig,2)+1/pow(beta,2);
  cons=wei/pow(sig,2)+alpha/pow(beta,2);
  L[j]=rnorm(cons/a,pow(a,-0.5),tipo);
}
for(j=0; j <= k; j++){
  num=0;
  wei=0;
  for(i=S[j]; i<S[j+1];i++){num+=N[i]; wei+=W[i];}
  a=num/pow(sig,2)+1/pow(beta,2);
  cons=wei/pow(sig,2)+alpha/pow(beta,2);
  L[j]=rnorm(cons/a,pow(a,-0.5),tipo);
}
mov=movimento(k);
u2=0;
num=0;wei=0;
num1=0;wei1=0;
if(mov!=3) TOT[1+mov]++;
if(mov==1){
```

```
    sl =sample((Amax-1));
    pos=0;
    while(u2 < k) {
      u2 = 0;
      sl =sample((Amax-1));
      pos = 0;
      for(i=1;i<=k;i++) {
        if(S[i] != sl) u2++;
        if(S[i] < sl) pos = i;
      }
    }
    for(i=S[pos]; i<S[pos+1];i++){num+=N[i];wei+=W[i];}
    for(i=S[pos]; i<sl;i++){num1+=N[i];wei1+=W[i];}
    h2 = L[pos];
    if(h2!=0){
     cons = (1/un(11+(tipo-1)*5) - 1);
      h1 = (S[pos+1]-S[pos])*h2/((S[pos+1]-sl)/cons+sl-S[pos]);
      h3 = h1/cons;
      ac=acei2(double(S[pos+1]-S[pos]),double(sl-
S[pos]),num1,num,wei1,wei,k,h1,h2,h3,int(sig),alpha,beta);
      if(ac>=0) ac=1;
      else {if(ac<-700) ac=0; else ac=exp(ac);}
      u=un(14+(tipo-1)*5);
      if(u <= ac) {
        increase_i(S,sl,pos + 1,kmax +2);
        L[pos] = h1;
        increase(L,h3,pos + 1,kmax+1);
        k = k + 1;
        TOT[mov-1]++;
      }
    }
  }
  if(mov==2){
    pos = sample(k);
    for(i=S[pos-1]; i<S[pos+1];i++){num+=N[i];wei+=W[i];}
    for(i=S[pos-1]; i<S[pos];i++){num1+=N[i];wei1+=W[i];}
    h1 = L[pos-1];
    h3 = L[pos];
    h2 = ((S[pos]-S[pos-1]) * h1 + (S[pos+1]-S[pos]) * h3)/(S[pos+1]-S[pos-1]);
    if(h2!=0){
    ac=acei_i2(S[pos+1]-S[pos-1],S[pos]-S[pos-
1],num1,num,wei1,wei,k,h1,h2,h3,sig,alpha,beta);
      if(ac<=0) ac=1;
      else {if(ac>700) ac=0; else ac = exp(-ac);}
      u = un(15+(tipo-1)*5);
      if(u <= ac) {
         k = k - 1;
         decrease_i(S,pos,kmax +2);
         L[pos-1] = h2;
         decrease(L,pos + 1,kmax+1);
         TOT[mov-1]++;
      }
    }
  }
  }
```