*Rev.R.Acad.Cienc.Exact.Fis.Nat. (Esp)*
*Vol. 93, N.º 3, pp 375-380, 1999*
*Monográfico: Métodos Bayesianos en las Ciencias*

# SPECIFYING AND SIMULATING COMPLEX MODELS USING BASSIST

**(Bayesian probability models/MCMC/simulation tools/nonparametric models/intensity models**

HANNU TT. TOIVONEN[1,3], HEIKKI MANNILA[2,3], MARKO SALMENKIVI[1,2] AND KARRI-PEKKA LAAKSO[2]

[1] Rolf Nevanlinna Institute, University of Helsinki, Finland.

[2] Department of Computer Science, University of Helsinki, Finland.

[3] Nokia Research Center, Finland. hannu.tt.toivonen@nokia.com

## ABSTRACT

Bassist is a simulation tool for Bayesian analysis. Given a high-level specification of a full probability model, Bassist generates a simulator for sampling from the joint posterior distribution of the model parameters and data. The Bassist simulator uses Markov chain Monte Carlo (MCMC) techniques, especially the Metropolis-Hastings-Green method.

General purpose MCMC simulation software has been almost non-existent, except for the BUGS system. Simulator programs have been written specifically for each model, either from scratch or by reusing previous code. In either case, the task is tedious and error-prone. Bassist aims at helping modelers at implementing and analyzing their models.

A difference between BUGS and Bassist is that Bassist is developed especially for the analysis of nonparametric models: the system supports the use of piecewise constant functions with a varying number of pieces. Moreover, Bassist supports the modeling of event sequences and intensity models, especially with piecewise constant functions. This makes the tool quite useful for modeling event data ubiquitous in epidemiology, fault management, and quality control.

The model specification language of Bassist is declarative, the design of the language emphasizes clarity in the model description by separating between the model specification, the simulation details, and the particulars of the data. Within the Bassist system, the model specifications are first compiled into a C++ program, which then reads the data and performs the actual simulation.

Bassist has been used in paleoecological reconstruction tasks, in epidemiology, and in the modeling of biodiversity. A limited, public version of Bassist is available at *www.rni.helsinki.fi/cs/bassist.*

## RESUMEN

### Especificación y simulación de modelos complejos utilizando Bassist

Bassist es una herramienta de simulación en análisis bayesiano. Dado un modelo probabilístico complejo, Bassist genera un simulador para obtener muestras de la distribución conjunta de los parámetros de modelos y de los datos. El simulador Bassist utiliza técnicas de cadenas markovianas de Monte Carlo (MCMC), especialmente las asociadas al método de Metropolis-Hastings.

El único programa general disponible para simulación por MCMC es BUGS. Los programas de simulación suelen escribirse específicamente para cada modelo, un trabajo tedioso en la que es fácil cometer errores. Bassist se propone facilitar la tarea del sieño y análisis de modelos probabilísticos complejos. Una diferencia entre BUGS y Bassist es que Bassist está específicamente desarrollado para el análisis de modelos no paramétricos: el sistema soporta el uso de funciones constantes a trozos con un número variable de elementos, lo que le hace especialmente apropiado en epidemiología y en control de calidad. Bassist ha sido utilizado en paleontología, epidemiología y modelización de la biodiversidad. Una versión limitada de acceso público de Bassist es accesible en la red en *www.rni.helsinki.fi/cs/bassist.*

## 1. INTRODUCTION

Bassist is a system for the analysis of Bayesian statistical models. It compiles a high-level specification of a full probability model to an executable simulator: given data, the simulator generates an approximate sample from the posterior distribution of the model parameters.

Models are specified using the Bassist language. At the basic level, a model consists of a number of *variables*

such as model and hyper parameters, covariates, and the data. The joint density of a model is specified as a product of densities of all variables in the model. These specified densities may be conditional on other variables.

More formally, a model is defined by a directed acyclic dependency graph where the set of nodes is the set $V$ of variables. The model specification is completed by defining the conditional probability $Pr(v \mid pred(v))$ for all $v \in V$, where $pred(v)$ is the set of immediate predecessors of variable $v$. The joint probability distribution of the model is then $\Pi_{v \in V} Pr(v \mid pred(v))$.

In Bayesian analysis some variables are given as the observed data, and the task is to compute the (posterior) distribution of the remaining variables. Bassist applies MCMC, in particular the Metropolis-Hastings-Green methods, for solving this task: the MCMC simulator generated by Bassist outputs a sample that approximates the desired posterior distribution.

Bassist is domain independent, and it aims at being a general purpose tool for Bayesian analysis. Thanks to the flexibility of the Metropolis-Hastings method, models with arbitrary probability distributions can be simulated, at least in principle, as long as the user is able to specify the density functions as C++ functions, for instance.

This paper is organized as follows. We first describe the goals and design of the Bassist system in Section 2 and the main features of the system architecture in Section 3. In Section 4 we give several examples that illustrate the use of Bassist in practical modeling tasks.

## 2. BASSIST GOALS AND DESIGN

We aim at a system and a modeling language that (1) is easy and natural to use; (2) contains a useful set of basic features; (3) does not require complex programming for features not directly supported; (4) allows easy modification of models, and (5) runs in a reasonable time. In contrast to BUGS (Spiegelhalter et al, 1996), a special emphasis is given at nonparametric and intensity models. Using Bassist requires, of course, a good understanding of statistical modeling and MCMC simulation.

More direct goals for Bassist are set by its actual applications in joint research with modelers. Preliminary versions of Bassist have been used in modeling tasks, e.g., in epidemiology, paleoecology, and biodiversity.

Bassist has been designed to support structured and clear model specifications. The Bassist language is declarative, and it separates the model specification from the simulation details and from the particulars of the data. In the following, we point out some important design decisions behind Bassist.

**Structured models.** Bassist supports user-defined records as a means of structuring data. A number of similar objects, e.g., patients in an epidemiological model, are handled by defining a record type for them and expressing patients as instances of that record type.

References between record instances allow expressing interdependencies between objects. For instance, in epidemiology, a child record might refer to a family record for a frailty parameter common to the whole family.

**Distributions.** In addition to standard library distributions, the user can specify arbitrary distributions for variables simply by defining the density function of the distribution in C++.

The user must also specify the proposal method for the Metropolis-Hastings method, and the user remains responsible for the convergence.

**Piecewise constant functions.** Bassist contains special constructs for defining nonparametric piecewise constant functions. Distributions are given for the number of pieces, the jump points, and the levels of the pieces in the model specification. All these parameters are simulated automatically according to the specified distributions.

**Event sequences and intensity models.** There are special primitives also for the specification and use of event sequences in models. Unobserved events can also be modeled within Bassist.

Intensity models can be defined: the intensity of an event sequence can be defined as piecewise constant function. Likelihoods are automatically computed for such intensities.

**Separation of model, data, and simulation.** The model specification only fixes the model, i.e., the variables and their dependencies. The modeling language makes no distinction between parameters (simulated variables) and data (variables with given, fixed values). The data files are specified at run-time, and they first define which variables are data and which are parameters. The number of record instances may also depend on the actual data given to the simulator. This flexible organization is especially useful when there is no clear data-parameter separation, e.g., when some data is missing.

The user can, and sometimes must, specify the proposal distributions for the parameters of the model. The user can request any library distribution to be used as the proposal distribution, including the random walk case.

Issues concerning a single simulation run are given as parameters to the simulator program at run time. These parameters include the data files, the number of sweeps and burn-in sweeps, and the sampling frequency.

## 3. BASSIST ARCHITECTURE

Bassist itself is a compiler: given a model specification, it produces an executable simulator for the given model. In more detail, the Bassist compiler generates a number of model specific C++ files, compiles them with a standard C++ compiler and links the results with the generic, model-independent Bassist files to an executable simulation program.

Bassist thus consists of a compiler for the modeling language and of libraries for the generic parts of simulators. These libraries obviously contain classes for the library distributions, but also a large fraction of simulation structures and algorithms and some simple data manipulation tools.

The system is implemented with standard C++, yacc, and lex tools, and simulation programs produced by the compiler are in standard C++.

## 4. EXAMPLES

We now give three examples to illustrate the use of Bassist in practical modeling tasks. The first example is a simple hierarchical model. We then continue with two more subtle cases with nonparametric intensity functions and hierarchical frailty structures.

### 4.1. Example: pumps

The following example was presented by George et al. (1993) and was also used by the BUGS group (Spiegelhalter et al., 1996) as a simple example of a conjugate gamma-Poisson hierarchical model. The example concerns with the operation of 10 power plant pumps. The lengths of the operation times ($t_i$) and the corresponding numbers of failures ($x_i$) for each pump are given in the data. The failure rate $\theta_i$ for pump $i$ is given a gamma prior distribution. Prior distributions for the hyperparameters $\alpha$ and $\beta$ are *Exp*(1) and *Gamma*(0.1,1), respectively.

Following is a Bassist definition of the model.

```
var alpha ~ exp(1);
var beta ~ gamma(0.1,1);

record pump {
    var theta ~ gamma(alpha,beta);
    var t;
    var lambda = theta*t;
    var x ~ poisson(lambda);
};
```

The variables in the model are indicated by the reserved word **var**. The record *pump* ties together all the parameters associated only with a single pump. These parameters are defined inside the record. The hyperparameters *alpha* and *beta*, which are common to all pumps, are defined separately but can be referred to anywhere in the model description. The variable *lambda* is not a random variable but is determined functionally from the values of *theta* and *t*.

### 4.2. Example: rats

The next animal carcinogenesis data was described by Mantel and Ciminera (1979) and it was also used by Clayton (1991). Following Clayton in broad outline, we model the problem using the Bassist description language.

In the experiment of Mantel and Ciminera, 50 litters of rats were followed. One rat from each litter was treated with carcinogen, while two others were chosen for controls. Clayton illustrates the Bayesian frailty modeling with this data using frailty factors shared by the members of the same litter.

The model is of the form:

$$\lambda_i(t) = \lambda_0(t)\ \xi_l \exp(\beta^T z_i),$$

where the frailty factors $\xi_l$ were assumed to be i.i.d. gamma variables with mean 1 and variance $\gamma$, i.e., $\xi_l \sim$ Gamma($1/\gamma$, $1/\gamma$). The hyperparameter $\gamma$ was given a prior distribution and was estimated from the data. The parameter $\beta$ describes the influence of the treatment and $z_i$ is the indicator of belonging to the treatment group.

A piece-wise constant function is used to represent the baseline hazard $\lambda_0$. In contrast to Clayton's model, in our model the change points of the function are not determined in advance. We also let the number of pieces in the model be one of the parameters, using the ideas of variable dimension MCMC (Green 1995). The intensity values in each piece are assumed to be i.i.d. The model description is as follows.

```
var g ~ gamma(0.00001,0.00001);
var beta ~ unif (-10,10);

record litter {
    var frailty ~ gamma(1/g,1/g);
};

record rat {
    ref litter;
    var time ~ Poisson process(
               lambda(time,carcinogen,
               litter.frailty));
    var carcinogen;
};

function lambda(time,carcinogen,frailty) {
    return lambda0(time) * frailty
    * exp(beta*carcinogen);
};

var pwcf lambda0(time) {
```

```
    pieces ~ geom(0.5);
    levels ~ unif(0.005,0.1);
    jump_points ~ unif(0,104);
};
```

The variables in this example are either global parameters (*beta* and hyperparameter *g*), parameters associated with a rat (*time, carcinogen*) or parameters associated with a litter (*frailty*).

Consequently, records for rats and litters need to be defined. It is also necessary to know which litter each rat belongs to; references (indicated by the keyword **ref** in record rat) are used to express connections between a rat and the corresponding litter.

The specification of the Poisson intensity *lambda* for the death time of a rat is given as a separate function definition. As a function argument, the value of the frailty factor of the corresponding litter is needed.

Here the function lambda contains a single statement to return the value of *lambda* determined by the commom time-dependent baseline hazard *lambda0*, the frailty factor of the litter of the rat, the covariate value *carcinogen* and the variable *beta* expressing the influence of the carcinogen treatment.

The piece-wise constant function (**pwcf**) *lambda0* is defined by giving the name and arguments of the function and specifying the distributions for the components of the function: the number of pieces, function values in each piece (*levels*) and the jump points.

### 4.3. Example: Ear infections in children

As the final example, we take a brief look at a relatively complex modeling task, occurrences of middle ear infections (acute otitis media). For more details of the problem background, data, modeling, computation and results, see Eerola et al. 1998.

In this study, 329 children were followed from the age of 2 to 24 months in Tampere region, Finland. In addition to the occurrences of ear infection episodes, changes in several risk factors were recorded during the follow-up. A lag time of 30 days was used to distinguish between episodes.

For simplicity, we only consider here the effect of one risk factor, daycare attendance. We distinguish between the risk factor for the first and recurrent episodes following the parameterisation in Eerola (1989).

The intensity for individual $i$ is of the form:

$$\lambda_i(t) = \lambda_0(t)\ \xi_i\ Y_i(t)\ \gamma_i(t)\ \alpha_i(t)\ \gamma_i^*(t),$$

where $\lambda_0$ is the age-dependent baseline hazard, $\xi_i$ is an individual frailty factor and $Y_i(t)$ an indicator function of being at risk at time $t$. The function $\alpha_i(t)$ separates between the first and recurrent infections: before the first infection it is defined to be 1, after that it is a constant to be estimated.

The time-dependent functions $\gamma_i(t)$ and $\gamma_i^*(t)$ indicate the influence of daycare attendance. Their values are defined to be 1 before daycare attendance, otherwise they are parameters to be estimated. The values <1 correspond to protective effect and the values >1 to increased risk. Before the first infection the influence of daycare attendance only consist of the value of $\gamma_i(t)$. After the first infection the additional coefficient $\gamma_i^*(t)$ is included and thus the total effect is $\gamma_i(t)/\gamma_i^*(t)$. We ignore the seasonal effect in this example.

We now describe the model in the Bassist language; the variable *cdaycare* in the following description corresponds to $\gamma_i$ above and *cdaycare_star* similarly to $\gamma_i^*$.

```
var cdaycare ~ unif(0.0001,10);
var cdaycare_star ~ unif(0.0001,10);
var alpha ~ unif(0.001,10);
var eta ~ unif(0.0001,100);

record child {
    var frailty ~ gamma(eta,eta);
    event sequence of infection ~ Poisson
              process(lambda(
              infection.daycare,frailty));
};

event infection {
    ref child;
    var daycare;
};

function lambda(time, daycare, frailty) {
    float dc_multiplier = 1;
    float dc_star_multiplier = 1;
    if (daycare == 1) {
        dc_multiplier = cdaycare;
        dc_star_multiplier = cdaycare_star;
    };
    if (time < first(infections))
        return frailty * dc_multiplier
        * lambda0(time)
    else if (time < prev(infections)+1)
        return 0;
    else
        return frailty * alpha
        * dc_multiplier
        * dc_star_multiplier
        * lambda0(time);
};

var pwcf lambda0(time) {
    pieces ~ poisson(5);
    levels ~ unif(0.00001,0.15);
    jump_points ~ unif(1.5,24.89);
};
```

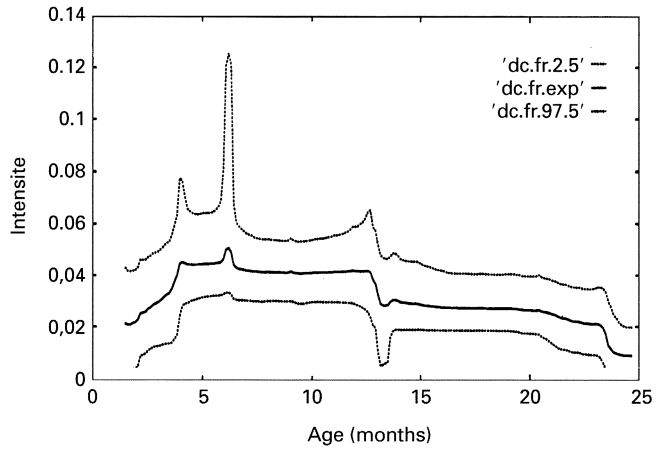


**Figure 1.** Posterior baseline intensity curves $\lambda_0(t)$ (mean and 95 % credible intervals).

The essential additional feature in this example compared with the previous one is the introduction of event sequences. A sequence of infections is associated with each child and Poisson intensity for the sequence is specified. The **event** type infection is defined in a separate record.

As in the previous example the Poisson intensity *lambda* is defined in a separate function. The argument *daycare* is tested to set the multipliers corresponding to daycare attendance to the right values. The intensity value at a given time instant also depends on the instant of the first infection, which is tested in the next if-statement. The lag time of 30 days is taken account of in the last if-statement by testing the distance from the previous infection. The functions **first** and **prev** used in the tests are special library functions of the Bassist system returning the first or previous event of the given sequence.

Finally, we show some results from the simulation of the posterior distribution of the described model and data. Figure 1 shows the posterior expectation and the 95 % credible intervals of the piecewise constant baseline function *lambda0*. The risk seems to increase sharply at the age of five months, maintain relatively stable until the age of 12 months, and decrease again after that.

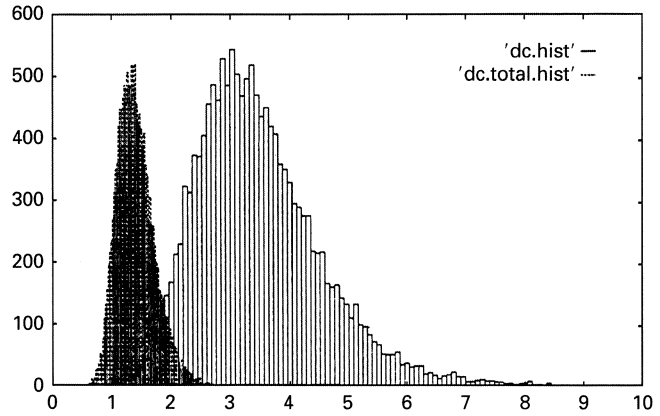


**Figure 2.** Posterior distributions of $\gamma_i$ (right) and the product $\gamma_i\gamma_i^*$ (left). Distributions are not in the same scale.

The posterior distributions of the parameters $\gamma_i$ and $\gamma_i^*$ are shown in Figure 2. Daycare attendance significantly increases the risk for the children who had no infections before the daycare attendance. After the first infection the influence seems to be less clear, indicated by the lower values of the product $\gamma_i\gamma_i^*$.

## 5. CONCLUSION

We described Bassist, a simulation tool for Bayesian analysis. Bassist generates MCMC simulators from high level model definitions. Bassist is currently under development. Preliminary versions have been used, e.g., in epidemiological modeling tasks where event sequences and intensity functions are material. Experiences have been encouraging: availability of an automatic simulator generator can cut down the programming effort significantly.

A publicly available version of Bassist can be found in *www.rni.helsinki.fi/cs/bassist.* The public version does not yet offer support for nonparametric intensity functions and the modeling of event sequences.

## ACKNOWLEDGEMENTS

This research has been supported by the Academy of Finland. We are grateful to Dr. Mervi Eerola for co-operation in the ear infection example, and Prof. Elja Arjas, Dr. Kari Auranen, and Mr. Tommi Härkänen for commenting on a preliminary version of this paper.

## REFERENCES

1. Clayton, D. G. (1991). A Monte Carlo Method for Bayesian Inference in Frailty Models. *Biometrics* **47,** 467-485.
2. Eerola, M. (1989). Repeatable events in event-history analysis; the effect of childhood separation on future mental health. *Bull. Int. Statist. Inst.,* **LIII**(1), 213-225.
3. Eerola, M., Mannila, H. and Salmenkivi, M. (1998). Frailty factors and time-dependent hazards in modelling ear infections in children Using BASSIST. In: *COMPSTAT'98 Proceedings in Computational Statistics,* 287-292. Physica-Verlag.
4. George, E., Makov, U. and Smith, A. (1993). Conjugate likelihood distributions. *Scandinvian Journal of Statistics* **20,** 147-156.

**5.** Green, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrica* **82,** 4, 711-732.

**6.** Mantel, N. and Ciminera, J. L. (1979). Mantel-Haenszel analysis of litter-matched time-to-response data with modifications for recovery of interlitter information. *Cancer Research* **37,** 3863-3868.

**7.** Spiegelhalter, D., Thomas, A., Best, N. and Gilks, W. (1996). *BUGS, Bayesian inference Using Gibbs Sampling Manual, Version 0.50.* MRC Biostatistics Unit, Institute of Public Health, Cambridge.