(-) -, Number -, pp. 1–21

Gibbs Sampling with JAGS: Behind the Scenes

Gianpaolo Coro*

Abstract. Gibbs sampling is a Bayesian inference technique that is used in various scientific domains to generate samples from a certain posterior probability density function, given experimental data. Several software implementations of Gibbs sampling exist, which generally adopt very different approaches, because it is not easy to make a Gibbs sampling implementation exactly correspond to the theoretical approach. In particular, these implementations may use different approximation algorithms to find solutions to sub-steps of the Gibbs sampling process. Scientists working in different domains often use Gibbs sampling software without knowing the details of the implementation. Nevertheless, it is our experience that understanding the implementation can be crucial to enhance the performance of a model, because a software configuration conceived to help the underlying implementation may end in better approximation of the estimated probabilities functions. JAGS (Just Another Gibbs Sampler) is a widely used open-source implementation of Gibbs sampling. Its installation and user's guide are accurate, but do not indicate how the software really implements Gibbs sampling and it is not easy to infer this information from the source code. The aim of this paper is to give a high-level overview of the JAGS algorithms and its extensions that implement Gibbs sampling. Our target reader is a scientist who may want to understand the basic concepts underlying Bayesian inference and Gibbs sampling and who want to be aware of what happens behind the scenes when building a model.

MSC 2010 subject classifications: Gibbs sampling, JAGS, Bayesian Inference, Markov Chains.

1 Introduction

Gibbs sampling is a technique for statistical inference that is used in several scientific domains. It produces samples from a posterior distribution conditioned on the observed data and thus allows to obtain final estimates of a model parameters. Gibbs sampling is used in many scientific domains, where prior knowledge is combined with observed data. For example, it has been used in marine science to estimate the health status of a fisheries stock (Froese et al., 2014), in computational biology to decipher DNA sequences (Lawrence et al., 1993), in financial applications (Eraker, 2001) and in medicine (Gilks et al., 1993).

Gibbs sampling can be implemented in several ways and the available software distributions (e.g. Best et al. (1995), Thompson et al. (2003), Spiegelhalter et al. (1996), Anders (2013)) generally adopt different approaches in their implementations. Further, it is not easy to make the theory correspond to one practical implementation, and realisations in many cases mix additional algorithms with the Gibbs sampling process.

*Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" ISTI – CNR
via Moruzzi 1, 56124, Pisa, Italy coro@isti.cnr.it

 DOI: -

As a result, a Gibbs sampling software if often used as a black box after a model has been configured. Instead, using a Gibbs sampling software being aware of what happens behind the scenes is important, especially when models become complex, have many parameters or need fine-tuning. Our assumption, supported by experience, is that understanding how a Gibbs sampling software works may result in better models (Coro, 2014; Froese et al., 2016; Rosenberg et al., 2014), which is also supported by general approaches to statistical inference (Breiman et al., 2001).

JAGS (Just Another Gibbs Sampler, Plummer (2003), Plummer (2011)) is an open-source implementation of Gibbs sampling that has been used in several experiments (Froese, 2013; Juntunen et al., 2014; Thorson et al., 2014). JAGS extends the standard Gibbs sampling process, using further algorithms to sample from the target posterior distribution. This software is endowed with a complete installation and user's documentation, and is also usable in the R programming language where it has accumulated a large number of dependencies. Unfortunately, documentation gives no indication about JAGS internal working. Books and papers exist that focus on the essentials of modelling using Gibbs sampling (Depaoli et al., 2016; Lunn et al., 2012; Ntzoufras, 2011) and sometimes also explain background mechanisms of particular implementations (e.g. WINBUGS, Lunn et al. (2000)). Instead, in the case of JAGS this background information is scattered across manuals, tutorials or dozens of R packages (e.g. Su and Yajima (2012), Plummer (2013)) that make it difficult to concretely understand the underlying mechanisms. Further, although the JAGS source code is accessible, concretely understanding how JAGS implements Gibbs sampling at a higher level than C++ coding is not easy, also because JAGS users are usually R programmers or are not computer scientists. For example, this may be of interest to a statistician who wished to understand the general rationale behind the implementation, e.g. (i) which representation corresponds to the configured model, (ii) how the software samples from functions with very complex forms, (iii) if the implementation is able to manage complex models containing many prior distributions and few likelihoods. Understanding these details is important, for example to implement penalty functions that avoid the variables estimations to fall in some ranges of values (Froese et al., 2016; Plummer, 2008).

The scope of this paper is to explain Gibbs sampling referring to the implementations in the JAGS software. We will give an introductive and high level explanation that is accessible also to people with basic statistical knowledge of probability theory. Nevertheless, the basic definitions of probability and probability density are assumed to be known. We start from the basic concepts behind Bayesian Inference and then we introduce Gibbs sampling, JAGS and the graphical hierarchical models used by this software. The aim is to make a JAGS user more aware of what happens behind the scenes, thus making modelling easier and opening the way to more advanced usages.

This paper is organized as follows: Section 2 presents the basic principles of Bayesian Inference to introduce non-expert readers to key concepts used by Gibbs sampling. Section 3 introduces the JAGS software and its basic modelling principles. Section 3.2, introduces the probabilistic graphical models JAGS is based on. Section 4 reports the definition of Markov chain. Section 5 explains Gibbs sampling based on the previously introduced concepts. Section 6 gives a practical example in the R programming language

of a JAGS model and explains how this implements the concepts previously explained. Finally, Section 7 draws the conclusions.

2 Bayesian Inference

In order to define Bayesian Inference, we first define *posterior probability distributions* (or *posterior probability densities*). The posterior probability distribution of a random event, given a certain set of data, is defined as the *conditional* probability density that is assigned to the event after relevant evidence has been taken into account. In other words, if there is experimental evidence of a certain probabilistic phenomenon, the posterior probability distribution relates this evidence with some random variables. For example, in the case of a random variable x having a Gaussian distribution $Norm(\mu, \sigma)$ (with μ and σ being the mean and the standard deviation respectively, both unknown and to be estimated), the posterior probability distribution would be indicated as $p(\mu, \sigma|x)$. This distribution allows to calculate the probability of a pair (μ, σ) given a certain value of x. Generally, given a set of experimental evidences $\bar{y} = \{y_1, y_2, .., y_n\}$ linked by a probabilistic relation to a set of random variables $\bar{\theta} = \{\theta_1, \theta_2, .., \theta_m\}$, the posterior probability density is indicated as $p(\bar{\theta}|\bar{y})$. Inferring the analytical form of a posterior probability when only experimental evidence is available can be difficult, but techniques exist to approximate the samples generated by these functions. As a reminder, if x is a continuous random variable with a $p(x)$ probability density associated, the probability of x falling in a given numeric interval $[x_1, x_2]$ can be obtained from the probability density as

$$P[x_1 \leq x \leq x_2] = \int_{x_1}^{x_2} p(x)\, dx \tag{1}$$

In this paper, we will use P to denote the probability of a certain event (e.g. x falling in a range or equal to a certain value) and p a probability density.

Another important concept in Bayesian Inference is the *likelihood* of the model to the data. This is a dual concept with respect to the posterior probability distribution. In fact, likelihood is defined as the probability density associated to the evidence $\bar{y}$ given the variables $\bar{\theta}$, and is indicated as $p(\bar{y}|\bar{\theta})$.

The likelihood and the posterior probability distribution are related through the Bayes' rule:

$$p(\bar{\theta}|\bar{y}) = \frac{p(\bar{y}|\bar{\theta})p(\bar{\theta})}{p(\bar{y})} \tag{2}$$

where $p(\bar{y})$ is the probability density (named *marginal density* of the sample data or *marginal likelihood*) associated to the sample data without considering the random variables. The term $p(\bar{\theta})$ is the *prior* probability distribution and indicates an *a priori* estimate of the distribution of the parameters $\bar{\theta}$ (Chib, 1995).

The Bayes rule for probability densities is conceptually different from The Bayes' rule for probabilities although their form is similar. In the case of probabilities, the rule

is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

where A and B are two probabilistic phenomena. The difference is that the rule for the distributions is derived from the definition of joint probability density, whereas the rule for the probabilities comes from the axioms of Probability.

Bayesian Inference is defined as a process based on the Bayes' rule for probability distributions, that estimates a posterior probability density using prior knowledge about the parameters (priors) and updates this estimate by means of likelihoods, as long as new evidence (from here on referred as "real data" or "real observations") is acquired. Thus, Bayesian Inference factorizes prior knowledge about parameters and likelihoods, using likelihoods as a means to relate the parameters to the real observations. Alternative methods simulate the posterior probability density or the likelihood functions directly (Huang et al., 2001; Stefano, 1998). Usually, Bayesian Inference is used to calculate the *best* parameters estimates given the observed data as the parameters that maximize the posterior probability distribution. Indeed, the best estimates depend on the form of the distribution. Usually, parameters belonging to the *central tendency* of the posterior probability density are taken, other times the maximum is selected (Maximum a Posteriori or MAP). Generally, the selection process should avoid involving outliers in the input data and biases in the estimate (Huber, 2011).

Once the best estimates of the parameters has been found, it can be used in many ways, for example to simulate a function (e.g. the best estimate of the coefficients of a linear combination) or to predict new data (Berger, 1985).

3 JAGS

JAGS (Just another Gibbs sampler) is a program developed by Martyn Plummer (Plummer, 2003, 2011) that implements Bayesian inference based on Gibbs sampling. To this aim, it implements a Markov Chain Monte Carlo (MCMC) approach. MCMC is a generic term indicating an algorithm that samples from probability distributions (sampler) by constructing a Markov chain (Section 4) that has the desired probability density as its equilibrium (or ergodic) distribution. MCMCs are often combined with Monte Carlo Integration (Robert and Casella, 1999) (Section 5.5) on the generated samples to estimate the variables best values. In this section, we will define the MCMC approach used by JAGS and will clarify its relationship with Gibbs sampling.

JAGS uses *hierarchical* models to instruct the sampler. Models are written using a dialect of the BUGS programming language. BUGS is a *declarative* language that allows a programmer to specify the known relations among the variables. In the JAGS modelling language the order in which the relations are specified is not important, because the constraints are automatically inferred from the variables declarations. The declarative approach used by JAGS is different from the one used by common imperative languages (e.g. R, C, Fortran, Pascal etc.), in which an algorithm is built as a sequence of steps "thought" from a computer's point of view. In JAGS, the relations

between the variables are specified in terms of probabilistic or deterministic functions. Further, likelihoods are identified as those functions that define a variable for which real observations are available.

Before running its MCMC sampler, JAGS analyses the distributions definitions and applies the most appropriate strategy to produce samples from the posterior probability densities defined in the model. JAGS tries to produce samples that come more and more from a density function that is the posterior density function the user is searching for, i.e. the optimal model given the data.

The general approach of JAGS to produce samples is compliant with the Gibbs sampling theory (Section 4), but JAGS also adds other techniques that enforce (or in some cases substitute) the standard Gibbs sampling approach. For example, the Metropolis–Hastings algorithm (Chib and Greenberg, 1995), the Slice sampling (Neal, 2003) and the Adaptive Rejection sampling (Gilks et al., 1995) algorithms may be used in some situations, but they are not strictly part of Gibbs sampling. In Section 5 we explain how these may intervene during the sampling process.

JAGS is an alternative to other tools that are based on BUGS, like WINBUGS (Espino-Hernandez, 2010; Ntzoufras, 2011) and OpenBUGS (Spiegelhalter et al., 2007). The main aim of JAGS is to provide a multi-platform open-source framework that is easily expandable with new algorithms and is also open to criticize graphical models. JAGS is endowed with a wrapper for the R language (rjags) (Plummer, 2011) that runs BUGS models, extends this language and retrieves the output as an R object, that allows for further processing (Hojsgaard, 2013).

3.1 BUGS models

The best way to explain a BUGS model is to report an example. A simple linear regression in BUGS looks like the following:

```
mu <- alpha + beta * (x - x.bar)
Y ~ dnorm(mu, tau)
x.bar <- mean(x)
alpha ~ dnorm(0.0, 1.0E-4)
beta ~ dnorm(0.0, 1.0E-4)
tau ~ dgamma(1.0E-3, 1.0E-3)
```

This code declares the dependencies among the variables in terms either of stochastic or of deterministic relations. Unlike imperative languages, the order of the declarations is not important. BUGS treats them as they were the hypotheses of a theorem to be demonstrated. The BUGS engine inside JAGS checks for model's coherence before executing the inference process. In the example R code, the *mu* variable is a linear combination of other variables, whereas Y is a stochastic variable distributed like a Gaussian function that depends also on other two variables: *mu* and *tau*. Also, *alpha* and *beta* are distributed like a Gaussian, whereas *tau* follows a Gamma distribution. In mathematical

terms, the second relation corresponds to the following analytical function:

$$G(Y, mu, tau) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(Y-mu)^2 tau}{2}} \tag{4}$$

(note that $tau = \frac{1}{\sigma^2}$ in JAGS).

At the initialisation phase of the model, some variables may have real data (i.e. observations, measures etc.) associated. In this case, the JAGS engine interprets its corresponding probability distribution as a likelihood. Referring to the example, if real observations were available for Y, JAGS would interpret $Y \sim dnorm(mu, tau)$ as a likelihood. The other distributions would be treated either as *priors* or as *conditional* probability distributions. The deterministic/analytical functions in the model act as definitions to simplify the syntax.

With respect to the BUGS language, JAGS also adds a number of probability functions to be used for variables definitions and conditional instructions (Plummer, 2015).

3.2 Graphical models

Graphical models are the basic concepts underlying JAGS models. The fundamental object in a graphical model is a node representing a variable in the model (either observed or unobserved). Nodes possibly have children and parents, and a dimension attribute (Bishop and Nasrabadi, 2006). JAGS allows defining nodes that represent stochastic parameters (*Stochastic* nodes), deterministic parameters (*Logical* nodes) and constants (*Constant* nodes). The relations among the nodes are automatically traced based on the names of the variables. Parameters depending on other parameters are seen as having these as "parents". Further, JAGS allows defining *Array* nodes that represent containers of parameters entirely related to other variables and *Subset* nodes related to other nodes by subscripting.

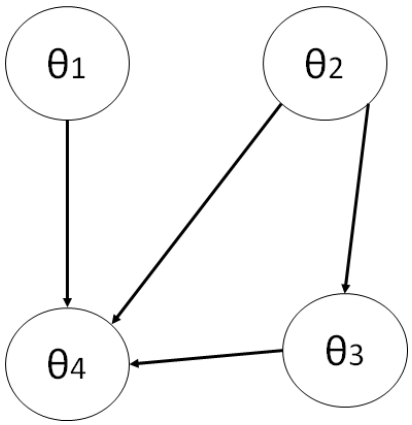

Figure 1: A probabilistic graphical model for 4 variables.

Generally, a graphical model is a probabilistic model in which a graph defines the conditional dependencies between the random variables. The graph gives a compact representation from which the independent and the conditioned variables are immediately evident. One example is in Figure 1, where a graph represents the relations between four variables: $\{\theta_1, \theta_2, \theta_3, \theta_4\}$.

The joint probability distribution for the variables in Figure 1 is

$$p(\theta_1, \theta_2, \theta_3, \theta_4) = p(\theta_1)p(\theta_2)p(\theta_4|\theta_1, \theta_2, \theta_3)p(\theta_3|\theta_2) \tag{5}$$

This formula multiplies all the conditional distributions of the variables. Based on the observation data associated to the variables, JAGS knows which of them must be interpreted as likelihoods and which as priors or conditional distributions. In Section 5 we explain how Gibbs sampling starts from this joint distribution to sample the posterior probability density.

By abstracting the graphical model of the figure, a general form of the joint distribution associated to a graphical model representing the variables $\{\theta_1, \theta_2, \ldots, \theta_n\}$ is

$$p(\theta_1, \theta_2, \ldots, \theta_n) = \prod_{i=1}^{n} p(\theta_i|par_i) \tag{6}$$

where par_i is the set of parents of the θ_i node.

Hierarchical models are a special case of graphical models, in which the edges have a *causal interpretation*, established by the hierarchical dependencies between the nodes (Clauset et al., 2008). In hierarchical models, conditional independence comes out directly from the hierarchy. JAGS models are hierarchical models.

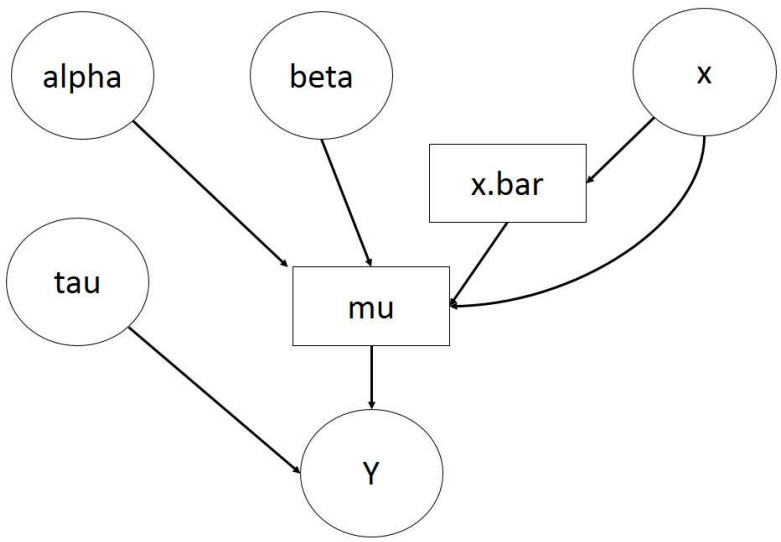

Figure 2: A probabilistic hierarchical model for the linear regression example of Section 3. The squares indicate deterministic functions, whereas the circles indicate probabilistic distributions.

The hierarchical model of the linear regression example of Section 3.1 (assuming x to be a random variable) is reported in Figure 2. The joint distribution associated to the model in Figure 2 is

$$\begin{aligned} p(alpha, beta, tau, mu, x.bar, Y, x) = p(alpha)p(beta)p(tau)p(x)p(x.bar|X) \\ p(mu|alpha, beta, x.bar, x)p(Y|tau, mu) \end{aligned} \tag{7}$$

4 Markov chains

A Markov chain is a model for a sequence of random variables. It is suited to manage stochastic processes that require to reduce the used amount of memory. Markov chains are often used to model sequences of observations in time, especially when time is a discrete variable.

Consider a sequence of random variables $\theta_1, \theta_2, \ldots, \theta_n$ that can take values from the same finite alphabet $O = \{s_1, s_2, \ldots, s_m\}$. For example, this could be the case of the Dow-Jones industrial average. The change of this quantity in time can be modelled assuming that at several discrete time instants a different random variable θ_t can take a value among $O = \{Up, Down, Unchanged\}$.

Generally, the joint probability distribution of a sequence of random variables is

$$p(\theta_1, \theta_2, \ldots, \theta_n) = p(\theta_1) \prod_{i=2}^{n} p(\theta_i | \theta_{i-1} \theta_{i-2} \ldots \theta_1) \tag{8}$$

This formula models the fact that a variable θ_k depends on all the preceding variables in the sequence. For example, in the case of a sequence of 3 variables

$$p(\theta_1, \theta_2, \theta_3) = p(\theta_1) p(\theta_2 | \theta_1) p(\theta_3 | \theta_2 \theta_1) \tag{9}$$

A sequence of random variables is said to form a *first–order* Markov chain if the probability of a variable in the sequence is conditioned only by the preceding variable. This means the following equality

$$p(\theta_i | \theta_{i-1} \theta_{i-2} \ldots \theta_1) = p(\theta_i | \theta_{i-1}) \tag{10}$$

and the formula for the joint probability density becomes

$$p(\theta_1, \theta_2, \ldots, \theta_n) = p(\theta_1) \prod_{i=2}^{n} p(\theta_i | \theta_{i-1}) \tag{11}$$

This equation is also known as *first–order Markov assumption*. Markov models are particularly indicated to model phenomena characterized by a sequence of observations in time, where each observation depends on the preceding one. Often, it is necessary to model relations with the m preceding variables instead of the first, which requires to use mth–order Markov chains, whose definition is easily deducible from the first–order one. In Gibbs sampling, a first–order Markov assumption is usually assumed.

The probability densities $p(\theta_i | \theta_{i-1})$ are named *transition* probabilities. Since the θ_i variables can assume values from the $\{s_1, s_2, \ldots, s_m\}$ set, a chain of variables will be associated to a sequence of s_i values. These values are usually named the *states* of the Markov chain. For example, in the case of the Dow-Jones average, one sequence could be *Up, Down, Down, Up, Unchanged*.

The probability of state s_i to be the state at time t in the sequence can be calculated as the overall probability to pass from a preceding state s_j to the following state s_i. Indeed, in the most general setting s_j could be s_i itself, because the Markov chain could present consecutive repetitions of the same state in the sequence. The formula of this overall probability is

$$P_t(s_i) = \sum_{j=1}^{g} P_{t-1}(s_j) P_t(s_i | s_j) \tag{12}$$

This formula indicates that the probability of s_i to be the state at time t in the sequence, is given by all the possible ways to reach s_i from any preceding s_j state. The

formula also accounts for the probability that s_j was the previous state in the sequence. The transition probability from s_i to s_j can be different depending on the time instant.

In order to complete the set of concepts that allow understanding the role of Markov chains in Gibbs sampling, we will introduce some other definitions. In particular, a Markov chain is said to be *homogeneous* or *stationary* if the transition probability does not depend on time. This means that $P_t(s_i|s_j)$ is simply $P(s_i|s_j)$ and depends only on the pair of states.

A Markov chain is said to have reached an *invariant* probability $D(s_i)$ over the states, when this probability persists forever. In other words

$$D(s_i) = \sum_{j=1}^{g} D(s_j) P_t(s_i|s_j) \tag{13}$$

If the chain is also homogeneous, we can substitute P_t with P. A finite Markov chain has always at least one invariant distribution.

A Markov chain is *ergodic* if $P_t(s_i)$ converges to an invariant probability for $t \to \infty$. Further, this is required to happen regardless of the choice of initial probabilities $P_0(s_i)$. An ergodic Markov chain can have only one invariant probability, named *equilibrium* probability. Gibbs sampling searches for ergodic Markov chains that converge to an invariant probability, which is associated to the searched posterior probability distribution (Section 5.4).

A Markov chain is *aperiodic* if the return of a state in the chain can occur at irregular times.

A chain is *irreducible* if it is always possible to go from one state to any other state (not necessarily in one step).

A Markov chain is *recurrent* if, for any given state i, if the chain starts at i it will eventually return to i with probability 1. The chain is *positive recurrent* if the expected return time to state i is finite, otherwise it is *null* recurrent.

The *ergodic theorem* states that if a Markov chain is aperiodic, irreducible and positive recurrent then it is ergodic (Neal, 1993). This theorem is used to demonstrate that Gibbs sampling based on Markov chains can be ergodic under some conditions. The extension of the explanation above to the case of continuous variables and probability densities is intuitive. We leave further details about Markov chains, as well as the demonstration of the ergodic theorem, to more specific papers (Huang et al., 2001; Norris, 1998; Walsh, 2004).

5 Gibbs sampling

In this section, we use the concepts defined so far to explain Gibbs sampling.

The main aim of Gibbs sampling is to sample the posterior probability distribution

$$p(\bar{\theta}|\bar{y}) = \frac{p(\bar{y}|\bar{\theta})p(\bar{\theta})}{p(\bar{y})} \tag{14}$$

The general aim is that samples from this posterior distribution can be used to estimate the $\bar{\theta}$ model parameters given the real data. The main issue with this approach is that it is difficult to draw samples from the density function when it is not a standard statistical distribution. There are several techniques to generate samples from standard distributions, because their shape is well known (Chib, 1995; Lyle Gurrin and Ekstrom, 2013; Neal, 1993). Unfortunately, although it is possible to obtain an analytical form of the posterior function from a hierarchical model, its form is seldom standard.

Gibbs sampling solves this issue using Markov chains (Casella and George, 1992; Resnik and Hardisty, 2010). In particular, it does not directly sample from the complete function, but rather from conditional distributions of the θ_i variables given all the other variables (full conditionals), i.e. $p(\theta_i|\theta_1, \ldots, \theta_i - 1, \theta_i + 1, \ldots, \theta_n, \bar{y})$. Generally, this is an iterative procedure that generates a Markov chain of samples (the details will be explained in the next sections). It can be demonstrated that the higher the number of iterations the closer the samples to the posterior density. In other words, the Markov chain samples are ergodically convergent to the posterior density values.

5.1 Rationale

In order to understand how Gibbs sampling works, we explain (i) how the samples from the full conditionals are linked to those of the posterior probability density, (ii) how the analytical form of a full conditional is built, (iii) how a Markov chain to sample the full conditionals is generated.

According to the Bayes's rule, a posterior probability density of a set of variables can be written in the following way

$$p(\theta_1, \theta_2, \ldots, \theta_n|\bar{y}) = p(\theta_1|\theta_2, \ldots, \theta_n, \bar{y})p(\theta_2, \ldots, \theta_n|\bar{y}) \tag{15}$$

The same rule is valid also for the other variables. The first term at the right side is the *full conditional* of θ_1. This means that sampling each full conditional in turn, gives values that are proportional to the posterior distribution. Gibbs sampling uses this property and samples iteratively each full conditional of a variable, leaving the other variables at their preceding values in time. When a full conditional is sampled in this way, a new value for the conditioned variable is picked and then immediately used to sample the other variables that have not been sampled yet. A full conditional is usually easier to sample than the complete posterior density. Further, it can hopefully have the form of a standard distribution, which is unlikely to happen for the complete posterior density.

Gibbs sampling is divided in two parts: the first aims at obtaining analytical forms for each full conditional. The second generates samples iteratively with a Markov approach, where these samples tend towards the posterior density ones (Lam, 2013).

5.2 Obtaining analytical forms of full conditionals

The algorithm to obtain the analytical forms for the full conditionals goes through the following steps

1. write the complete formula of the posterior probability;
2. pick one parameter θ_i;
3. from the formula of the posterior probability, drop all the factors that do not depend on θ_i;
4. at this point, if the other parameters are fixed to their current values, a formula for the full conditional of θ_i is obtained;
5. use automatic simplification procedures to figure out if the conditional probability can be reduced to a known statistical distribution;
6. repeat step 2-5 for all the parameters.

This algorithm is also used by JAGS. In the case the relations came from a hierarchical model (e.g. in JAGS), the formula of the posterior probability density would be a multiplication of densities reflecting the hierarchy (Section 3.2).

The work of the algorithm can be shown with an example, where the analytical formula of the complete posterior probability is

$$p(\lambda_1, \lambda_2, \beta|\bar{y}, \bar{t}) = \prod_{i=1}^{2} \lambda_i^{(y_i-1)} e^{-(t_i+\beta)\lambda_i} \beta^9 e^{-20\beta} \tag{16}$$

where $\bar{y}$ and $\bar{t}$ are real observations and $\{\lambda_1, \lambda_2, \beta\}$ are random variables.

This formula cannot be reduced to a standard distribution. Nevertheless, by dropping the factors that depend on λ_1, λ_2 and β in turn we obtain

$$p(\lambda_1|\lambda_2, \beta, \bar{y}, \bar{t}) = \lambda_1^{(y_i-1)} e^{-(t_i+\beta)\lambda_1} \propto Gamma(y_i, t_i + \beta) \tag{17}$$

which holds also for λ_2.

Further, as or β, the full conditional is

$$p(\beta|\lambda_1, \lambda_2, \bar{y}, \bar{t}) = \beta^9 e^{-20\beta} \propto Gamma(10, 20) \tag{18}$$

Thus, the conditional distributions are much more easy to be sampled because their distribution is well known. In the case this reduction were not possible, other techniques are used (Section 6).

5.3 Sampling algorithm

In order to explain how the sampling algorithm works, we will use an example with three random variables $\{\theta_1, \theta_2, \theta_3\}$ having real observations $\bar{y}$ associated. The posterior probability distribution is $p(\theta_1, \theta_2, \theta_3|\bar{y})$ and the algorithm samples from this function based on the formulae of the full conditionals obtained at the previous step (Wilkinson, 2013).

The steps of the Gibbs sampling algorithm (Gibbs sampler), also used by JAGS, are the following

1. pick a vector of starting values for the random variables, using the prior distributions of the variables;
$$\theta^{(0)} = \{\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}\} \tag{19}$$
2. select θ_1 and draw a sample for this variable ($\theta_1^{(1)}$) from its full conditional by fixing the values of the other variables to $\theta_2^{(0)}$ and $\theta_3^{(0)}$. In other words, draw a sample from $p(\theta_1|\theta_2^{(0)}, \theta_3^{(0)}, \bar{y})$;
3. select θ_2 and draw a sample from $p(\theta_2|\theta_1^{(1)}, \theta_3^{(0)}, \bar{y})$, i.e. using the updated value of θ_1;
4. select θ_3 and draw a sample from $p(\theta_3|\theta_1^{(1)}, \theta_2^{(1)}, \bar{y})$, i.e. using both the previously updated values;
5. build the vector $\theta^{(1)} = \{\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(1)}\}$;
6. build a sequence of vectors $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(t)}$ by using the above sampling procedure;
7. stop after a number of M steps, i.e. at $t^* = M$.

After a certain number of iterations, the algorithm will have produced M samples for the variables. The algorithm above can be easily adapted to the case of more than 3 variables. In the next section, we will explain that the last produced samples are likely to be the most reliable ones, if M is large enough.

5.4 Gibbs sampling correctness

The Gibbs sampler produces a Markov chain of samples, because at each step it estimates each variable using values of the other variables either at the previous time instant or at the current time instant. Indeed, for a set of variables $\{\theta_1, \ldots, \theta_n\}$ the Gibbs sampler builds the following transition probabilities distributions

$$p_t(\theta_i^{(t)}|\theta_1^{(t)}, \ldots, \theta_{i-1}^{(t)}, \theta_{i+1}^{(t-1)}, \ldots, \theta_n^{(t-1)}, \bar{y}) \tag{20}$$

which are transition probability distributions of a Markov chain.

This chain can be demonstrated to be *invariant* (Neal, 1993), based on the fact that each transition probability distribution only depends on the other variables with fixed values and on the data. Further, full conditionals are proportional to the posterior density (Section 5.1) and the iterative samples production approximates more and more those of the full conditionals by construction. If the transition probabilities turn to be all non-zero (i.e. the chain is irreducible) then the probability of remaining in the same state is non-zero. Overall, this chain can be demonstrated to satisfy the conditions of the ergodic theorem (Neal, 1993), thus it converges ergodically to the samples of a posterior distribution.

5.5 Producing optimal estimations from the samples

The expected value of a function of a random variable ranging between two numbers a and b is

$$E[a(x)] = \int_a^b a(x)p(x)\mathrm{d}x \tag{21}$$

where $p(x)$ is the probability distribution of x.

Based on this definition, *Monte Carlo Integration* is a technique to approximate this integral by means of the samples of x (MacKay, 1998; Walsh, 2004). This technique estimates the expected value of a random variable with an average of the samples of x drawn from $p(x)$, where $p(x)$ could also be a posterior probability density.

In other words, the expected value is approximated by

$$E[a(x)] \approx \frac{1}{n}\sum_1^n a(x_i) \tag{22}$$

where $\{x_1, \ldots, x_n\}$ are n samples of x drawn from $p(x)$.

In the same way, the expected value of x based on the Gibbs sampling output can be estimated as

$$E[x] = \int_a^b xp(x)\mathrm{d}x \approx \frac{1}{n}\sum_1^n x_i = \tilde{\mu} \tag{23}$$

The variance of x is defined as $V[x] = E[(x - E[x])^2]$ and in Monte Carlo Integration, this is approximated as

$$V[x] \approx \frac{1}{n-1}\sum_1^n (x_i - \tilde{\mu})^2 \tag{24}$$

The rationale behind Monte Carlo Integration, is that the approximation becomes more accurate as soon as the number of samples increases. This is a direct consequence of the Strong Law of Large Numbers (Loeve, 1963) if the samples are independent. Indeed, Gibbs sampling generates samples that are slightly dependent, but there are practices that allow to reduce this bias. In particular, the Markov chain produced by the Gibbs sampler usually begins to converge after the generation of many samples. Thus, it is

good practice to discard the first k produced samples, where k depends on the speed of convergence of the chain. These samples are usually referred to as *burn–in* iterations. Further, in order to break the dependency between the draws in the Markov chain, one draw every d can be kept, where d is heuristically chosen (Froese et al., 2014; Lyle Gurrin and Ekstrom, 2013). This practice is named *thinning*. Finally, sensitivity to the starting point of the chain can be reduced by producing several Markov chains of samples that start from different initial values and finally merging them (*multiple chains*). Burn–in, thinning and the number of multiple chains are initialization parameters of JAGS.

Monte Carlo Integration can be applied to the Gibbs sampling output. In this case, the expected value of each variable is estimated as the average of the samples from the posterior probability. This is a valid approach if the Gibbs sampling produces many values from the ranges where most of the probability is concentrated. Instead, if the posterior distribution is strongly skewed or has a complex shape, it may be difficult that samples come from these ranges and using percentiles or taking the mode of the samples could give better estimates.

6 The JAGS approach to Gibbs sampling

JAGS implements the algorithms explained in Section 5 and applies Gibbs sampling in a transparent way to its users. Users are only asked to indicate the probability density functions for the priors, the conditionals and the likelihoods and thus define a hierarchical model. During the execution of a model, JAGS produces the formula of the posterior distribution and applies a simplification process to write the full conditionals of all the variables (Section 5.2). Gibbs sampler is then applied, which uses complex strategies (e.g. Slice sampling or Adaptive Rejection sampling) to sample the full conditionals, should these be not associable to any known statistical distribution. For too complex functions, JAGS uses the Metropolis–Hastings algorithm directly, instead of the standard Gibbs sampling, in order to directly approximate and sample the posterior probability density.

Important input parameters to JAGS are (i) the number of Markov chains to produce (in order to possibly avoid non–ergodic behaviour), (ii) the burn–in iterations (to reduce the unreliability of the first samples), (iii) the thinning parameter (to enhance samples independency), (iv) indication on variables associated to real observations (to enable likelihoods detection).

In the end, JAGS produces samples on which the user can apply Monte Carlo Integration to calculate the optimal values for the parameters. Alternatively, the user can use percentiles or other quantities to estimate these values.

```
#Activate required R packages
library(R2jags)
library(coda)

#True parameters, reported for verification and priors initialisation
a = 10
b = 20
slope = b/a

#Build the Theoretical Hockey-Stick function using the true values
numberOfRealdata = 100
xsamples<-seq(length=numberOfRealdata, from=1, to=numberOfRealdata)
theoretical_hockeyStick<-ifelse(xsamples < a, xsamples * slope, b)

#Add noise to the theoretical hockey-stick function
noisy_hockeyStickSamples<-theoretical_hockeyStick+runif(numberOfRealdata, -1, 1);

#From this point on, we 'forget' the theoretical Hockey-Stick and use the noisy samples to reconstruct it back

#Assume now that a, b and slope are just initial indicative values for the best estimates. This is a prior assumption of the model.

#Admit an error around these indicative values
SD.slope = 0.01
SD.a = 0.01
SD.b = 0.01
N = numberOfRealdata

#Initialize JAGS by stating which are our reference data, i.e. the prior values and the noisy data
jags.data <- list("N","a","b","slope","SD.a","SD.b","SD.slope","xsamples","noisy_hockeyStickSamples")

#Indicate to JAGS that we are interested in obtaining estimates for a,b and slope given the real data
jags.params <- c("random_a","random_b","random_slope")

#Build the BUGS model - the order of the variables declarations is not important
Model = "
model {
#Static definitions
random_slopetau<-pow(SD.slope,-2)
random_a_tau <- pow(SD.a, -2)
random_b_tau <- pow(SD.b, -2)

#Prior probability densities
random_slope ~ dnorm(slope, random_slopetau) #BUGS uses tau instead of the standard deviation directly
random_a ~ dnorm(a,random_a_tau)
random_b ~ dnorm(b,random_b_tau)

#Likelihoods - JAGS understands these are likelihoods nature from the fact that real data are available for the variables
#Assume that each sample comes from a normal distribution around the random prior values
for (j in 1:N){
    #definition
    y[j] <- ifelse(xsamples[j] < random_a, xsamples[j] * random_slope, random_b)
    #likelihood
    noisy_hockeyStickSamples[j] ~ dnorm (y[j], random_b_tau)
  }
}
"
# Write the BUGS model in a file
JAGSFILE="r2ssb.bug"
cat(Model, file=JAGSFILE)

#Setup the Gibbs sampling parameters
Nchains = 2 #number of Markov chains - to account for non-ergodic convergence
Nburnin = 100 #burn-in iterations - n. of initial iterations to discard
Niter = 1000 #total n. of iterations
Nthin = 10 #thinning - take every 10 samples to lower the dependency among the samples

#Run the Gibbs sampling
jagsfit <- jags(data=jags.data, working.directory=NULL, inits=NULL, jags.params,
                model.file=JAGSFILE, n.chains=Nchains, n.thin=Nthin, n.iter=Niter, n.burnin=Nburnin)

#Recover the samples
random_a_samples<-jagsfit$BUGSoutput$sims.list$random_a
random_b_samples<-jagsfit$BUGSoutput$sims.list$random_b
random_slope<-jagsfit$BUGSoutput$sims.list$random_slope

#Perform Monte Carlo Integration to get the best estimates of a and b
a_best<- mean(random_a_samples)
SD.a_best<- apply(as.matrix(a_best),2,sd)
b_best<- mean(random_b_samples)
SD.b_best<- apply(as.matrix(random_b_samples),2,sd)

cat("Best estimate for a :",a_best, "\n")
cat("Best estimate for b :",b_best, "\n")

#plot observation data
plot(xsamples,noisy_hockeyStickSamples,col="blue")
#plot the true value of a and the true hockey-stick function
abline(v=a, lty=3, col="red",lwd=1.5)
text(x=a+10, y=9,"True value of a")
#plot the re-estimated hockey-stick function
lines(x=c(0,a_best,numberOfRealdata), y=c(0,b_best,b_best),col="green")
text(x=a+40, y=17,"Estimated Hockey-Stick function")
```

Figure 3: Example of R script estimating a hockey-stick function based on a set of noisy data. The model produces an ergodic Markov chain of samples for the a and b parameters of the estimated hockey-stick function.

We report a complete example of an R program[1] that simulates a hockey-stick function in Figure 3, with in-line comments. This example was built because a hockey–stick function is the basis of models in several domains, e.g. in marine biology (Froese et al., 2014), environmental and climate monitoring (Mann, 2013; Tol and Fankhauser, 1998), finance (Safer, 2003) etc. The example also shows how conditional instructions

[1]Downloadable as a script at the following link
`http://data.d4science.org/bXRtcjFlYTdqeDVaZWYvY3g2bDZNb2RoZDdvOFpxSUpHbWJQNStISON6YzO`

are written in JAGS. The program uses JAGS with Monte Carlo Integration to get optimal estimates for the provided input parameters. For each step, Figure 3 reports how JAGS interprets the instruction, the models variables, the likelihoods etc. Further, it shows how a simple MCMC model is initialised in JAGS and how the MCMC output can be used in a Monte Carlo Integration to estimate the hockey stick parameters. In other words, this example summarises several concepts explained so far and shows how these are practically used in JAGS.

The output of the code are estimates for the a and b parameters of the hockey-stick function, along with the charts displayed in Figure 4, e.g.:

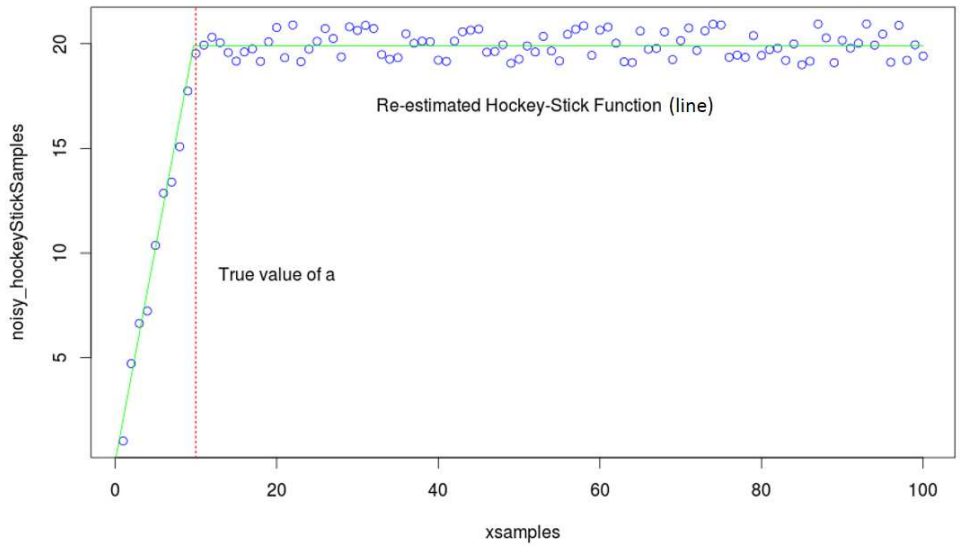

Figure 4: Output of the example R process (Section 6) that estimates a hockey-stick function using an ergodic JAGS model. The noisy observations, the estimated value of a and the reconstructed hockey-stick function are indicated.

```
>Best estimate for a : 9.74
>Best estimate for b : 19.9
```

By running the code several times the re–estimated values are always very similar, even when the uncertainty on the prior expectations is set to a high value. This demonstrates that the produced Markov chain is indeed ergodic.

7 Conclusions

In this paper we have reported the basic principles underlying Gibbs sampling and its algorithmic realization in the JAGS software. We have first described the basic principles of Bayesian Inference, Markov chains and Monte Carlo Integration to give a background that allowed to understand Gibbs sampling. The main aim of this paper is to give JAGS users a better understanding of what happens behind the scenes, because this may help developing better models.

The complete R example of Section 6, shows a practical way to implement a JAGS model and precisely indicates (i) how JAGS interprets the lines of the model, (ii) how the MCMC model is initialised and (iii) how Monte Carlo Integration can be used on the JAGS output to obtain a final estimate of the parameters of an hockey-stick function. This example merges information that is spread and segmented across several documents, manuals and theoretical books and simplifies the explanation of this powerful tool to new JAGS users.

References

Anders, R. (2013). "CCTpack: cultural consensus theory applications to data." *R package version 0.9*.

Berger, J. O. (1985). *Statistical decision theory and Bayesian analysis*. New York, USA: Springer.

Best, N., Cowles, M. K., and Vines, K. (1995). "CODA* convergence diagnosis and output analysis software for Gibbs sampling output Version 0.30." *MRC Biostatistics Unit, Cambridge*, 52.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 1. New York, USA: Springer New York.

Breiman, L. et al. (2001). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)." *Statistical Science*, 16(3): 199–231.

Casella, G. and George, E. I. (1992). "Explaining the Gibbs sampler." *The American Statistician*, 46(3): 167–174.

Chib, S. (1995). "Marginal likelihood from the Gibbs output." *Journal of the American Statistical Association*, 90(432): 1313–1321.

Chib, S. and Greenberg, E. (1995). "Understanding the metropolis-hastings algorithm." *The American Statistician*, 49(4): 327–335.

Clauset, A., Moore, C., and Newman, M. E. (2008). "Hierarchical structure and the prediction of missing links in networks." *Nature*, 453(7191): 98–101.

Coro, G. (2014). "Report of the breakout group on CMSY at the WKLIFE V workshop at IPMA in Lisbon, 5-9 October 2015." `http://www.fishbase.de/rfroese/CMSY_WKLIFE_V_ReportFinal.docx`.

Depaoli, S., Clifton, J. P., and Cobb, P. R. (2016). "Just Another Gibbs Sampler (JAGS) Flexible Software for MCMC Implementation." *Journal of Educational and Behavioral Statistics*, 1076998616664876.

Eraker, B. (2001). "MCMC analysis of diffusion models with application to finance." *Journal of Business & Economic Statistics*, 19(2): 177–191.

Espino-Hernandez, G. (2010). "WinBUGS for Beginners." `http://www.stat.ubc.ca/lib/FCKuserfiles/WinBUGSforbeginners.pdf`.

Froese, R. (2013). "FishBase goes FishBayes: R, JAGS and Bayesian Statistics." *FishBase online publication: www.fishbase.org/fishonline/english/FOL_FishBasegoesFishBayes.htm*.

Froese, R., Coro, G., Kleisner, K., and Demirel, N. (2014). "Revisiting safe biological limits in fisheries." *Fish and Fisheries*.

Froese, R., Demirel, N., Coro, G., Kleisner, K. M., and Winker, H. (2016). "Estimating fisheries reference points from catch and resilience." *Fish and Fisheries*.

Gilks, W. R., Best, N., and Tan, K. (1995). "Adaptive rejection Metropolis sampling within Gibbs sampling." *Applied Statistics*, 455–472.

Gilks, W. R., Clayton, D. G., Spiegelhalter, D. J., Best, N. G., and McNeil, A. J. (1993). "Modelling complexity: applications of Gibbs sampling in medicine." *Journal of the Royal Statistical Society. Series B (Methodological)*, 39–52.

Hojsgaard, S. (2013). "A short introduction to using JAGS and R." `http://people.math.aau.dk/\textasciitildekkb/Undervisning/Bayes13/sorenh/docs/JAGS--intro--slides.pdf`.

Huang, X., Acero, A., Hon, H.-W., et al. (2001). *Spoken language processing*, volume 15. Saddle River, New Jersey, USA: Prentice Hall PTR New Jersey.

Huber, P. (2011). "Robust Statistics." In Lovric, M. (ed.), *International Encyclopedia of Statistical Science*, 1248–1251. Berlin, Germany: Springer Berlin Heidelberg. URL `{http://dx.doi.org/10.1007/978-3-642-04898-2\_594}`

Juntunen, T., Tsikliras, A., Mantyniemi, S., and Stergiou, K. (2014). "A Bayesian population model to estimate changes in the stock size in data poor cases using Mediterranean bogue (Boops boops) and picarel (Spicara smaris) as an example." *Mediterranean Marine Science*, 15(3): 587–601.

Lam, P. (2013). "MCMC Methods: Gibbs Sampling and the Metropolis-Hastings Algorithm." `http://www.people.fas.harvard.edu/\textasciitildeplam/teaching/methods/mcmc/mcmc\_print.pdf`.

Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., Wootton, J. C., et al. (1993). "Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment." *SCIENCE-NEW YORK THEN WASHINGTON-*, 262: 208–208.

Loeve, M. (1963). "Probability theory." *Graduate Texts in Mathematics*, 45: 12.

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. (2012). *The BUGS book: A practical introduction to Bayesian analysis*. Boca Raton, Florida, USA: CRC press.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000). "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility." *Statistics and computing*, 10(4): 325–337.

Lyle Gurrin, S. H., Bendix Carstensen and Ekstrom, C. (2013). "Practical Data Analysis with JAGS using R." `http://bendixcarstensen.com/Bayes/Cph-2012/pracs.pdf`.

MacKay, D. J. (1998). "Introduction to monte carlo methods." In *Learning in graphical models*, 175–204. Berlin, Germany: Springer.

Mann, M. E. (2013). *The hockey stick and the climate wars: Dispatches from the front lines*. New York City, USA: Columbia University Press.

Neal, R. M. (1993). "Probabilistic inference using Markov chain Monte Carlo methods." `https://www.cs.princeton.edu/courses/archive/fall07/cos597C/readings/Neal1993.pdf`.

— (2003). "Slice sampling." *Annals of statistics*, 705–741.

Norris, J. R. (1998). *Markov chains*. Number 2008 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge, UK: Cambridge university press.

Ntzoufras, I. (2011). *Bayesian modeling using WinBUGS*, volume 698. New York, USA: Wiley.

Plummer, M. (2003). "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003). March*, 20–22.

— (2008). "Penalized loss functions for Bayesian model comparison." *Biostatistics*, 9(3): 523–539.

— (2011). "rjags: Bayesian graphical models using MCMC." *R package version*, 2(0).

— (2013). "rjags: Bayesian graphical models using MCMC." *R package version*, 3.

— (2015). "JAGS Version 4.0.0 user manual." `http://ftp.tw.freebsd.org/distfiles/mcmc-jags/jags_user_manual.pdf`.

Resnik, P. and Hardisty, E. (2010). "Gibbs sampling for the uninitiated." Technical report, DTIC Document.

Robert, C. P. and Casella, G. (1999). "Monte Carlo Integration." In *Monte Carlo Statistical Methods*, 71–138. New York City, USA: Springer.

Rosenberg, A. A., Fogarty, M., Cooper, A., Dickey-Collas, M., GutiÚrrez, N., Hyde, K., Kleisner, K., Kristiansen, T., Longo, C., Minte-Vera, C., et al. (2014). *Developing new approaches to global stock status assessment and fishery production potential of the seas...* Rome, Italy: FAO, Roma (Italia).

Safer, A. M. (2003). "A comparison of two data mining techniques to predict abnormal stock market returns." *Intelligent Data Analysis*, 7(1): 3–13.

Spiegelhalter, D., Thomas, A., Best, N., and Gilks, W. (1996). "BUGS 0.5: Bayesian inference using Gibbs sampling manual (version ii)." *MRC Biostatistics Unit, Institute of Public Health, Cambridge, UK*, 1–59.

Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2007). "OpenBUGS user manual, version 3.0.2." *MRC Biostatistics Unit, Cambridge*.

Stefano, B. (1998). "Using linear regression analysis and the Gibbs sampler to estimate the probability of a part being within specification." *Quality and reliability engineering international*, 14(4): 237–246.

Su, Y.-S. and Yajima, M. (2012). "R2jags: A Package for Running jags from R." *R package version 0.03-08, URL* `http://CRAN.R-project.org/package=R2jags`.

Thompson, W., Rouchka, E. C., and Lawrence, C. E. (2003). "Gibbs Recursive Sampler: finding transcription factor binding sites." *Nucleic acids research*, 31(13): 3580–3585.

Thorson, J. T., Cope, J. M., and Patrick, W. S. (2014). "Assessing the quality of life history information in publicly available databases." *Ecological Applications*, 24(1): 217–226.

Tol, R. S. and Fankhauser, S. (1998). "On the representation of impact in integrated assessment models of climate change." *Environmental Modeling & Assessment*, 3(1-2): 63–74.

Walsh, B. (2004). "Markov chain monte carlo and gibbs sampling." `http://web.mit.edu/~wingated/www/introductions/mcmc-gibbs-intro.pdf`.

Wilkinson, D. (2013). "Gibbs sampler in various languages." `http://darrenjw.wordpress.com/2011/07/16/gibbs-sampler-in-various-languages-revisited/`.