Applications of Business Intelligence

– SEMINAR WINTER SEMESTER 2014/2015 –

Gibbs Sampling

– SEMINAR PAPER –

Submitted by:
Nicole Ludwig

Advisor:
Prof. Dr. Dirk Neumann

Contents

1 Introduction

Almost immediately after the invention of computers, the simulation of random processes began. They were, at first playfully, called Monte Carlo methods, because of the most famous casinos there. Soon after, Markov chain Monte Carlo methods (MCMC) were developed in connection with the development of the atomic bomb. The physicist Metropolis used them to calculate thermal equilibria. They were soon used in especially physics and chemistry. With the generalization of the Metropolis algorithm by Hastings (1970), it was more widely used. Geman and Geman (1984) introduced simultaneously an algorithm which is a special case of the Metropolis-Hastings algorithm called Gibbs sampling. The Gibbs sampler simulates the posterior distribution and hence enables full Bayesian inference. In general, one can say that MCMC methods are a way of drawing posterior samples from a target distribution, without having to know the Bayesian normalization.
This paper first introduces Markov chains and their characteristics in Section 2, it then explains Markov chain Monte-Carlo methods in form of the Metropolis-Hastings algorithm in Section 3 and, more specifically, Gibbs sampling in Section 4. In Section 5, we implement Gibbs sampling in the statistical software R.

2 Markov Chains

The general idea of Markov chain Monte Carlo methods is that instead of unrelated points, we sample a Markov chain. To do so, we have to understand Markov chains which are thus explained in this section.

Definition 2.1 *Two variables x and y are conditionally independent given z, thus* $x \perp\!\!\!\perp y|z$*, iff the conditional joint can be written as a product of conditional marginals (Murphy 2012), i. e.,*

$$x \perp\!\!\!\perp y|z \iff p(x,y|z) = p(x|z)p(y|z)$$

.

Definition 2.2 *A sequence of random variables* $x_1, x_2, \ldots$ *in discrete time on the state space E, is a* Markov chain *for* $n = 1,2,\ldots$ *if*

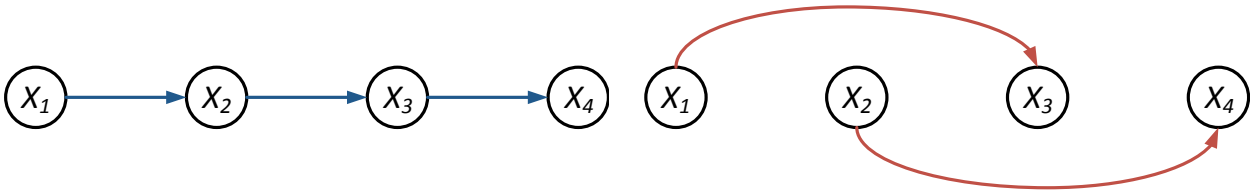
$$x_{n+1} \perp\!\!\!\perp (x_0, x_1, \ldots, x_{n-1})|x_n.$$

The above definition indicates that the future state x_{n+1} is independent of all previous states, given the present state. This is also referred to as the *fundamental Markov property* (Rønn-Nielsen and Hansen 2014).
So, Markov chains are stochastic processes that generally do not have any memory. But, if the Markov chain has a certain memory m and the future process is thus not entirely independent of the past states, we can create Markov chains of higher order. Where a Markov chain of order m represents a chain that depends on the past m states.

Definition 2.3 *A stochastic process $x = x_n$ is a Markov chain of order m $\forall i_0, i_1, \ldots, i_{t+1}, n \geq p+1$ iff*

$$p(x_{n+1} = i_{n+1} | x_n = i_n, \ldots, x_{n-m+1} = i_{n-m+1}, \ldots, x_0 = i_0) = p(x_{n+1} = i_{n+1} | x_n = i_n, \ldots, x_{n-m+1} = i_{n-m+1}).$$

In Figure 1 we can see a Markov chain of first-order, which means that the future state is only dependent on the current state x_n and we can see a second-order Markov chain, where the future state x_{n+1} is dependent on the state x_{n-1}.



(a) A first-order Markov Chain. **(b)** A second-order Markov Chain.

Figure 1: The time dependences of a first-order and a second-order Markov chain, where arrows indicate the depenedencies.

Transition Probability

The *transition probabilities* of a Markov chain, give the probability of the chain moving from one state to another. The visualization of the transition matrix $\boldsymbol{T}$ is given in Figure 2. Those probabilities are mathematically defined by

$$T_n(x_n, x_{n+1}) \equiv p(x_{n+1} | x_n).$$

If all transition probabilities $T_n = T$, the Markov chain is said to be *homogeneous* with respect to time.

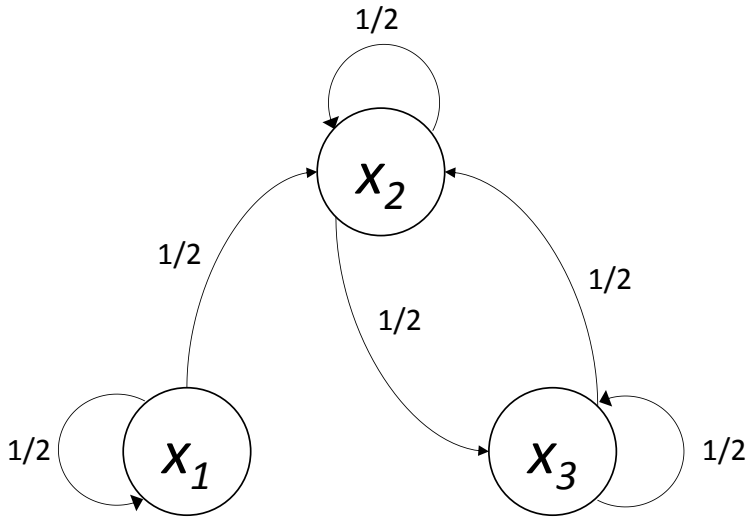


Figure 2: Visualization of the transition probabilities in a Markov chain.

In the following, we will restrict our analysis to homogeneous Markov chains. Since the transitions T_n are a probability distribution, we may write

$$\begin{aligned} T_n &\geq 0, \\ \sum_{j \in E} T_{ij} &= 1, \end{aligned}$$

where i is the current state in n and j is the state in $n+1$. The resulting *transition matrix* $\boldsymbol{T}$ is then a *stochastic matrix* (Beyn et al. 2014), where the element t_{ij} describes the probability of moving from i to j in one step. If we want to consider a transition over two steps from n to $n+2$, we use the so-called Chapman-Kolmogorov equation (Beyn et al. 2014) given by

$$p(x_{n+2}) = \sum_{x_{n+1}} p(x_{n+1}|x_n) p(x_{n+2}|x_{n+1}),$$

thus the sum over the transition probability from state n to state $n+1$ times the transition probability from state $n+1$ to state $n+2$. We can give the transition matrix from Figure 2 in matrix notation by

$$T = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}.$$

Here the chain moves with probability $1/2$ from state one to state two in one step and returns to that state in two moves with probability $1/2 \cdot 1/2 = 1/4$. The second and third state are *persistent*. A state is persistent or *recurrent* if we have a probability that we will return to this state. Whereas the first state is *transient*, which means it has a non-zero probability that we are not going to that state again.

Stationarity

Another important characterization of Markov chains is that of stationarity. A Markov chain is stationary or in an equilibrium if each step in the chain leaves the marginal probability for a particular variable unchanged, i. e.

$$p(x_{n+1}) = \sum_{x_n} p(x_{n+1}|x_n) p(x_n).$$

Therefore, we can write a stationary distribution π on the state space E as

$$\pi^t = \pi^t \boldsymbol{T}.$$

Thus, all the x_n are identically distributed, but generally not independent. But, the existence of a stationary distribution does not imply a convergence of the chain to that distribution.

Reversibility – Detailed Balance

A sufficient condition for invariance or stationarity is *detailed balance*. A Markov chain is said to have detailed balance if the probability of moving from x_1 to x_2 is the same as the probability of moving from x_2 to x_1

$$\pi(x_1)p(x_2|x_1) = \pi(x_2)p(x_1|x_2).$$

If detailed balance holds and the chain is at stationarity, then the time-reversal of the chain and the chain itself are the same. This is called *reversibility* of a chain with respect to π.

Ergodicity

If a Markov chain is *ergodic*, we can reach every state starting from any other state in a finite number of steps and, for $n \to \infty$, we will reach the stationary distribution $\pi(x)$ (Bishop 2006). Formally, it follows for any function $f : E \to \mathbb{R}$ and any initial distribution π^0 that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(x_k) = \sum_{i\in E} f(i)\pi(i)$$

Speed and Accuracy

An ongoing discussion between several researchers concerns the so-called *burn-in phase* of a Markov chain. Some researchers argue that samples we draw before the chain has reached its stationary distribution are not from the target distribution π. One should thus discard those samples as they can lead to wrong conclusions (Murphy 2012). Others, argue that you may easily discard to many samples and also make wrong conclusions. If discarding some of the samples makes sense depends on how close the starting point is to the target distribution. As we most of the time do not know this, trial and error seems inevitable.
The amount of time it takes a Markov chain to converge to the stationary distribution and forget its initial state is called *mixing time*. The mixing time τ of a chain is defined as the maximum mixing time τ_ε over all initial states x_0 which is defined as

$$\tau = \max \tau_\varepsilon(x_0) \equiv \min\{t : ||\delta_{x_0}(x)\boldsymbol{T}^t - \pi||_1 \leq \varepsilon\},$$

where $\delta_{x_0}(x)$ is the distribution with all its mass in state x_0 and $\boldsymbol{T}$ is the transition matrix. Therefore, $\delta_{x_0}(x)\boldsymbol{T}^T$ is the distribution after t steps.

3 Metropolis Hastings Algorithm

The Gibbs sampler, which we discuss in Section 4, is considered a special case of the Metropolis-Hastings algorithm. Thus, we introduce the more general algorithm in this section.
The basic idea behind the Metropolis-Hastings algorithm includes the proposal of a move from our current state x_n to the next state x_{n+1} with a probability distribution q, which we call the

proposal distribution. We can choose the proposal freely and would get an *independence sampler* if we choose the new state to be independent from the current state; thus fulfilling

$$q(x_{n+1}|x_n) = q(x_{n+1}).$$

After having proposed a move to the next state, we decide whether to accept this proposal and go to the new state x_{n+1}, or reject the proposal stay in state x_n and repeat the sample. We base this decision on a formula which helps as to ensure that we spent time in each state proportional to the target distribution $\pi(x)$. For a symmetric proposal $q(x_{n+1}|x_n) = q(x_n|x_{n+1})$, which means that our probability distribution is the same regardless of the way we take, from state n to state $n+1$ or the other way round, is given with an acceptance probability r of

$$r = \min\left(1, \frac{\pi(x_{n+1})}{\pi(x_n)}\right).$$

In the case of the proposal being asymmetric, we need a correction to take care of the proposal distribution favouring specific states. Our acceptance probability is then given by (Murphy 2012)

$$\begin{aligned} r &= \min(1, \alpha) \\ \alpha &= \frac{\pi(x_{n+1})q(x_n|x_{n+1})}{\pi(x_n)q(x_{n+1}|x_n)} = \frac{\pi(x_n)q(x_{n+1}|x_n)}{\pi(x_{n+1})q(x_n|x_{n+1})}, \end{aligned}$$

where α is called the *acceptance rate*.
We can see that one of the most important tasks here is the choice of the proposal distribution. Our proposal distribution is valid given the target distribution π, if it gives a non-zero probability of moving to the states that have non-zero probability in the target (Murphy 2012). The advantage of the next algorithm we discuss is, that we do not need to choose the proposal distribution.

4 Gibbs Sampling

As stated above, the Gibbs sampler can be seen as a special case of the Metropolis-Hastings algorithm. We wish to sample from $p(x) = p(x_1,\ldots,x_n)$ and have chosen an initial state for the Markov chain. In this case, we want to sample each variable alternately, dependent on all other variables in the distribution (Bishop 2006), thus sampling the *full conditional* for variable i

$$p(x_i|\boldsymbol{x}_{-i}),$$

where $\boldsymbol{x}_{-i}$ denotes all x without i. We could also look at it as if we drill through the full distribution sampling only along one coordinate direction in turns.
Let us take a look at a simple example for a distribution with 3 variables $p(x_1,x_2,x_3)$. At step t we have ${x_1}^t,{x_2}^t,{x_3}^t$, we now sample

$$\begin{aligned} &\text{Step 1: } {x_1}^{t+1} \sim p(x_1|{x_2}^t,{x_3}^t), \\ &\text{Step 2: } {x_2}^{t+1} \sim p(x_2|{x_1}^{t+1},{x_3}^t), \\ &\text{Step 3: } {x_3}^{t+1} \sim p(x_3|{x_1}^{t+1},{x_2}^{t+1}), \end{aligned}$$

which we could easily generalize for more variables (Murphy 2012).
We said before Gibbs sampling has the advantage that we do not need to specify a proposal distribution. This results from using the full conditional without changing $\boldsymbol{x}_{-i}$. We can show that the Metropolis-Hastings algorithm now has an acceptance rate α of 100 %. For the purpose of a clear representation, we use $x_n = x$ and $x_{n+1} = x'$ in the following

$$\alpha = \frac{p(x')q(x|x')}{p(x)q(x'|x)} = \frac{p(x'_i|\boldsymbol{x'}_{-i})p(\boldsymbol{x'}_{-i})p(x_i|\boldsymbol{x'}_{-i})}{p(x_i|\boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})p(x'_i|\boldsymbol{x}_{-i})},$$

exploiting the facts

$$\boldsymbol{x'}_{-i} = \boldsymbol{x}_{-i},$$

and

$$q(\boldsymbol{x'}|\boldsymbol{x}) = p(x'_i|\boldsymbol{x}_{-i}).$$

Then, we get

$$\alpha = \frac{p(x'_i|\boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})p(x_i|\boldsymbol{x}_{-i})}{p(x_i|\boldsymbol{x}_{-i})p(\boldsymbol{x}_{-i})p(x'_i|\boldsymbol{x}_{-i})} = 1.$$

A problem that occurs with Gibbs sampling concerns its efficiency. If the probability mass, thus the part where the density is concentrated, of the distribution is in a lower dimensional manifold than the dimensional space (Figure 3), then considering all dimensions alternately is not efficient. As going into the dimensions where no probability mass is, gives us no information. This is mostly the case if some of the variables are tightly coupled. Faster computation can thus be achieved if one finds an intrinsic dimension.

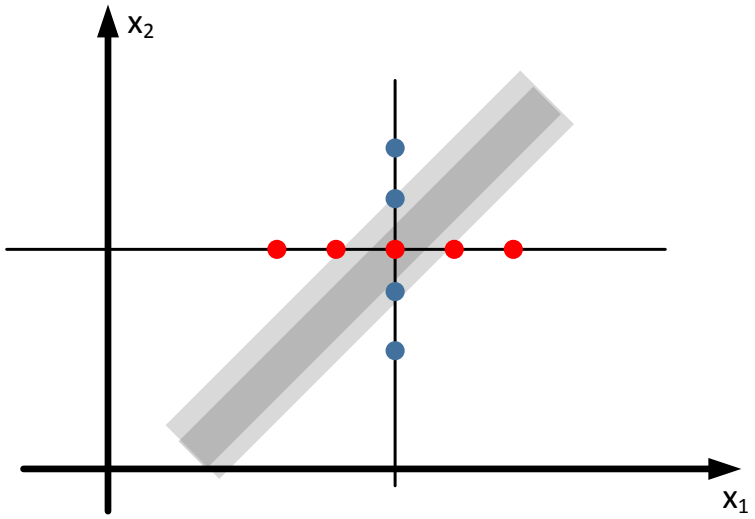


Figure 3: Visualization of the Gibbs samplers problem if the probability is defined on a lower dimensional manifold. In this example we see that the probability mass is focused on a one-dimensional line, while we have a two-dimensional state space.

Another problem is that Gibbs sampling can be slow as it only updates one parameter at a time and we may take time to move to another state if there are high correlations between the variables. An alternative that can help with those correlations is *blocked Gibbs sampling*. Instead of sampling each parameter, we group the highly correlated parameters and sample those groups block-wise. This helps us to make larger steps through the state space and reach our equilibrium.

5 Application of Gibbs Sampling in R

We now have explained the theoretical foundations of Gibbs sampling. This section gives an introduction to the implementation in the statistical software R. We use the code published by Darren Wilkinson[1] and the package `gibbs.met` by Li (2012). In our example we sample a bivariate normal distribution with $n = 1000$ and a correlation of $\rho = 0.75$.
First, we have a look at Wilkinson's code. Here, we define a function, where we input the parameters n and ρ. We create a 2×2 matrix into which we sample alternately the variables for x and y, always dependent on our previously sampled variables. Afterwards we test our function for our example.

```
gibbs <- function (n, rho) {
  mat <- matrix(ncol = 2, nrow = n)
  x <- 0
  y <- 0
  mat[1, ] <- c(x, y)
  for (i in 2:n) {
    x <- rnorm(1, rho * y, sqrt(1 - rho^2))
    y <- rnorm(1, rho * x, sqrt(1 - rho^2))
    mat[i, ] <- c(x, y)
  }
  mat
}

bvn <- gibbs(10000,0.75)
```

For the package we first define the probability distribution function.

```
log_pdf_mnormal <- function(x, mu, A)
{
  0.5 * ↩
     (-length(mu)*log(2*pi)+sum(log(svd(A)$d))-sum(t(A*(x-mu))*(x-mu)) ↩
     )
}
```

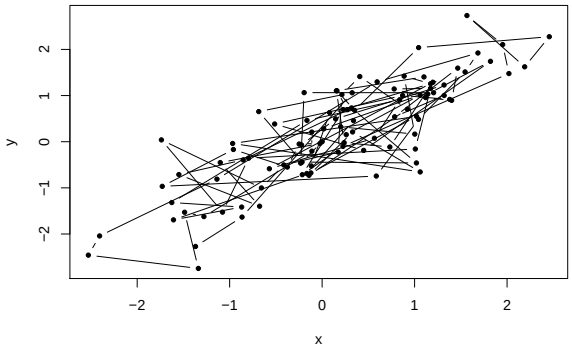
Where `mu` is the mean vector and `A` is the inverse covariance matrix, with `$\rho$` on the secondary diagonal, of the multivariate normal distribution. The main function used for the sampling is `gibbs_met`, for which we define the matrix A. The function additionally uses a metropolis acceptance step.

```
require{gibbs.met}
A <- solve(matrix(c(1,0.75,0.75,1),2,2))
mc_mvn <- gibbs_met(log_f=log_pdf_mnormal,no_var=2,
                    ini_value=c(0,0),iters=1000,iters_met=1,
                    stepsizes_met=c(0.5,0.5), mu=c(0,5), A = A)
```

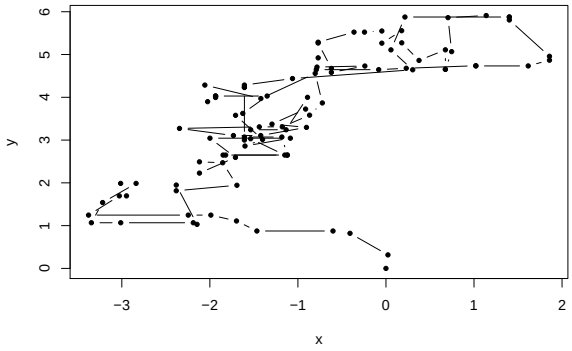
The results of both implementations are compared in Figure 4. We can see that both sample the distribution efficiently but different. The more advanced method of the `gibbs.met` package,

1 http://www.mas.ncl.ac.uk/~ndjw1/teaching/sim/gibbs/gibbs.html. Retrieved on 30.03.2015

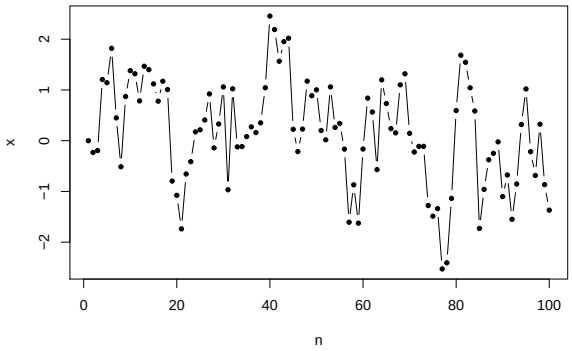
which includes a Metropolis step reaches a slightly more accurate chain after 1000 iterations. We can also observe in Figure 4c and Figure 4d that the first algorithm has a better mixing than the second algorithm. Comparing Figure 4e and Figure 4f shows that a burn-in phase might be good for the gibbs with the Metropolis step but does not seem to be necessary for the Wilkinson algorithm.



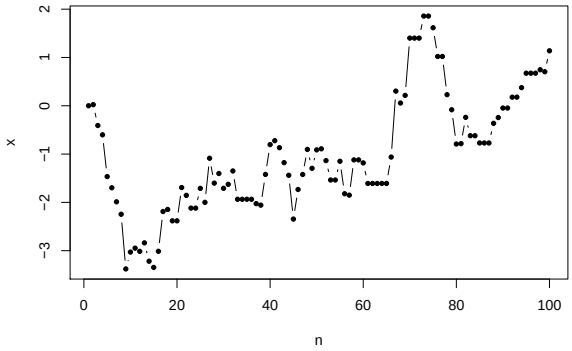
(a) The first 100 iterations for the Wilkinson sampling.



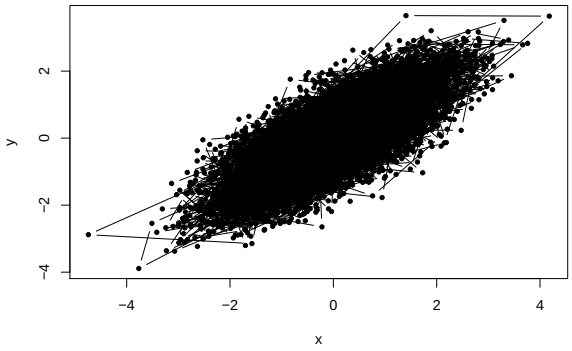
(b) The first 100 iterations for the `gibbs.met` package sampling.



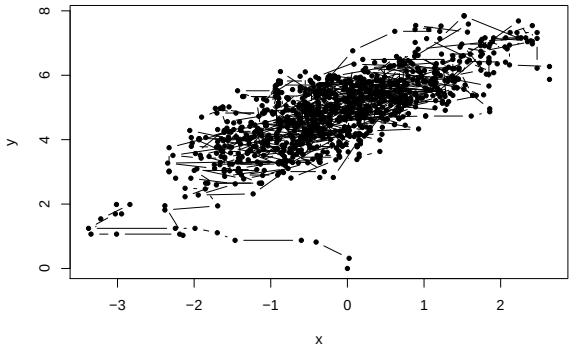
(c) Trace of the x variable for the first 100 iterations by Wilkinson.



(d) Trace of the x variable for the first 100 iterations by the `gibbs.met` package.



(e) Completed sampling by the Wilkinson algorithm with a final correlation of 0.736.



(f) Completed sampling by the `gibbs.met` package with a final correlation of 0.741

Figure 4: Comparison of the Gibbs sampling results for both implementation variants.

Gibbs sampling in R can be quite slow, but we can e. g. use C++ to fasten the sampling process. The following code is first implemented in R then incorporated into C++. The code was published by Darren Wilkinson on his research blog (https://darrenjw.wordpress.com/2011/07/31/faster-gibbs-sampling-mcmc-from-within-r/. Retrieved on 30.03.2015). It constructs a Gibbs sampler for the bivariate distribution

$$f(x,y) = kx^3 \exp\{-xy^2 - y^2 + 2y - 4x\}.$$

The pure R code takes 1088s to run[2]. Running the code in C++ in comparison takes only 34s.

```
require(Rcpp)
require(inline)

gibbscode = '
int N = as<int>(n);
int thn = as<int>(thin);
int i,j;
RNGScope scope;
NumericVector xs(N),ys(N);
double x=0;
double y=0;
for (i=0;i<N;i++) {
  for (j=0;j<thn;j++) {
    x = ::Rf_rgamma(3.0,1.0/(y*y+4));
    y= ::Rf_rnorm(1.0/(x+1),1.0/sqrt(2*x+2));
  }
  xs(i) = x;
  ys(i) = y;
}
return Rcpp::DataFrame::create( Named("x")= xs, Named("y") = ys);
'

RcppGibbsFn <- cxxfunction( signature(n="int", thin = "int"),
                            gibbscode, plugin="Rcpp")

RcppGibbs <- function(N=50000,thin=1000)
{
    RcppGibbsFn(N,thin)
}
```

2 Run on an Intel(R) Core(TM) i3 CPU @2.13GHz with 4GB RAM and Windows 7, 32-bit.

6 Conclusion

This paper introduces to Gibbs sampling by first explaining Markov chains and their characteristics and then introducing the Metropolis-Hastings algorithm. We also implemente Gibbs sampling in R and see that a connection to C++ helps us fastening the sampling process. Markov chain Monte Carlo methods and especially Gibbs sampling are powerful simulation methods, which helps us to sample from the posterior distribution of the full Bayesian.
This paper restricts itself to the analysis of Gibbs sampling in the discrete case. But one could further extent it to the analysis of the continuous case. Additionally, discussing Gibbs sampling in undirected graphical models e. g. Markov Random fields is an interesting extension.

A References

BEYN, W.-J. et al., eds. (2014). *Current Challenges in Stability Issues for Numerical Differential Equations*. Vol. 2082. Lecture Notes in Mathematics. Cham: Springer International Publishing.

BISHOP, C. M. (2006). *Pattern recognition and machine learning*. Information science and statistics. New York: Springer.

GEMAN, S. and D. GEMAN (1984). *Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 6, pp. 721–741.

HASTINGS, W. K. (1970). *Monte Carlo sampling methods using Markov chains and their applications*. In: *Biometrika*, Vol. 57, No. 1, pp. 97–109.

LI, L. (2012). *gibbs.met: Naive Gibbs Sampling with Metropolis Steps*. R package version 1.1-3. URL: http://CRAN.R-project.org/package=gibbs.met.

MURPHY, K. P. (2012). *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. Cambridge, Mass.: MIT Press.

RØNN-NIELSEN, A. and E. HANSEN (2014). *Conditioning and Markov properties*. University of Copenhagen: Department of Mathematical Sciences.

B List of Figures