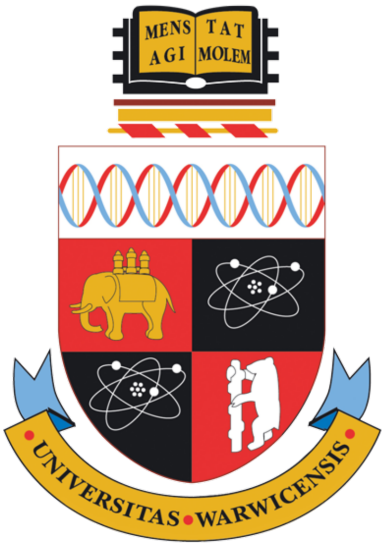
WARWICK
THE UNIVERSITY OF WARWICK



Bayesian inference under hidden Markov models

by

Gabriel Valentin Dreismann

Dissertation

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Msc. Statistics

Department of Statistics

Aug. 2019

Contents

List of Figures

Acronyms

a.s. almost surely.

HMM Hidden Markov Model.

MCMC Markov Chain Monte Carlo.

tpm transition probability matrix.

Chapter 1

Introduction

1.1 Hidden Markov Models

1.2 Motivation

Let us first motivate the model presented herein. Suppose we are interested in drawing inferences from a system which is not directly observable.
In particular, let us assume that the system comprises $m \in \mathbb{N}$ states between which it switches. The state the system is in at time $t \in \mathbb{N}$ is given by the value (in $(1, \dots, m)$) the random variable C_t assumes. Hence, the behaviour of the system through time can be described as a *process* $(C_1, C_2, \dots)$. As those states cannot be observed, they are also referred to as *hidden states*.
At each point of time, the system emits an *observation*, which can be observed. The state the system is it at any given time governs which value is emitted at that time. Hence, each state has an associated distribution, according to which the observations are distributed. Let us denote the observation at time $t \in \mathbb{N}$ by X_t. Then, analogously to the sequence of states, we have a sequence of observations $(X_1, X_2, \dots)$ where X_t is distributed according to the distribution indicated by C_t.
The aim of the algorithms developed throughout this dissertation is to draw inferences on the transitions between states, the sequence of the hidden states themselves and the parameters describing their respective distributions.

1.3 Definition

A *Hidden Markov Model* (abbrev.: HMM) - for the purposes of this dissertation - comprises the following:

- an unobservable, discrete-time, discrete-space time-homogeneous *Markov Pro-*

cess $(C_1, C_2, \dots)$ with $m \in \mathbb{N}$ distinct hidden states.
The states' realisations determine the observations' distributions.

- a transition probability matrix (abbrev.: tpm) $\Gamma \in \mathbb{R}_+^{m \times m}$ governing the state transitions of the Markov Chain. The tpm exhaustively describes $P(C_{t+1} \mid C_t)$[1].
- observable random variables $X_1, X_2, \dots$. The random variable X_t is distributed as the distribution indicated by C_t. In particular, their distribution depends *only* on an unobservable state.

Without loss of generality we shall assume that (C_t) is indexed by $\mathbb{N}$.
Note that while Γ describes the probability $P(C_{t+1} \mid C_t)$, it does not give us any information on the *initial distribution* of the chain, i.e. on C_1. Let us denote this initial distribution by $\delta \in [0,1]^m$ with $\sum \delta_i = 1$. A common choice is the so-called *stationary distribution*, which we shall explain later on.
We shall use λ to parameterise the states' distributions and use $\Theta := (\delta, \lambda, \Gamma)$ to describe all parameters of the model.
Furthermore, denote $(C_1, C_2, \dots, C_T)$ as $C^{(T)}$, $(C_1, C_2, \dots)$ as $\mathcal{C}$, $(X_1, X_2, \dots, X_T)$ as $X^{(T)}$ and $(X_1, X_2, \dots)$ as $\mathcal{X}$.
The intuitive definitions above can be described mathematically as follows:

- $P(C_{t+1} \mid \Theta, \mathcal{X}, \mathcal{C}) = P(C_{t+1} \mid C_t)$
- $P(C_{t+1} = i \mid C_t = j) = \Gamma_{i,j}$
- $P(X_t \mid \Theta, \mathcal{X}, \mathcal{C}) = P(X_t \mid C_t)$

The dependencies can easily be visualised in the following way:

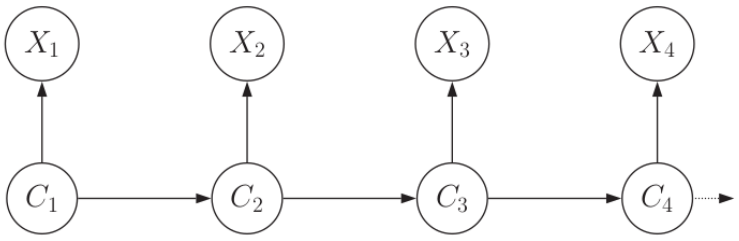


Figure 1.1: Graph of dependencies within an HMM; figure as featured in Zucchini [10]

Here, each circle represents a random variable and each arrow a conditional dependency. In particular, each variable is conditionally independent of all other variables, if it is conditioned on the variable it is connected to in this graph.

[1] The tpm is independent of t, hence *time-homogeneous*. C_t depends only on one previous time-step, which is why this process is named *Markov Process*

1.4 Aim

The aim of this dissertation is to compare two MCMC algorithms used to estimate the parameters of an HMM; namely Metropolis Hastings and Gibb's Sampler. We seek to practically compare both algorithms rather than employ theory to argue about their respective advantages. In particular, we are interested in the dependency of the requried number of iterations and running time on the complexity of the model as well as the sample size.
Although there is an abundance of research on both Metropolis Hastings and Gibb's sampler, there is a lack of direct comparisons of both methods. However, related comparisons have been drawn; for instance Ryden[8] compared the EM algorithm with MCMC methods.
For instance, we would like to empirically answer the following questions:

- How many samples does one need to reliably estimate a model of a given complexity?
- Which algorithm converges faster towards the true model?
- How does runtime depend on the model complexity?
- Do both algorithms produce chains of comparable quality?

Note that due to time constraints it is beyond the scope of this thesis to compare the state-of-the-art for both algorithms - implementing the state-of-the-art itself would be an undertaking worth of a dissertation. Rather, this dissertation compares both algorithms on a more foundational basis, which is unlikely to fundamentally change when algorithms are tweaked to achieve state-of-the-art performance.

1.5 Stationarity

This section explains an effect of Markov Chains known as *stationarity*. In particular, under a set of rather weak constraints[6], there exists a unique, discrete probability distribution $\pi \in \mathbb{R}_+^m$ named the *stationary distribution*, with the following properties:

$$\pi\Gamma = \pi$$
$$P(C_t = i) \overset{t\to\infty}{\longrightarrow} \pi_i \quad \forall 1 \leq i \leq m$$

The former condition states that when $P(C_t) \sim \pi \implies P(C_{t+1}) \sim \pi$ with $P\left(C_{t+1=j} \,|\, C_t = i\right) = \Gamma_{i,j}$ as usual. Hence, π is *invariant* under Γ - when the

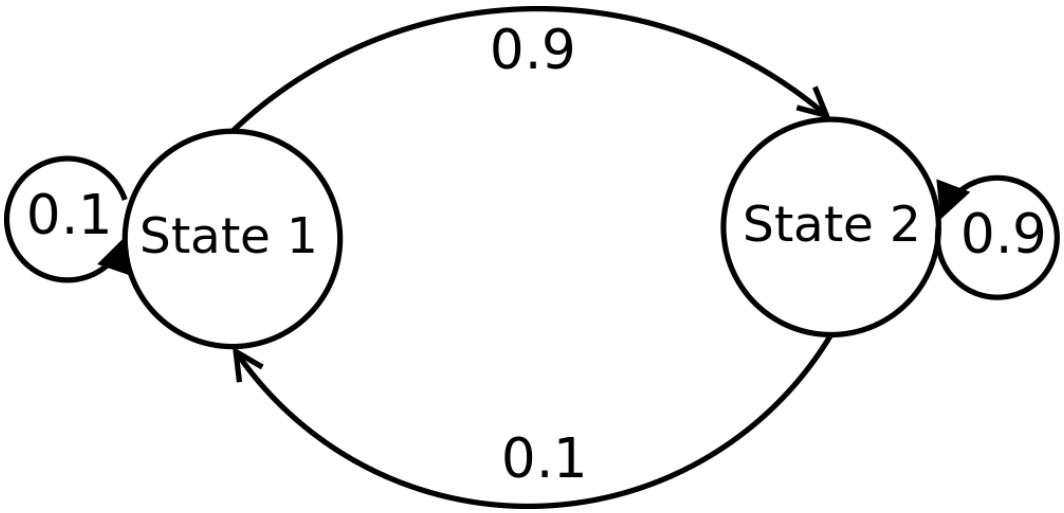


Figure 1.2: The probabilities are crafted s.t. the system is very likely to rest permanently in state 2 and return there quickly whenever it transitions into state 1

chain is at a point where the states are assumed with a probability given by the stationary distribution, the states' probabilities do not change with a step in time. The latter property states that the chain converges towards its stationary distribution as $t \to \infty$.
For details on the conditions and a proof, please refer to Breuer[2].

1.5.1 Models

In this section, we define models to which we will be referring throughout the dissertation. 5Those models are either meant to generate synthetic data for simulations or are crafted to expose a specific aspect of an algorithm.

Simple Switching Model

The *Simple Switching Model* consists of two states whose Bernoulli distributions are defined by

$$P(X_t = 0 \,|\, C_t = 1) = 1 \quad \text{i.e.} \quad X_t \,|\, C_t = 1 \sim \text{Ber}(0)$$
$$P(X_t = 1 \,|\, C_t = 2) = 1 \quad \text{i.e.} \quad X_t \,|\, C_t = 2 \sim \text{Ber}(1)$$

Figure 1.2 shows illustrates the transition matrix Γ; the initial distribution is defined to be $\delta = (0.1, 0.9)$

1.6 Approaches

There are several approaches to solve for Θ or $\mathcal{C}$, the most common being Maximum Likelihood Estimation (abbrev. MLE), Expectation-Maximization (abbrev. EM) and Markov Chain Monte Carlo Methods (abbrev. MCMC). This thesis shall explore the latter in depth. Nonetheless, we provide a brief account of the other techniques as well.
In order to outline the core ideas behind each approach, a few preliminaries are needed. These are shortly introduced. The presentation herein follows the notation in Zucchini[10].

1.7 Forward Probabilities

For the algorithms to follow, we need a means to compute the probability of being in a certain state at a certain time, as well as the probability of a string of observations occuring, given Θ. The forward probabilities provide such a means.
To be precise, forward probabilities provide a *dynamic programming* approach to this problem; they allow us to express the solution to a bigger problem as a set of sub-problems.

1.7.1 Mathematical approach

Suppose that we are able to compute the quantity

$$\alpha_t(j) = P\left(X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t, C_t = j\right)$$

For $t = 0$ this is obvious, as we have $\alpha_0 = \delta$[2].

[2](there are no observations available)

Then we able to compute $\alpha_{t+1}(i)$ as follows:

$$\begin{aligned}
\alpha_{t+1}(i) &= P\left(X^{(t+1)} = x^{(t+1)}, C_{t+1} = i\right) \\
&= \sum_{k=1}^{m} P\left(X^{(t+1)} = x^{(t+1)}, C_{t+1} = i, C_t = k\right) \\
&= \sum_{k=1}^{m} P\left(X^{(t+1)} = x^{(t+1)}, C_t = k \,|\, C_{t+1} = i\right) P\left(C_{t+1} = i\right) \\
&= \sum_{k=1}^{m} P\left(X^{(t)} = x^{(t)}, C_t = k \,|\, C_{t+1} = i\right) P\left(X_{t+1} = x_{t+1} \,|\, C_{t+1} = i\right) P\left(C_{t+1} = i\right) \\
&= \sum_{k=1}^{m} P\left(X^{(t)} = x^{(t)}, C_t = k, C_{t+1} = i\right) p_i(x_{t+1}) \\
&= \sum_{k=1}^{m} P\left(X^{(t)} = x^{(t)}, C_{t+1} = i \,|\, C_t = k\right) P\left(C_t = k\right) p_i(x_{t+1}) \\
&= \sum_{k=1}^{m} P\left(X^{(t)} = x^{(t)} \,|\, C_t = k\right) P\left(C_{t+1} = i \,|\, C_t = k\right) P\left(C_t = k\right) p_i(x_{t+1}) \\
&= \sum_{k=1}^{m} \alpha_t(k)\, \Gamma_{k,i}\, p_i(x_{t+1}) \qquad (1.1)
\end{aligned}$$

Note that there are $\mathcal{O}(m)$ calculations for each time step and entry of α_t, hence $\mathcal{O}\left(m^2\right)$ calculations for each α_t and $\mathcal{O}\left(Tm^2\right)$ calculations for processing of the entire chain. Hence, this approach is also suitable for very long chains.
Also by the law of total probability, we have

$$\sum_j \alpha_T(j) = P\left(X_1 = x_1, X_2 = x_2, \ldots, X_T = x_t\right)$$

i.e. the probability (or density) of this string of observations occuring.

1.7.2 Computational Approach

The computations given below solely express the approach described above in a convenient matrix notation. Let

$$P(x_i) := \begin{pmatrix} p_1(x_i) & \ldots & \ldots & 0 \\ 0 & p_2(x_i) & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & \ldots & p_m(x_i) \end{pmatrix}$$

where x_i is a realisation and $p_1, \ldots, p_m$ are the states' probability functions in case the distributions are discrete or probability density functions otherwise, i.e.

$$p_i(x_t) := P\left(X_t = x_t \,|\, C_t = i\right)$$

Then we have:

$$\alpha_t = \delta\, P(x_1)\, \Gamma\, P(x_2)\, \ldots \Gamma P(x_t)$$

Note that if the observations are not consecutive, the powers of Γ need to be adjusted accordingly:

$$P(X_1 = x_1, X_3 = x_3, X_T = x_t) = \delta\, P(x_1)\, \Gamma^{(3-1)}\, P(x_3)\, \Gamma^{(T-3)}\, P(x_t)\, 1'$$

where $1'$ is a vertical vector with components all 1.

1.8 Maximum Likelihood Estimation

As we have developed a way to compute $P\left(X^T = x^t \,|\, \Theta\right)$, we can use an appropriate method to maximise this expression. In fact, if we wanted to estimate Θ, we could leverage Bayes' law as follows:

$$\underbrace{P\left(\Theta \,|\, X^T = x^t\right)}_{\text{posterior distribution}} = \frac{P\left(X^T = x^t \,|\, \Theta\right) P\left(\Theta\right)}{P\left(X^T = x^t\right)} \tag{1.2}$$

$$\propto P\left(X^T = x^t \,|\, \Theta\right) P\left(\Theta\right) \quad \} \text{Bayesian estimation} \tag{1.3}$$

$P\left(\Theta\right)$ is also referred to as the *prior distribution*.
Gradient-based methods are commonly used to (locally) estimate Θ by choosing it as to maximise the probability of the data.

1.9 Expectation Maximisation

The basic idea of EM (Expectation Maximisation) is to treat the hidden states as missing data and estimate them based on an estimate of Θ. Then, having *complete data* at its disposal, it in turn estimates Θ.
The algorithm thus alternates between two steps:

1. **Expectation:**
   Complete the data (i.e. hidden states) by using their *expected values* given Θ

2. **Maximisation:**
   Given complete data (i.e. also the estimated states), choose Θ as to maximise

the likelihood

Note that the second step is often easy to compute, as *complete data* is available. The algorithm differs significantly from MCMC methods in that it

- makes point estimates of both hidden states and parameters
- uses the expectation for this estimate

As opposed to this, MCMC in the Bayesian context operate on *distributions* of parameters. The target values in EM are hence distributions in MCMC and the expectation in EM is replaced by sampling from a posterior distribution in MCMC.

1.10 Markov Chain Monte Carlo Methods

In a Bayesian understanding, model parameters are not just values, but are thought of as distributions themselves.
Unfortunately, the complexity of the problems at hand does often not allow for an analytical solution to the parameters' posterior distribution. Still, we would like to gain an understanding of the distributions' shape. Aside from the general shape of the posterior distributions, point estimates like mean, variance, Kurtosis etc. are also of interest.
A canonical solution to this problem is to sample $\Theta_1, \Theta_2, \dots$ independently from the posterior distribution. By the central limit theorem (CLT) and the delta method, we can then approximate all quantities of interest to arbitrary accuracy as the length of the sample approaches infinity.
In reality, the algorithms we use for sampling (except for Rejection-Sampling) generally do not produce mutually independent samples; hence $(\Theta_1, \Theta_2, \dots)$ is also referred to as a *chain*. Among others, the level of (in)dependence of the samples is a quality measure of the chain itself. For details on how we ascertain the sampler's quality, please refer to the respective chapter of measures.
As aforementioned, the EM algorithm operates on maxima and hence can be thought of a specific case of the Bayesian interpretation - namely, operating only on the point estimates instead of the distributions themselves. These point estimates are often computed by analytically maximising likelihood; hence no sampling is required. On the other hand, as point estimates, they provide less information than the posterior distributions one obtains in a Bayesian framework.

1.10.1 Metropolis-Hastings Sampler

The Metropolis-Hastings Sampler (abbrev. MH) can be thought of as a *stochastic gradient descent*.

Starting from a given initial value Θ_1, MH sampler produces a Markov Chain of $(\Theta_1, \Theta_2, \dots)$ by repeatedly choosing a new theta in the direct vicinity of the previous one.
A new estimate of Θ is obtained by modifying the current sample slightly (usually with a step sampled from a normal distribution), estimating the new probability using 1.3 and accepting the sample based on this new probability.
Hence, a Metropolis Hastings Sampler can be defined by providing

- $\tilde{P}(\Theta) := P\left(\Theta \mid X^T = x^T\right)$
  The posterior probability of the new sample
- $Q(\Theta, \Theta')$
  The probability of considering Θ' as the new candidate when Θ is the current candidate.
  That is, the new candidate is sampled from $Q(\Theta, \cdot)$.

After drawing a new sample Θ' starting from Θ, Θ' is accepted depending on the following:

1. Let
$$\alpha := min\left\{1, \frac{\tilde{P}(\Theta')}{\tilde{P}(\Theta)} \frac{Q(\Theta', \Theta)}{Q(\Theta, \Theta')}\right\} \tag{1.4}$$
2. Draw $u \sim U[0, 1]$
3. If $u \leq \alpha$, accept Θ' (i.e. set $\Theta_{t+1} = \Theta'$), otherwise reject it (i.e. $\Theta_{t+1} = \Theta_t$).

If the probability of acceptance is low, then the chain often rests at the same position and hence exhibits a high autocorrelation - the chain is then also called "sticky". This is obviously detrimental for statistical independence and hence reduces the information inherent to the generated chain. The measure of *effective sample size* tries to capture the loss of information due to this effect and shall be elaborated upon later on.
We will be using a symmetric Q for this thesis; note that in this case, α simplifies to

$$\alpha := min\left\{1, \frac{\tilde{P}(\Theta')}{\tilde{P}(\Theta)}\right\}$$

Note that convergence of the chain produced by this sampler is by no means obvious; [6] may be consulted for details.

1.10.2 Gibb's Sampler

The Gibb's sampler samples by repeatedly sampling one component while fixing all other components. When fixing all other components, a the *joint distribution* becomes a *marginal distribution*, which is often much easier to sample from. In other words:
To illustrate the Gibb's sampler, let Y be an m-dimensional vector parameterising a given model completely. Let Y_{-i} denote all the parameters *except for* the ith parameter. We again consider a chain $Y^{(1)}, Y^{(2)}, \ldots$ of parameters; $Y_i^{(t)}$ then denotes the ith component of the parameterisation at time step t.
The Gibb's sampler works as follows in each step:

- Draw index $i \in \{1, \ldots, m\}$ uniformy
- Draw $\tilde{Y}_i^{(t+1)} \sim P\left(Y_i^{(t)} \,|\, Y_{-i}^{(t)}\right)$
- let $Y^{(t+1)} := \left(Y_1^{(t)}, Y_2^{(t)}, \ldots, Y_{i-1}^{(t)}, \tilde{Y}_i^{(t+1)}, Y_{i+1}^{(t)}, \ldots, Y_m^{(t)}\right)$

The variant presented herein is called the *Random Scan Gibb's Sampling* because of the uniform choice of i. When i is chosen successively (for instance to ensure that all parameters are regularly updated), this is referred to as the *Systematic Scan Gibb's Sampler*.
The Gibb's sampler is in fact a special case of the Metropolis-Hastings sampler. Note that

$$Q(Y^{(t)}, Y^{(t+1)}) = \frac{1}{m} P\left(Y_i^{(t+1)} \,|\, Y_{-i}^{(t)}\right)$$

where i is the altered component. Then, in equation 2.4, we have:

$$\begin{aligned}
\frac{P(Y_i^{(t+1)})}{P(Y_i^{(t)})} \frac{Q(Y^{(t+1)}, Y^{(t)})}{Q(Y^{(t)}, Y^{(t+1)})} &= \frac{P\left(Y^{(t+1)}\right)}{P\left(Y^{(t)}\right)} \frac{P\left(Y_i^{(t)} \,|\, Y_{-i}^{(t+1)}\right)}{P\left(Y_i^{(t+1)} \,|\, Y_{-i}^{(t)}\right)} \\
&= \frac{P\left(Y_i^{(t+1)} \,|\, Y_{-i}^{(t)}\right)}{P\left(Y_i^{(t)} \,|\, Y_{-i}^{(t)}\right)} \frac{P\left(Y_{-i}^{(t)}\right)}{P\left(Y_{-i}^{(t)}\right)} \frac{P\left(Y_i^{(t)} \,|\, Y_{-i}^{(t+1)}\right)}{P\left(Y_i^{(t+1)} \,|\, Y_{-i}^{(t)}\right)} \\
&= 1
\end{aligned}$$

Hence we have $\alpha = min\{1, 1\} = 1$ and all samples drawn are accepted.

Chapter 2

Implementation

2.1 Working in Log Space

A commonly encountered problem in numerical computation is underflow and overflow. Computers allocate a limited amount of memory to store numbers; hence, numbers must neither grow to large or too small, nor can all numbers be represented to an arbitrary accuracy. To which accuracy numbers are stored depends on a number of factors, most notably the programming language and system architecture. For most every-day applications however, those restrictions are negligible. For our purposes, only underflow will be relevant.
In the following, we will

- show that the computations our algorithms conduct are indeed affected by numerical underflow
- present a mathematical approach to address this issue
- discuss the limitations of this approach

Numerical Underflow

Numerical underflow describes the phenomenon that a number becomes so small that a program is not able to store it any more. The following pseudocode illustrates a program that outputs the first power p of 10 s.t. $\left(\frac{1}{10}\right)^p$ cannot be represented any more:

Triggering Numerical Underflow

$c \leftarrow 1$
$p \leftarrow 1$
while not(c == 0.0) **do**
    $c \leftarrow \frac{c}{10.0}$
    $p \leftarrow p + 1$
end while
print(p)

When running this algorihm on the machine used for this dissertation, the algorithm finishes at $p = 325$. Note that this is already considerably higher than the precision provided by the underlying operating system. For instance, having 64 bits of precision, we expect the lower bound for representable numbers to be roughly at

$$-\log_{10}\left(\left(\frac{1}{2}\right)^{64}\right) = \sum_{i=1}^{64} \log_{10}(2) < 64 \times 0.4 = 25.6$$

Practical Example

Having triggered numerical underflow itself, we move on to show how numerical underflow will occur naturally in the context of HMMs.
In particular, recall that

$$\alpha_t(j) = P(X_1 = x_1, \ldots, X_t = x_t, C_t = j)$$

This quantity must be evaluated by several algorithms presented later; it is used to sample hidden states from a marginal distribution.
Unfortunately, $\alpha_t(j)$ is prone to numerical underflow. Consider the Simple Switcher Model as introduced in the Models section. In particular, consider the string of observations

$$(X_1 = 1, X_2 = 1, \ldots, X_T = 1)$$

By construction, it follows that $C_1 = 2, C_2 = 2, \ldots, C_T = 2$. As $P(X_t = 1 \,|\, C_t = 2) = 1$, we have

$$P(X_1 = 1, X_2 = 1, \ldots, X_T = 1) = 0.9^T$$

Just as in the example above, this probability tends to zero exponentially and will trigger numerical underflow. As such, we will use it to confirm that the circumvention developed below indeed outperforms a naiive implementation.
Note that while this example is especially crafted to trigger numerical underflow,

even in real-world examples, $\alpha_t(j)$ will be the culprit. In general, the probability of specifically observing *any* given series of observations will tend towards zero, as it is a product of conditional probabilities. In particular, all indices of $\alpha_t(j)$ will normally tend towards zero as $t \to \infty$.
A common approach to circumvent this problem is to work in *log space* instead of the regular probability space. That is to say, if we work with probabilities only in terms of their logarithm, numerical underflow will not occur[1].

Mathematical Approach

In the following, we describe how the recursion in equation 1.1 can be adapted to work with probabilities in log space.
Recall that the forward algorithm is defined as follows:

$$\begin{aligned} \alpha_0 &:= \delta \\ \alpha_i &:= \alpha_{i-1}\, \Gamma^t\, P(x) \quad \text{for } i \geq 1 \end{aligned}$$

The latter can be expressed as follows:

$$\alpha_t(k) = \sum_{j=0}^{m} \alpha_{t-1}(j)\, \Gamma^{d_t}(j,k)\, P_{(k,k)}(x_t)$$

The first step of the induction is straightforward, as we can simply define

$$\tilde{\alpha}_0(k) := log\,(\sigma_k)$$

For the recursion, we leverage the following identity[2]:

$$log_b\left(\sum_{i=0}^{m} a_i\right) = log_b(a_0) + log_b\left(1 + \sum_{i=1}^{m} b^{log_b(a_i) - log_b(a_0)}\right) \tag{2.1}$$

for $a_0 > 0, a_i \geq 0$ and $a_0 \geq a_i \,\forall\, 1 \leq i \leq m$ and any $b > 0$.
In particular, we note that the expression on the right-hand side can be evaluated

[1] However, overflow could occur. Note that as $t \longrightarrow 0$, $log(t) \longrightarrow -\infty$. However, as we have seen in the example above, the log probability is proportional to the length of the chain itself. Hence, only if the length of the chain exceeds the magnitude our environment can store can overflow occur. However, this means that on a 64 bit computer, a chain of length $\propto 2^{64}$ is necessary. Generally, those chains then approach a length at which the overall runtime of the algorithm is of greater concern than numerical overflow.

In short, overflow in log space will only occur at magnitudes where the algorithm cannot be applied due to other reasons in the first place.

[2] `https://en.wikipedia.org/wiki/List_of_logarithmic_identities`

by *exclusively* relying on the logarithm of a_i.
Let $\beta_{j,k,i} := \Gamma^{d_i}(j,k)P_{(k,k)}(x_i)$. Note that neither $\Gamma^{d_i}(j,k)$ nor $P_{(k,k)}(x_i)$ depend on the position within the Markov Chain; $\alpha_{i-1}(j)$ does, however.
By using the newly defined entities, we rewrite and have:

$$\alpha_i(k) = \sum_{j=0}^{m} \underbrace{\alpha_{i-1}(j)\,\beta_{j,k,i}}_{a_{j,k}} \tag{2.2}$$

By combining equations (2.2) and (2.1) we obtain

$$\begin{aligned} log_b\left(\alpha_i(k)\right) &= log_b\left(\sum_{j=0}^{m} a_{j,k}\right) \\ &= log_b\left(\alpha_{i-1}(0)\,\beta_{0,k,i}\right) + log_b\left(1 + \sum_{i=1}^{m} b^{log_b(\alpha_{i-1}(j)\,\beta_{j,k,i}) - log_b(\alpha_{i-1}(0)\,\beta_{0,k,i})}\right) \\ &= log_b(\alpha_{i-1}(0)) + log_b(\beta_{0,k,i}) \\ &\quad + log_b\left(1 + \sum_{i=1}^{m} b^{log_b(\alpha_{i-1}(j)) + log_b(\beta_{j,k,i}) - log_b(\alpha_{i-1}(0)) - log_b(\beta_{0,k,i})}\right) \end{aligned}$$

Further setting

$$\tilde{\alpha_i}(k) := log\left(\alpha_i(k)\right) = log_b\left(\sum_{j=0}^{m} a_{j,k}\right)$$

we obtain

$$\tilde{\alpha}_i(k) = \tilde{\alpha}_{i-1}(0) + log\left(\beta_{0,k,i}\right) + log\left(1 + \sum_{j=1}^{m} b^{\tilde{\alpha}_{i-1}(j) + log_b(\beta_{j,k,i}) - \tilde{\alpha}_{i-1}(0) - log_b(\beta_{0,k,i})}\right) \tag{2.3}$$

Hence, we have expressed the log probability at a given time t in terms of the log probabilities at $t-1$ and thus have defined a recursion for the forward algorithm in log space.
Note that we have chosen a_0 to be the maximum of all a_i. In practice, a_i can be reordered such this condition is fulfilled.

Why this approach is numerically stable

Let us shortly review why this approach does indeed not suffer from numerical underflow. As pointed out above, neither $\Gamma^{d_i}(j,k)$ nor $P_{(k,k)}(x_i)$ depend on the

position within the Markov Chain but $\alpha_{i-1}(j)$ does. Hence, only the latter could suffer underflow depending on and due to the length of the chain.
Hence, in recursion 2.3 we only need to consider $\tilde{\alpha}_{i-1}(0)$ and the sum. The former does not suffer numerical underflow by induction. In the latter, $log_b\left(\beta_{j,k,i}\right), log_b\left(\beta_{0,k,i}\right)$ can be discarded and by

$$\begin{aligned} b^{\tilde{\alpha}_{i-1}(j)+log_b(\beta_{j,k,i})-\tilde{\alpha}_{i-1}(0)-log_b(\beta_{0,k,i})} &= b^{\tilde{\alpha}_{i-1}(j)-\tilde{\alpha}_{i-1}(0)}\, b^{log_b\left(\beta_{j,k,i}\right)-log_b\left(\beta_{0,k,i}\right)} \\ &\propto \underbrace{b^{\tilde{\alpha}_{i-1}(j)-\tilde{\alpha}_{i-1}(0)}}_{A:=} \end{aligned}$$

we only need to consider whether A could underflow. This could happen iff the power of b becomes very small, which implies that $\tilde{\alpha}_{i-1}(j)$ and $\tilde{\alpha}_{i-1}(0)$ differ by magnitudes. By the experiment conducted above, we know that R can handle differences in magnitude up to a power of 325 (for base 10). If an underflow were to occur in this setting, we know that while one state is *very likely*, the other state would be *so* unlikely that the possibility of it being assumed is basically negligible.
In particular, if the probabilities decrease at about the same rate[3], then no underflow can occur. This is what we expect to happen in practice.
However, if the probabilities differ by such a magnitude that one state is *very* probably and another one *very* unprobable, underflowing probabilities can be ignored by an algorithm which is adapted accordingly; this algorithm will then only consider states which belong to the class of somewhat probable states. All other states would not impact the result in any numerically significant way.

Ancillary Result

Last but not least, we are interested in the overall probability

$$P\left(X_1 = x_1, \ldots, X_t = x_t\right)$$

As defined above, this quantity can be computed by utilising

$$P\left(X_1 = x_1, \ldots, X_t = x_t\right) = \sum_{k=0}^{m} \alpha_T(k)$$

However, in the log-implementation, we are given $\tilde{\alpha}_T(k) := log\left(\alpha_T(k)\right)$ and we are interested in $log\left(\sum_{k=0}^{T} \alpha_T(k)\right)$. How to express the sought term in terms of the log

[3]rate specifically in the sense that their decreases all belong to the same complexity class as in the Landau Θ notation

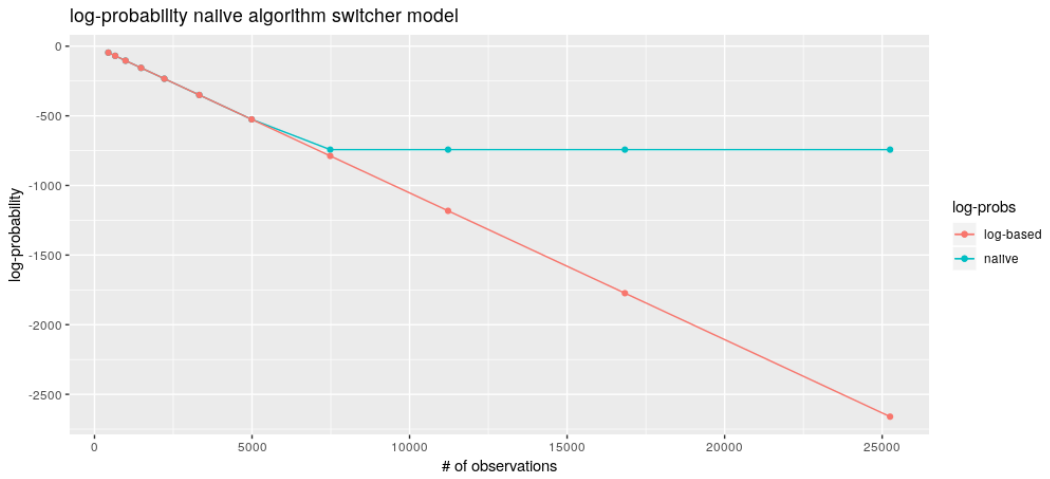


Figure 2.1: The log probability of the naiive implementation deviates from the result we would expect mathematically (straight line with constant negatie slope), but conforms to our prediction with assumed underflow. As expected, log implementation does not exhibit any deviation.

probabilities becomes obvious when reading 2.1 from the right hand side to the left hand side, e.g. we can solve the problem by applying the trick backwards.

Results

Let us shortly prove with a practical example that the approach described above solves the underflow problem. As derived above, for the Simple Switcher Model, we expect a total probability of 0.9^T, which translates into $log\left(0.9^T\right) \propto -T$, i.e. a straight line with negative slope. However, we expect the log-probability of the naiive implementation to snap and become a horizontal line as soon as numerical underflow occurs. At the same time, the log-probability of the log-space implementation should be unchanged.
Figure 2.1 clearly shows that the experimental results are consistent with our predictions and that the log implementation appears to work as intended.

2.2 Measures of Convergence

The approaches presented herein are inspired by Johansen[6].
Let (X_t) be the chain under consideration, $X_i \sim \mathcal{D}$ with $\mathcal{D}$ some distribution and $\phi : \mathbb{R} \to \mathbb{R}$ be a function.
ϕ is used to estimate key quantities describing $\mathcal{D}$ by evaluating $\int \phi(x)\, dP(x)$[4]; setting $\phi = id$ yields the mean and $\phi(x) := x^2$ can then be used to estimate the variance.

[4]whenever this term exists and is well-defined

With this setup, there are several aspects of convergence to consider:

- whether the chain has reached a stationary regime
- whether the chain explores the full support of the target distribution
- the independence of the chain's elements

In the following, we will not address the second aspect. This is because the distribution we are sampling from is a posterior distribution, whose closed form is unavailable to us. Hence, there is no straightforward way to define the support we expect the distribution to have.
Often, convergence of the chain is confirmed by visual inspection. However, we are interested in developing measures which can be used automatically (i.e. without human intervention) to ascertain convergence. Those methods will come in handy when evaluating how the algorithms' performance depends on the dimension of the problem.

2.2.1 Stationarity

Let

$$\hat{F}_T(x) := \frac{\sum_{i=1}^{T} 1_{X_i \leq x}}{T}$$

This is called the *empirical distribution function*, computed based on the first T elements of the chain. The indicator functions $1_{X_i \leq x}$ each assume 1 if $X_i \leq x$ and 0 otherwise [5].
By the *Glivenko-Cantelli theorem*, we have

$$\left\| \hat{F}_n - F \right\|_\infty \xrightarrow{a.s.} 0$$

Once the chain has reached its stationary regime, we expect the empirical distribution function to have converged as well. Hence, we can use differences between empirical distribution functions of the chain at different points in time to measure convergence to stationarity. We adopt an idea originally presented by Dr. Pieralberto Guarniero in a practical seminar in the context of Riemann sums: The chain of length T is split into three components of equal size $s := \lfloor \frac{T}{3} \rfloor$:

$$S_1 := (X_1, \ldots, X_s), S_2 := (X_s, \ldots, X_{2s}), S_3 := (X_{2s}, \ldots, X_T)$$

[5] Even though it is referred to as a *function*, it depends, in fact, on the underlying $\omega \in \Omega$ of the probability space and is hence actually a so-called *sample function*.

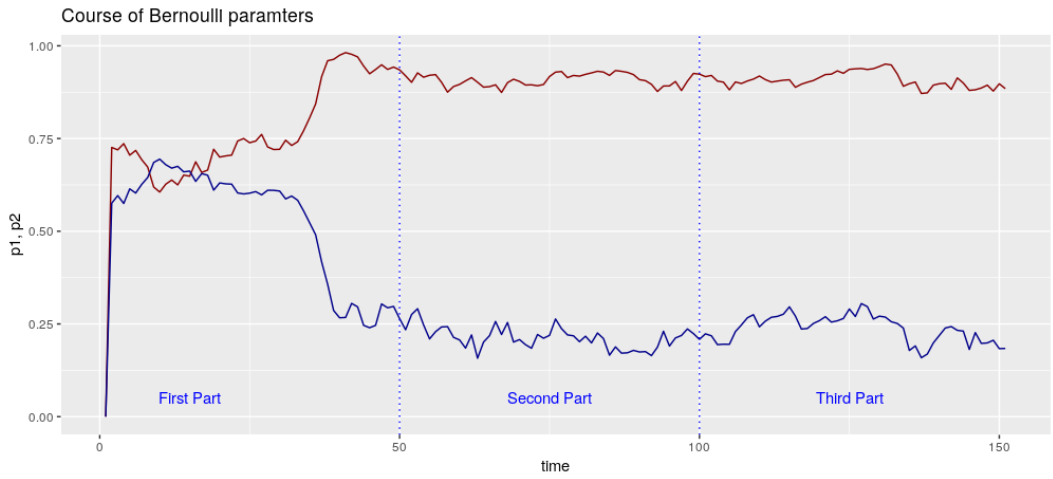


Figure 2.2: Course of Bernoulli parameters for Gibb's sampler on simple rain model

Figure 2.2 illustrates how the chain is split. Note that in the first part, the parameters are still diverging from their initial values; this part is referred to as "burn-in period". Note that because the way from the initial estimates is included, this part's empirical distribution function is usually far from the desired distribution. Therefore, the very first part is usually ignored in this analysis.
Upon a brief, graphical inspection, one might conclude that the second and third part's distribution seems to be equal and in particular, that the distribution reached stationarity after about 50 samples.
Figure 2.3 shows the estimated densities[6] for all three parts of the chain; one can clearly see that the second and third chains' modes are already "zeroing in" on the original probability (which is 0.9), while the first density is still highly skewed towards the initial choice.
Note how the second chain's density still exhibits multiple modes, whereas the third chain only has one main mode left (which is incidentally closer to the original value). Visual inspection of the chain itself fails to reveal this, making it a prime example of how an quantitative analysis can outdo mere visual inspection of the chain. However, the median (indicated by a green line) is the same for the second and third part.
Figure 2.4 allows us to quantitatively confirm the results from the visual inspection. We observe:

- the third part reaches its stationary regime earlier than the second part
- the second part's quantiles converge toward the third part's quantiles

[6] as they are graphically easier to understand than the estimated empirical distribution function itself; the densities are estimated using "geom_density" from ggplot2 with Gaussian smoothing (n=128, adjust = 0.25)

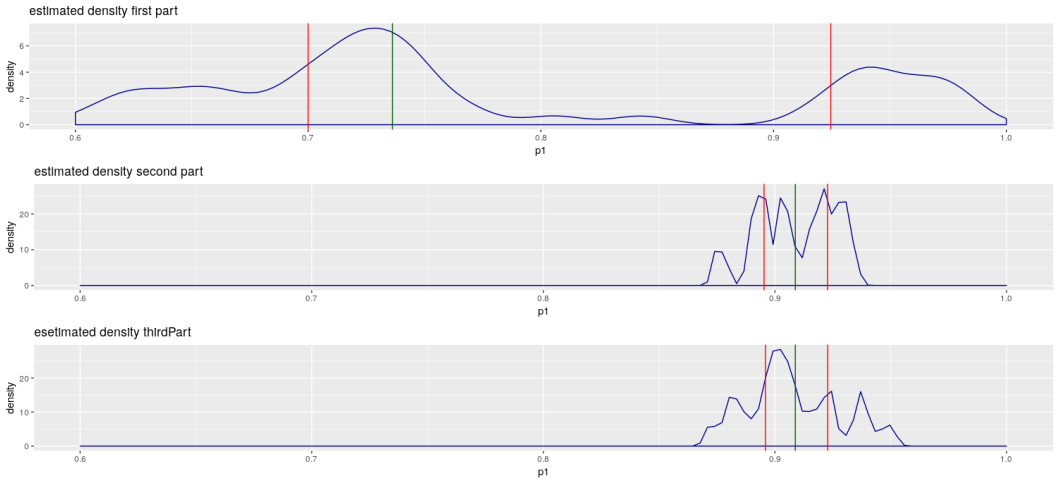


Figure 2.3: Density estimates for S_1, S_2 and S_3 with $0.25, 0.5$ and 0.75 quantiles (red, green, red) indicated by lines

Note that the second and third parts' quantiles in figure 2.4 become visually indistinguishable after a sample size of about 120.
At this point, the maximum difference has fallen below $0.01(1\%)$. Also, the second part comprises elements from about $X_{40}, \ldots, X_{80}$ whereas the third part comprises $X_{80}, \ldots, X_{120}$. This is consistent with our intuitive conclusion from figure 2.2 that the chain reaches its stationary regime after about 50 samples.

Insignificance of δ

Also note that for large sample sizes, the choice of δ usually becomes irrelevant. This is because compared to the overall length of the chain, the contribution of δ tends towards insignificance. For this reason, δ is hardly estimable, i.e. we expect great fluctuation in its quantiles. Hence, for convergence analysis, δ is discarded.

In conclusion, we have developed a quantitative measure for stationarity of the chain's distribution. We shall adopt the criterion developed herein with a cut-off criterion of 0.01. In our experiments, a convergence up to said threshold usually implied a good conversion of the overall sampling distribution. That is to say:

We assume that the chain has reached convergence once the maximum difference between the $0.25, 0.5$ and 0.75 quantiles of the second and third part are smaller than 0.01.

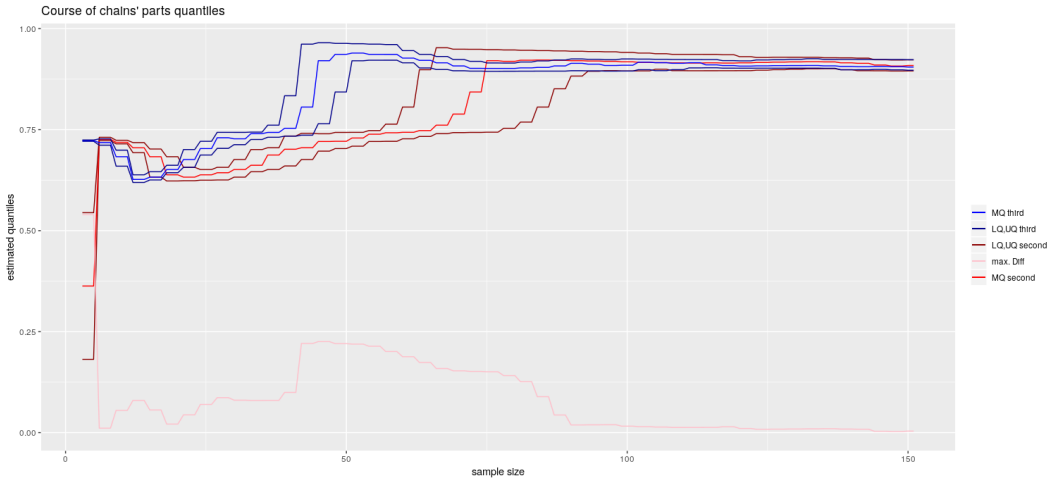


Figure 2.4: Course of quantiles for the second and third part of the chain. "LQ, MQ and UP" are shortcuts for "lower", "middle" and "upper' quantile and refer to the $0.25, 0.5$ and 0.75 quantiles, respectively. "second" and "third' refer to which part of the chain is considered for computing the quantiles. "Max. Diff" (pink) denotes the maximum difference between respective quantiles at each point in time.

2.2.2 Independence of Elements

Independence of the elements is also crucial to the quality of the chain's approximation. The more consecutive samples depend on each other, the lower the amount of information for inferential purposes is.

For instance, when estimating the variance, if the samples (X_t) are *not independent*, then the estimate of the variance of $\phi(X_t)$ is

$$Var\left(\frac{1}{T}\sum_{t=1}^{T}\phi(X_t)\right)$$

This will typically be greater than the variance of i.i.d. samples, which is $\frac{Var(\phi(X_t))}{T}$. The effective sample size (ESS) quantifies statistical dependence by computing the number of samples (X_t) which would yield the same variance as if i.i.d. samples were used.

For illustrative purposes, we follow the methodology presented in Johansen [6]; in practice, more sophisticated solutions would be in order.

Assume that (X_t) is a second-order stationary time series, that is $Var\left(\phi(X_t)\right) = \sigma^2$ and $p(X_t, X_{t+\tau}) = p(\tau)$ for $\forall t, \tau \in \mathbb{N}$ and $p(\tau)$ is the *correlation* at lag τ.

Then, Johansen [6] shows that:

$$T\,Var\left(\frac{1}{T}\sum_{i=1}^{T}\phi(X_t)\right) \overset{T\to\infty}{\longrightarrow} \sigma^2\left(1+2\sum_{i=1}^{\infty}p(\tau)\right)$$

For i.i.d. samples we have

$$Var\left(\frac{1}{T_{\text{ess}}}\sum_{i=1}^{T}\phi(X_t)\right) = \frac{\sigma^2}{T_{\text{ess}}}$$

Equating both evaluations yields

$$T_{\text{ess}} = \frac{1}{1+2\sum\limits_{i=1}^{\infty}p(\tau)}\,T$$

As the correlations can be computed empirically, this gives us an estimate of the ESS.

2.2.3 Hidden States

One of the algorithms under scrutiny, namely the Gibb's sampler, estimates the hidden states explicitly. Hence, this estimate can be used to ascertain whether the chain has converged towards its stationary distribution as well.
There are two things to note here:

- this comparison will not be meaningful depending on the system in question. If the states do not differ significantly, they are hardly estimable by only considering the observations.
- when applying the algorithms to real data, the hidden states are usually not available. Hence, this measure is only used to confirm the viability of solutions found for synthetic data. In this respect, it backs up the measures defined earlier.

Note that the algorithm's hidden states samples generally will not and should not equal the states used when originally sampling the observations. This is for the simple reason that the observations are not drawn according to the maximum likelihood. For each state, we draw from the state's distribution; thus we will not just draw the value which is most likely to be drawn. This way, the overall likelihood is also less than the maximum likelihood. The hidden states' estimate will have a lower likelihood than the MLE solution.

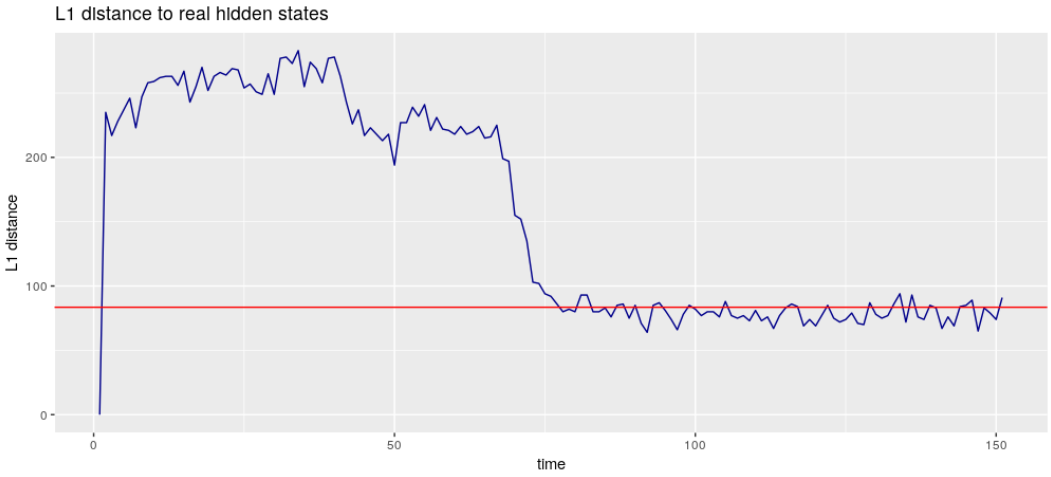


Figure 2.5: The red line indicates the expected number of wrong hidden states, the blue number the number of wrong hidden states at points t of the sampled chain. We clearly see that after about 70 iterations, the algorithm has reached the expected value.

However, there is an (analytical) way to compute the *expected deviation* from the original chain of hidden states. To this end, let us consider the $L_1(\mathbb{R}^T)$[7] distance between the original chain $(C_1, \ldots, C_T)$ and the sampled chain $\left(\tilde{C}_1, \ldots \tilde{C}_T\right)$.
We expect $P(C_t = i)$ to be high (i.e. i likely to be sampled as the hidden state at time t) iff $P\left(X_t = x_t \,|\, C_t = i\right)$ is high. Let $j := \text{maxarg}P\left(X_t = x_t \,|\, C_t = j\right)$. Then we expect to misestimate the state with a probability of $\omega := \sum_{i=1, i\neq j}^{T} P\left(X_t = x_t \,|\, C_t = i\right)$.
Furthermore, once the chain as reached stationarity, this allows us to compute the expected number of deviated states: it is $T * \omega$.

2.2.4 Gradual increase of model complexity

Let us shortly consider the increase of model complexity as a function of m, the number of states. There are $m(m-1)$ parameters to be estimated for Γ, and m parameters, one for each state's distribution. Let us neglect δ in this argument, as it quickly becomes insignificant as the sample size increases.
This means that for $m = 2$, we have 6 parameters (4 parameters to estimate) and for $m = 3$, this number rises already to 12 (8). We observe that for $m = 2$, convergence is reached quite fast with our vanilla implementation. However, for $m = 3$ the picture is markedly different: the number of iterations needed for convergence becomes large. We suspect that complexity rises exponentially in the number of parameters to estimate. If this is indeed the case, $m = 2$ and $m = 3$, or 6 and 12 parameters

[7] $\|x\| := \sum |x|$

respectively, are indeed extremely different in terms of computational complexity. Hence, we need to find a means to analyse complexity in a more fine-grained manner. An obvious approach to do so is to fix a number of parameters and only estimate the remaining parameters. In the following, we suggest a simple scheme.

Fixing Parameters

To simplify the modifications to our model, we restrict ourselves to fix only the most important parameters. Those are, by construction, the diagonal of Γ as well as the states' parameters λ.
When fixing parameters, we first fix elements on the diagonal of Γ; after having fixed m parameters, we continue to fix parameters in λ in increasing order. Hence, if 1 parameter is fixed, only $\Gamma_{1,1}$ is fixed. If we fix $k = 5$ parameters in a $m = 3$ model, we fix $\Gamma_{i,i}$ for $1 \leq i \leq 3$ as well as λ_1, λ_2.
When talking about a model with a complexity of l parameters, we choose the model of lowest complexity which requires at least l parameters to be estimated and fix all the remaining parameters. For instance, if we would like a model of 4 parameters, we will choose $m = 2$; this model has 6 parameters, of which we will fix 2.
Our hope is that by introducing this scheme, we are able to analyse the increase of complexity in a more fine-grained way.

2.3 Metropolis-Hastings Sampler

As outlined in section 1.10.1, the Metropolis Hastings Algorithm can be defined by its posterior probability and $Q(\Theta, \tilde{\Theta})$. The former can be computed recursively as shown in section 1.7.1; so Q remains to be defined.
Symmetric Proposals, i.e. proposals where $Q(\Theta, \tilde{\Theta}) = Q(\tilde{\Theta}, \Theta)$, are commonly employed. In absence of prior information indicating otherwise, this choice is natural as it does not arbitrarily favour some parameters over others. A common choice for a symmetric proposal is the normal distribution $\mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$.

Autocorrelation

The choice of σ directly influences the auto correlation the chain's samples will exhibit and hence the effective sample size. In particular, small values of σ lead to proposals which are always very close to the current sample, incurring a high auto correlation. On the other hand, if σ is too great, then the newly proposed value is likely to be extreme and hence unlikely. This makes the new sample very likely to be rejected. Rejection leads to strings of identical values, which are of course perfectly auto correlated.

Figures 2.6a and 2.6b show two chains sampled with different choices of σ for the same data and proposal distribution. They clearly illustrate how a σ chosen too big can lead to poor behaviour of the resulting chain.

Exploration of Search Space

Small σ lead to local suggestions; hence, multi-modal distributions are unlikely to be explored as the chain gets "stuck" in local maxima. A commonly used technique known as "Simulated Annealing" circumvents this phenomenon by decreasing σ in the course of time. In the beginning, chains are then able to explore the search space, whereas later on with a small σ, they are also likely to zero in on the sought value. Please note that in Bayesian Inference, we are interesting in sampling a *distribution* instead of a *point estimate*. Hence, σ cannot not be chosen arbitrarily small.
For these reasons, it is important to choose σ wisely. Indeed, Johansen [6] suggests to tweak σ such as to obtain an acceptance rate of 0.44 in the 1-dimensional case and 0.28 in the multi-dimensional case. Often, those optimisations are performed manually by trial & error; however, [5] suggests an algorithm adapting σ automatically which appears to work well for many practical examples.

Modification of acceptance criterion

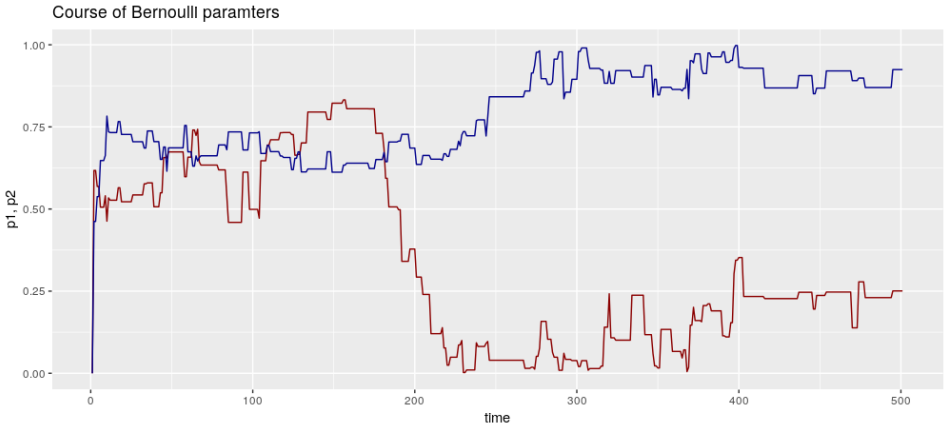
As our algorithms work in log space, we modify the acceptance criterion as follows:

Metropolis-Hastings Acceptance Criterion in Log Space

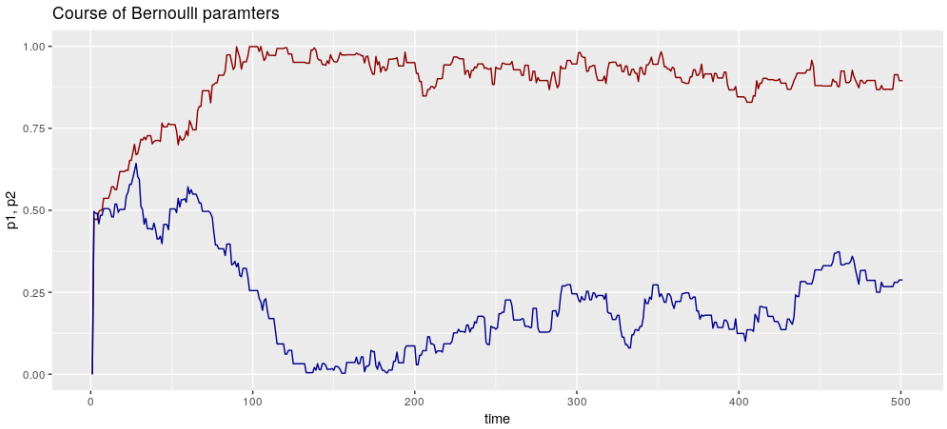
1. Let
$$\tilde{\alpha} := min\left\{0, log\left(\tilde{P}\left(\Theta'\right) Q(\Theta', \Theta)\right) - log\left(\tilde{P}\left(\Theta\right) Q(\Theta, \Theta')\right)\right\} \quad (2.4)$$
2. Draw $u \sim U[0, 1]$
3. If $log(u) \leq \tilde{\alpha}$, accept Θ' (i.e. set $\Theta_{t+1} = \Theta'$), otherwise reject it (i.e. $\Theta_{t+1} = \Theta_t$).

2.3.1 Constrained Parameters

For our purposes, Θ contains Γ, δ and a parameterisation of the states' distributions. Those are typically Bernoulli or Poisson distributions.
All of those parameters are constrained:



(a)



(b)

Figure 2.6: In figure 2.6a, the path of the Bernoulli paramters of a Metropolis-Hastings chain with proposals $\mathcal{N}(0, 0.08^2)$ is shown; 2.6b uses proposals from $\mathcal{N}(0, 0.03^2)$. We can clearly see that the chain in 2.6a is very "stick", i.e. often the chain rests with the current sample. On the other hand, the steps its Metropolis-Hastings sampler takes are usually greater than those of the not so sticky chain 2.6b.

- $0 \leq \delta_i \leq 1$ with $\sum \delta_i = 1$

- $0 \leq \Gamma_{j,i} \leq 1$ with $\sum_{i=1}^{m} \Gamma_{j,i} = 1$
  for all $1 \leq j \leq m$

- $0 \leq p \leq 1$ for Bernoulli distributions or $\lambda \geq 0$ for Poisson distributions

When choosing a new proposal $\tilde{\Theta}$ from the current sample Θ, with a step sampled from a normal distribution, it is not immediately clear how to ensure that the new proposal satisfies the constraints.
The easiest option is to simply reject all samples which are not valid. While incurring a high rejection rate[8], this methodology becomes unfeasible if we consider constraints as in (1) above.

Natural and Working Parameters

A common approach to deal with this is to apply transformations such as the *log*-transformation. For instance, if a parameter is constrained to $[0, 1]$, we could work in a transformed space instead. For instance, if our algorithm was to suggest values in $(-\infty, \infty)$ (so-called *working parameters*), we could apply the log-transformation to map those values into $[0, 1]$ (so-called *natural parameters*); this would ensure that all new proposals are valid from a model point of view.
However, the steps are usually normally distributed in the space of the natural parameters and it is unclear what the distribution of a random variable X would be s.t. $exp(X) \sim \mathcal{N}(0, \sigma)$.

"Bumping" of Parameters

When naiively using a symmetric distribution to obtain a new proposal for a parameter from the current one, we may come up with an invalid proposal. For instance, if the paramter in question is constrained to $[0, 1]$, choosing a $\mathcal{N}(0, \sigma)$ step to alter the current sample may yield a proposal outside of these bounds.
There are two approaches to this problem: Either, one alters the new proposal such that it corresponds to the constraints, or one uses a sampling method respecting the constraints in a natural way. "Bumping" is a simple methodology for the former.
Let $p \in [0, 1]$ be a parameter and $s : [0, 1] \to \mathbb{R}$ be a function mapping p to a new proposal. Then we draw new samples as follows:

[8] Which can be prohibitively high for high-dimensional problems

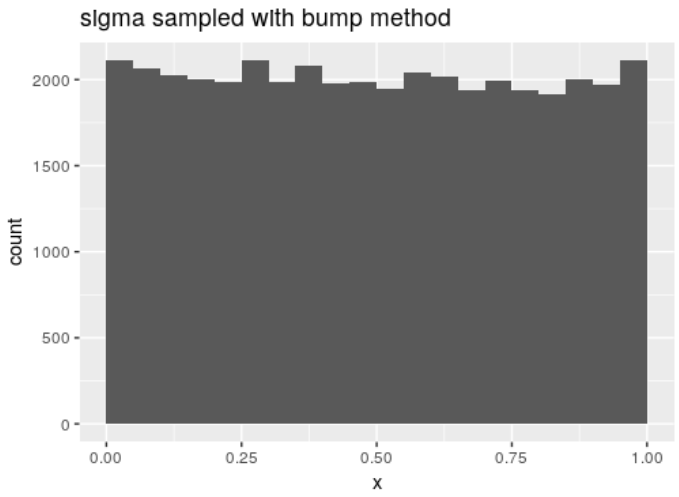


Figure 2.7: Sampling of $p \in [0, 1]$ with bumping method and $\sigma = 0.1$ for 200 chains with each 200 samples.

Bumping of parameters

$\tilde{p} \leftarrow s(p)$
if $\tilde{p} \in [0, 1]$ **then**
    $p \leftarrow \tilde{p}$
else if $min\{abs(p - 0), abs(1 - p)\} < 1$ **then**
    if $\tilde{p} < 0$ **then**
        $p \leftarrow -\tilde{p}$
    else
        $p \leftarrow 1 - (\tilde{p} - 1)$
    end if
else
    reject new proposal
end if

Figure 2.7 shows that sampling with the bumping method does indeed uniformly cover the interval $[0, 1]$[9] As we shall see, this is not true for many other methods. We note the following:

1. the interval can easily be generalised

2. the proposal is symmetric

[9]Note that as the sample size increases, the distribution continues to converge towards the uniform distribution. This has been confirmed generating several histograms and verifying that the overall shape attained is consistently almost constant.

3. if the proposed new sample is farther than 1 away from both 0 and 1, the new sample is rejected. The variance of the proposal distribution should be chosen small s.t. this case becomes unlikely.

Drawing from a constrained distribution

We would prefer to directly draw a parameter from a distribution which respects the constraints the parameter is subject to over rejecting or bumping a parameter after it has been drawn.
One way to do so is to use a *truncated normal distribution.* Let us denote the normal distribution, truncated to the interval $[a, b]$ with $X \sim \mathcal{N}_a^b(\mu, \sigma^2)$.

Lemma 1 *The cdf of X is*

$$F(x) = \begin{cases} 0 & x \leq a \\ \frac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} & otherwise \end{cases}$$

where Φ is the cdf of a standard normal random variable.

Proof 1 *Let $\tilde{X} \sim \mathcal{N}(\mu, \sigma^2)$.*
Then we have

$$\begin{aligned} F_{\tilde{X} \mid \tilde{X} \in [a,b]}(x) &= \frac{P\left(\tilde{X} < x \cap \tilde{X} \in [a, b]\right)}{P\left(\tilde{X} \in [a, b]\right)} \\ &= \frac{P\left(\frac{\tilde{X}-\mu}{\sigma} < \frac{x-\mu}{\sigma} \cap \frac{\tilde{X}-\mu}{\sigma} \in \left[\frac{a-\mu}{\sigma}, \frac{b-\mu}{\sigma}\right]\right)}{P\left(\frac{\tilde{X}-\mu}{\sigma} \in [\frac{a-\mu}{\sigma}, \frac{b-\mu}{\sigma}]\right)} \\ &\overset{x \geq a}{=} \frac{\Phi\left(\frac{x-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \end{aligned}$$

Theorem 2.3.1 (The Inversion Method) *Let $X \sim F$ where F is some cdf and let F^{-1} be the inverse[10] of F. Then*

$$F_X^{-1}(U[0, 1]) \sim X$$

Lemma 1 provides us with the cdf of a truncated normal distribution; using the inversion method, we can thus sample from a truncated normal distribution. Please

[10] or if the inverse does not exist, the pseudo-inverse

note that neither the cdf nor the inverse are available in analytical forms; still, numerical approximations are readily available in R.
Specifically, we have

$$U[0,1] \sim F_{\tilde{X} \mid \tilde{X} \in [a,b]}(\tilde{X} \mid \tilde{X} \in [a,b])$$

$$U[0,1] \sim \frac{\Phi\left(\frac{\tilde{X}-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}$$

$$U[0,1]\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right) \sim \Phi\left(\frac{\tilde{X}-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)$$

$$\Phi\left(\frac{a-\mu}{\sigma}\right) + U[0,1]\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right) \sim \Phi\left(\frac{\tilde{X}-\mu}{\sigma}\right)$$

$$\Phi^{-1}\left(\Phi\left(\frac{a-\mu}{\sigma}\right) + U[0,1]\left(\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right)\right)\sigma + \mu \sim \tilde{X}$$

where $\tilde{X} = \tilde{X} \mid \tilde{X} \in [a,b]$ for the last few lines for notational convenience.
Note that this method of sampling can be expressed in R very concisely as

```
x = qnorm(  pnorm(a,mu,sigma) + runif(1)*(pnorm(b,mu,sigma) -
 pnorm(a,mu,sigma))  * sigma + mu
```

Note that sampling methods which do not involve Φ or Φ^{-1} are also available: notably, Robert[7] proposes a method based on Rejection-Sampling.

Robert's Drawing from Truncated Normal Distribution

(Considering only the case that $0 \in [a,b]$)

$z \leftarrow U[a,b]$
$\delta \leftarrow exp\left(-\frac{z^2}{2}\right)$
$u \leftarrow U[0,1]$
if $u \leq \delta$ **then**
    accept sample
else
    restart at step 1
end if

However, numerical experiments I have conducted with R have shown that sampling using the approximative method presented is faster than the Rejection Sampling proposed by Roberts.
Unfortunately, as figure 2.8 shows, sampling with the truncated normal distribution method also introduces bias as the samples are not uniformly distributed.

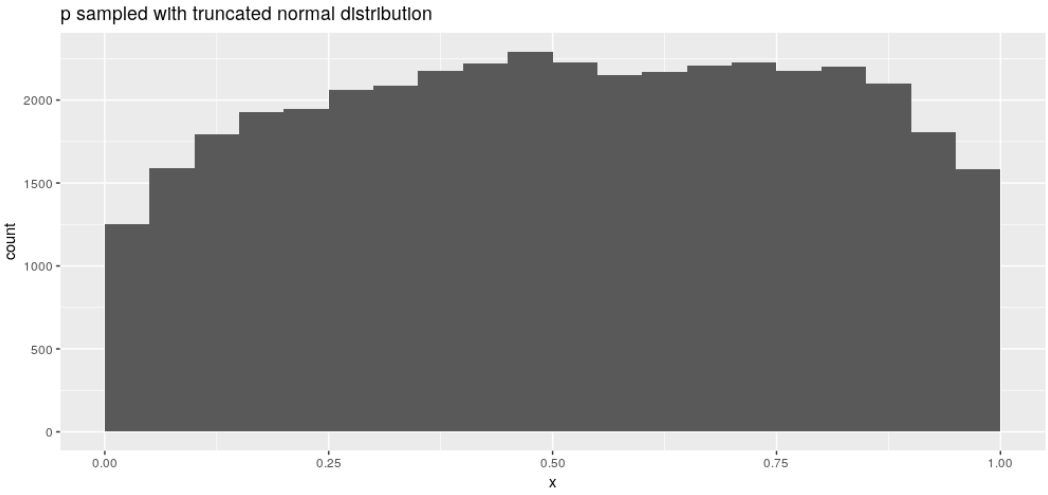


Figure 2.8: Samples drawn using truncated normal distribution as proposal; 200 chains with 200 samples each, $\sigma = 0.1$

Multiplicative Random Walk

Cappé [3] proposes an interesting class of Metropolis-Hastings-Steps which are all based on a *multiplicative random walk*. Let us first consider a parameter p with constraint $p \in \mathbb{R}_{\geq 0}$. Then let the new proposal $\tilde{p}$ be defined as

$$log\left(\tilde{p}\right) := log(p) + X$$

where $X \sim \mathcal{N}(0, \sigma)$.

Lemma 2 *When defined as above, we have* $Q\left(log(\tilde{p}), log(p)\right) = Q\left(log(p), log(\tilde{p})\right)$. *Note, that we do not have*

$$Q\left(\tilde{p}, p\right) = Q\left(p, \tilde{p}\right)$$

Please note that even though Q is symmetric, we need to invoke the substitution method to obtain the correct acceptance probability. In particular, note that

$$\begin{aligned}\int f(x)dx = \int_{log(exp(y))} f(x)dx &= \int_{log(y)} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(exp(x)-0)^2}{2\sigma^2}} \left(exp(x)\right)' \, dx \\ &= \int_{log(y)} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(exp(x)-0)^2}{2\sigma^2}} exp(x) \, dx\end{aligned}$$

This means that when manipulating p in log-space, the acceptance rate needs to be

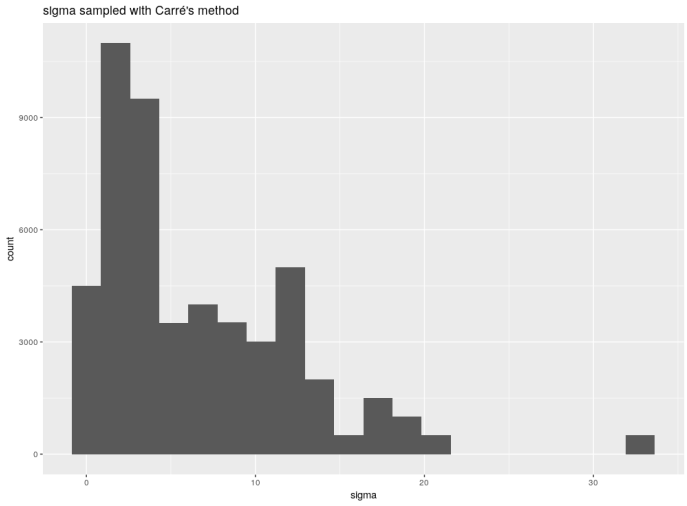


Figure 2.9: Sampling σ by applying Metropolis-Hastings with the multiplicative random walk, 200 chains with each 200 steps and the initial sample drawn from $U[0, 10]$

altered as follows[11]:

$$\alpha := min\left\{1, \frac{\tilde{P}\left(\tilde{\Theta}\right)}{\tilde{P}\left(\Theta\right)} \frac{\tilde{p}}{p}\right\}$$

Note that while the sampling is uniform *on the log space*, it is not uniform in normal space. Figure 2.9 shows a histogram of σ sampled by the aforementioned method[12]. As σ is unbounded in this case, it is indeed not possible to uniformly sample σ[13]. However, this methodology (needlessly) introduces a prior which is *inherent* to the sampling method itself and will affect any intentional prior which might be defined in addition.

Cappé [4] further suggests to extend this methodology to accommodate parameters $p_1, \ldots, p_n \in [0, 1]$ with additional constraint $\sum p_i = 1$. This is achieved by obtaining $\tilde{p_i}$ as defined above and then setting

$$p_i' := \frac{\tilde{p_i}}{\sum_{j=1} \tilde{p_j}}$$

as the new proposed value. Cappé suggests to further use $Exp(1)$-priors on p_i'.

Note that - just as noted above, the resulting distributions are heavily *biased*. Fig 2.11 and 2.10 show estimated densities and histograms of $p1, p2 \in [0, 1]$ with $p1 + p2 = 1$ sampled with 200 chains each having 200 steps and $\sigma = 0.1$, the initial values being uniformly sampled from $[0, 1]$. Please note that the sample is biased regardless of

[11] Note that this also implies that 0 must not be part of the domain of p.

[12] the initial values are distributed as in $U[0, 10]$

[13] There exists no unbounded, uniform distribution

whether the proposed priors are used or not. Hence, this method of drawing numbers has an *inherent prior* which will distort any extra, intended prior that could be defined in addition.

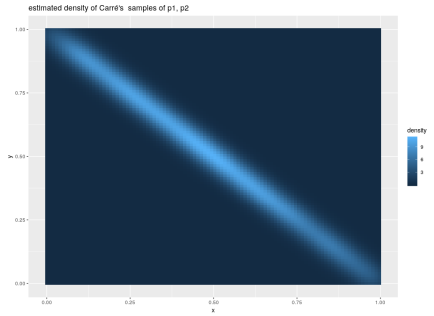


Figure 2.10: The estimated density of the samples p1, p2 drawn by Carré's method (200 chains, 200 steps, $\sigma = 0.1$). The samples are clearly biased towards the middle and not uniformly distributed

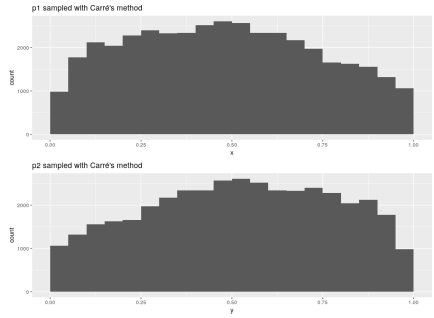


Figure 2.11: The estimated histograms of the samples p1, p2 drawn by Carré's method (200 chains, 200 steps, $\sigma = 0.1$). The samples are clearly biased towards the middle and not uniformly distributed

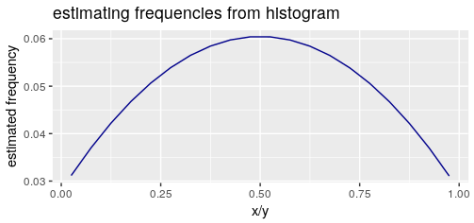


Figure 2.12: Estimated density from histogram of x, y as sampled by the Carré multiplicative random walk method

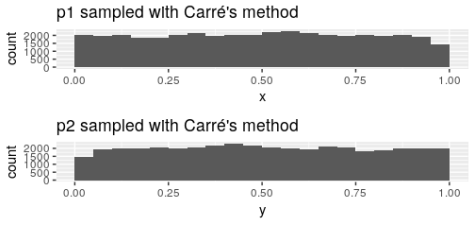


Figure 2.13: Approximately Corrected Density for Carré multiplicative random walk sampled values

2.3.2 Corrective Measures

We suggest to correct for this bias by reweighting the probabilities accordingly. As the steps are normally distributed and there is no closed-form to the normal distribution, we do not expect to find an analytical solution to the sampling distribution. Hence, we resort to an approximative correction. In particular, the frequency of any given sample is estimated with the aid of the histograms shown above; the density is fitted with a polynomial of degree 2^a; the result is shown in figure 2.12. This density is then used to correct the sampling density by a factor as follows:

[a] The actual estimated density obtained is $f(x) = 0.028 + 0.13 * x - 0.1301 * x^2$ in this case

$$\tilde{\text{samplingDensity}}(x) := \text{samplingDensity}(x)\,\frac{max_x\{f(x)\}}{f(x)}$$

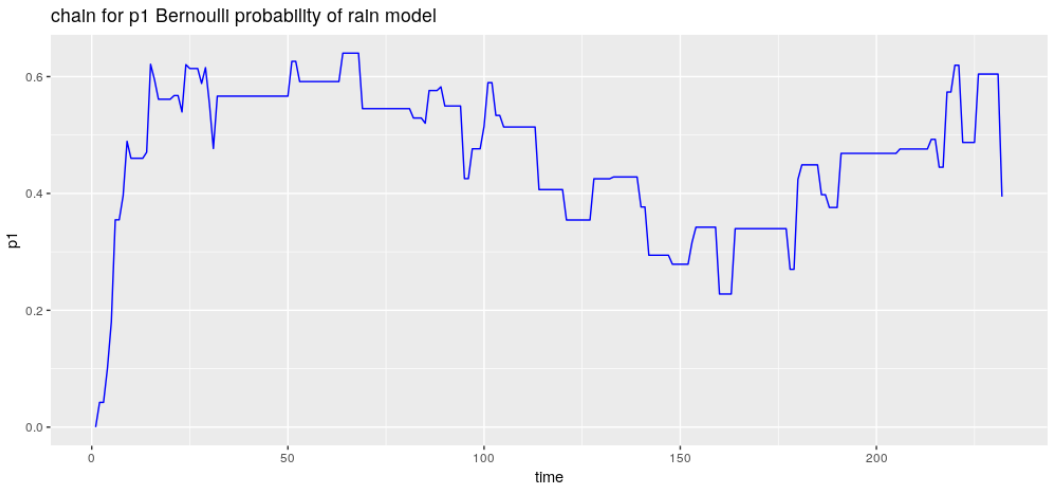


Figure 2.14: Chain is very sticky due to low acceptance ratio

Figure 2.13 clearly shows that the resulting distribution is indeed much closer to the uniform distribution.
Note that this simple, approximative method can be used for arbitrary sampling methods. While it is not necessary to correct for the "bumping" method, the "bumping" method alone can not be used to enforce contraints, for instance that the individual values some to one.

2.3.3 Final Implementation

The final implementation uses the follwing set of parameters:

- **Bernoulli probabilities**:
  Proposal $\mathcal{N}(p, \sigma = 0.1)$
- **Gamma probabilities**:
  1. select row to update uniformly
  2. apply Proposal $\mathcal{N}(\Gamma_{i,j}, \sigma = 0.08)$
  3. normalize sum of 1 by Cappé's method
- δ **probabilities**:
  1. apply Proposal $\mathcal{N}(\delta_i, \sigma = 0.03)$
  2. normalize sum of 1 by Cappé's method

All parameters are "bumped back" with the "Bumping method" as described above.

The parameters have been manually tweaked to yield an acceptance ratio of 0.23[14] for a model with $m = 2$ states, i.e. 6 parameters.
A sample chain with this configuration is shown in 2.14. Note that because of the very low acceptance ratio, the chain is quite "sticky" (i.e.: highly autocorrelated). This can be alleviated by applying a technique called *thinning*, which is simply leaving out every xth element of the chain. As the focus of this dissertation focuses on the runtime complexity of the algorithm, we do not use thinning.

Adjusting σ

When using Metropolis Hastings as sampling method, the choice of σ governs the acceptance rate. Atchadé[1] has shown that under certain conditions, for high dimensions, 0.23 is the ideal acceptance rate. If σ is chosen too small, the acceptance rate will be too high; conversely, if σ is too big, the acceptance rate will be lower. Especially as the complexity of the model increases, lower σ are in order[15]. We use the following very simple scheme to adjust σ:

Adjustment of σ

$f \leftarrow 1.0$
while algorithm is running **do**
  if acceptanceRate < 0.23 **then**
    $f \leftarrow \frac{f}{1.01}$
  else if acceptanceRate > 0.23 **then**
    $f \leftarrow f\,1.01$
  else
    $\sigma_{\text{adjusted}} \leftarrow \sigma_{\text{original}}\, f$
  end if
end while

The σ for sampling δ, Γ and λ are tuned manually; their respective magnitudes are retained as the algorithm above makes adjustments as all σ are scaled by the same factor f.

[14] this acceptance ratio is commonly considered to be optimal for high-dimensional Monte Carlo chains

[15] To understand why, consider an m-dimensional standard normal random variable with identity as the covariance matrix. If we consider any point at a fixed distance d from the mean, the density at this point will decrease as m increases, keeping d fixed. d can be thought of as a proxy for σ, so a Metropolis Hastings algorithm will produce less likely samples for the same σ as m increases.

Adjusting Convergence Criterion

As outlined in chapter 2.2.1, convergence is measured by the maximum (absolute) difference between respective quantiles of the last two parts of the chain. When this statistic falls below a predefined threshold, the chain is considered to have converged. Unfortunately, as model complexity increases, σ will be scaled down as explained in the preceding section. This has a direct effect on the expected distance the Metropolis Hastings algorithm is going to explore. Note that

$$\mathbb{E}\left[|X|\right] = \sigma\sqrt{\frac{2}{\pi}} \propto \sigma \quad \text{for} X \sim \mathcal{N}(0, \sigma^2) \tag{2.5}$$

i.e. the expected length of one step (i.e. the difference to the next proposal when using a normal distribution as proposal distribution) of the Metropolis Hastings Algorithm is directly proportional to σ. The smaller σ becomes, the shorter the steps of the chain.

As the number of dimensions grows and σ gets adjusted to be smaller, we expect the chain to move in smaller steps. Thus, the distance covered by the chain is much more likely to fall below the threshold even though the chain has not reached convergence yet. Let us underline this with a short example:

Suppose $\sigma = 0.08$ and the scaling factor is 0.001 with a threshold of 0.05. The expected stride length for one dimension - according to the formula above - without adjustment would then be ~ 0.064, which is way above the threshold. With adjustment, however, the expected stride length would be only ~ 0.000064. So an upper bound for the distance covered by the chain in 200 steps is $0.000064 \times 200 \sim 0.0128 < 0.05$[16].

The algorithm defined in this thesis begins to first evaluate the convergence criterion after the first 200 iterations. Following this argument, given the numbers above, we do not expect the chain to exceed the threshold - regardless of whether the chain has converged or not. This obviously renders our convergence criterion practically useless. Note that the numbers chosen in this example are extreme, but serve solely to illustrate the issue.

Figure 2.15 illustrates the issue with a stimulated dataset. Please note that the algorithm checks whether the chain has converged only after the first 200 iterations - the preceding part is considered to be the "burn-in" period. In this period, we expect the deviation to be very low in the beginning (as the chain has not explored much), then rise for a longer time and reaching a plateau until it finally falls when convergence is reached.

As sdFac is decreased considerably, we deduce that the initial stride size is too big, i.e. the acceptance rate was too low. Interestingly, we observe that the second bump

[16] Note that this upper bound widely overerstimates the actual expected distance covered.

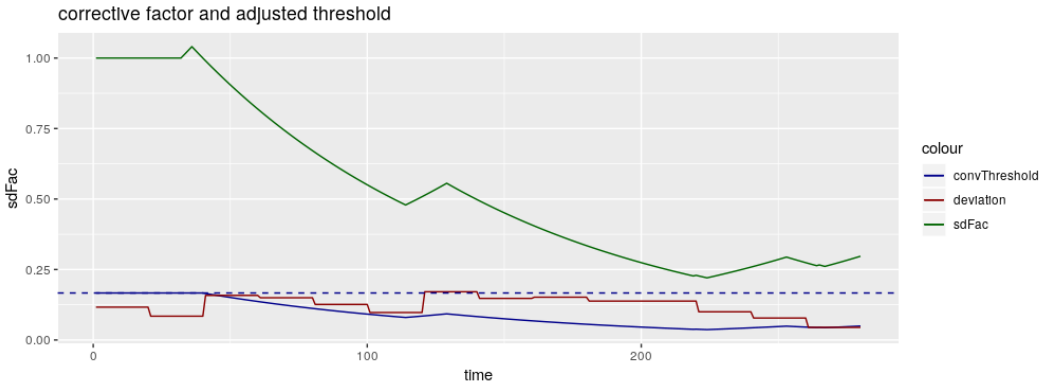


Figure 2.15: model: $m = 2$, 4 parameters to estimate, 2 parameters fixed; sample size $1,000$ generated by Bernoulli model generator.
convThreshold: adjusted threshold, if statistic falls below threshold, convergence is assumed to be reached; **deviation**: current statistic, updated every 20 samples, **sdFac**: corrective factor as defined in "Adjusting σ". convThreshold and sdFac have been scaled by $\frac{1}{0.3}$ to aid in visual inspection by amplifying differences. The original threshold $\frac{0.05}{0.03}$ is shown as a dashed blue horizontal line.

in sdFac coincides with a bump in deviation - apparently, due to an increased stride, the chain has explored a new realm here which changed the empirical distribution function significantly.
As expected, the convergence threshold is simply linearly dependend on sdFac.
Note that - judging from the original, unadjusted threshold - the chain would have been considered to have reached stationarity earlier than using the adjusted criterion (as it is below the original threshold already at exactly 200 iterations).
Note that this model is not even a full $m = 2$ model, as it estimates only 4 of the 6 parameters of the full model. Nonetheless, this example showed that even in this case, the adjustment of the convergence criterion is necessary.

2.4 Gibb's Sampler

The implementation herein follows the techniques presented in Zucchini[10].

2.4.1 Sampling of Hidden States

The states are sampled successively in a fixed order from $C_T, \ldots, C_1$.
Specifically, rely on the following for sampling:

$$P\left(C_t \,|\, C_{t+1}^T, x^T, \Theta\right) \propto P\left(X_t, C_t\right)\, P\left(C_{t+1} \,|\, C_t\right) = \alpha_t(i)\, \Gamma_{i,C_{t+1}}$$

$\alpha_t(i)$ is available from a forward-pass and $\Gamma_{i,C_{t+1}}$ is a constant given C_{t+1}. Hence,

the discrete probabilities for drawing C_t are readily available.

2.4.2 Sampling Γ

It is useful to establish a few properties of the Dirichlet distribution:

Lemma 3 *Let $X \sim Dir(\alpha_1, \ldots, \alpha_k)$ with $\alpha_i > 0$ and $\sum \alpha_i = 1$.*

Then $supp(X) = (x_1, \ldots, x_k)$ with $x_i \in (0, 1)$ and $\sum x_i = 1$. Furthermore we have

$$\mathbb{E}[X_i] = \frac{\alpha_i}{\sum_j \alpha_j} \tag{2.6}$$

In effect, the Dirichlet distribution can be used to sample discrete distributions with m components. As equation 2.6 shows, the distribution's parameters determine the expected value of the different components. Also the degree to which mass is centered around the means specified above can be controlled; the higher the distribution's parameters are in magnitude, the more likely are we to draw less peaked (i.e. more uniform) distributions[17].
Note that as the chain's length approaches infinity, the proposal distribution will hence also converge towards a uniform distribution. However, the literature consulted for this methodology suggests to indeed use absolute counts for estimates $\tilde{\Gamma}$ instead of converting those to relative probabilities. To be comparable to this original research, we also use absolute counts in this setting.
Γ is sampled in two steps:

- Firstly, given the current sequence of states, the entries are estimated as

$$\tilde{\Gamma}_{i,j} := \frac{\left|\left\{t \,|\, c_t = i \wedge c_{t+1} = j\right\}\right|}{\left|\left\{t \,|\, c_t = i\right\}\right|}$$

  Note that $\tilde{\Gamma}$ is set to 0 should the denominator be zero.

- Secondly, $\Gamma_{i,\cdot}$ is drawn as

$$\Gamma_{i,\cdot} \sim \text{Dir}(10 \times (\text{prior} + \tilde{\Gamma}_{i,\cdot}))$$

  where Dir is the *Dirichlet* distribution.

The prior can be chosen arbitrarily and defaults to $(m, \ldots (m \text{ times}))$.

[17] This notion is captured by the term *concentration parameter*

2.4.3 Sampling Bernoulli Probabilities

For Bernoulli probabilities, we apply a uniform prior on a naiive estimate. With

$$\begin{aligned} S_i &:= \{t \,|\, C_t = i\} \\ k &:= |\{t \,|\, C_t = i \wedge X_t = 1\}| \\ n &:= |\{t \,|\, C_t = i\}|\,, \end{aligned}$$

we have

$$\begin{aligned} X_t \sim Bern(\lambda_i) \quad \forall t \in S_i \\ \sum_{t \in S_i} X_t \sim \mathrm{Bin}(n, \lambda_i) \end{aligned}$$

Then, given n, k, we have $\lambda_i \sim \mathrm{Beta}(k+1, n-k+1)$[18].

2.4.4 Sampling Poisson Parameters

Sampling from a Poisson model is significantly more involved than a simple Bernoulli model. We will present the approach taken by Scott[9].
To increase tractability of the model and avoid a common problem known as *label switching*, Scott proposes to parameterise the Poisson distributions in terms of the *differences in* λ.
In particular, let us assume w.l.o.g. that the natural parameters $\lambda_1, \ldots, \lambda_s$ are ordered, i.e. $\lambda_i < \lambda_j$ for $i < j$. Then define $\tau_i := \lambda_i - \lambda_{i-1}$ with $\tau_1 \equiv 0$ by definition, i.e. we parameterise the differences in rates instead of the absolute rates. Note that by summing those individual differences, all of the original rates can be obtained. Note that

$$X + Y \sim Poi(\lambda_1 + \lambda_2) \quad \text{if } X \sim \mathrm{Poi}(\lambda_1), Y \sim \mathrm{Poi}(\lambda_2)$$

which provides a mathematical foundation for this decomposition.
Hence, we work with Poisson distributions whose rates correspond to the differences in rates of the original distributions. Let us shortly denote the former as "working distributions".Summing the first i working distributions will hence yield the ith original distribution. From now on, we will almost exclusively refer to the "working distributions". Hence, insofar obvious from context, we will simply call them "distributions" as well.

[18] This can be easily verified by comparing the respective densities.

Regimes

Let us further define *regimes*; a regime describes which portions of the distributions are active at any given time. The ith regime includes the first i distributions. Hence, the first ith regime is said to be *active* if and only if the first i distributions are active.
In particular, let $1 \leq h_t \leq s$ denote the regime being active at time t, let

$$d_{it} = \begin{cases} 1 & \text{iff } i \leq h_t \\ 0 & \text{otherwise} \end{cases}$$

denote whether distribution i is active at time t (it is active iff d_{it} assumes 1).
Note that this notation ties in neatly with our previous definitions: in particular, $C_t = i \iff h_t = i$[19].
Furthermore, we assign each (working) distribution a "contribution" toward any observation: Let x_{it} (and X_{it} respectively) be the contribution of the ith distribution toward x_t.

Sampling Contributions

Let us establish the following non-obvious result:

Lemma 4

$$P\left(X_{jt} = x_{jt}, 1 \leq j \leq s\right) \sim MN(\eta_1, \ldots, \eta_s)$$

with MN denoting the Multinomial Distribution and

$$\eta_i := \frac{\tau_i}{\sum_k^{h_t} \tau_k}$$

Proof 2 *As $X_{jt} \sim Poi(\tau_j)$, we have*

$$P\left(X_{jt} = x_{jt}\right) = \tau_j^{x_{jt} d_{jt}} \frac{e^{-\tau_j d_{jt}}}{x_{jt}!} \quad \textit{hence}$$

$$P\left(X_{jt} = x_{jt}, 1 \leq j \leq s\right) = \begin{cases} \prod_j \frac{e^{-\tau_j d_{jt}}}{x_{jt}!} & \textit{if } \sum_j x_{jt} = x_t \\ 0 & \textit{otherwise} \end{cases}$$

[19] in fact, we only introduce this additional notation to be consistent with Scott

With $\sum_j x_{jt} = x_t$ $\left(\sum_j X_{jt} = X_t\right)$ *we have*

$$P\left(X_t = x_t\right) = \left(\sum_{j=1}^{h_t} \tau_j\right)^{x_t} \frac{e^{-\sum_j^{h_t} \tau_j}}{x_t!}$$

So altogether we have

$$\begin{aligned}
P\left(X_{jt} = x_{jt}, 1 \leq j \leq s \,|\, X_t = x_t\right) &= \frac{P\left(X_j t = x_j t \wedge X_t = x_t\right)}{P\left(X_t = x_t\right)} \quad \textit{omitting indices j=1, 2, ..., s} \\
&= \frac{x_t!}{\prod_j x_{jt}!} \frac{\prod_j \tau_j^{x_{jt} d_{jt}}}{\left(\sum_{j=1}^{h_t} \tau_j\right)^{x_t}} \\
&= \frac{x_t!}{\prod_j x_{jt}!} \quad \prod_j \left(\frac{\tau_i}{\sum_{k=1}^{h_t} \tau_k}\right)^{x_{jt} d_{jt}} \\
&= \frac{x_t!}{\prod_j x_{jt}!} \quad \prod_j \left(\eta_j\right)^{x_{jt} d_{jt}}
\end{aligned}$$

as claimed.

Note that as h_t sampled by sampling C_t; this is accomplished by sampling backwards as outlined above. Hence, d_{jt} is known and x_{jt} may be sampled using aforementioned distribution.

Chapter 3

Results

3.1 Model Generation

For the purposes of experimentation, we implemented a simple model generator. The generator generates a model of given complexity m as follows:

- $\delta \sim \mathrm{Dir}(1, 0.5, 0.5, \dots)$
- $\lambda_i \sim U[0, 1]$
- $\Gamma_{i,\cdot} \sim \mathrm{Dir}(0.5, \dots, \Gamma_{i,i-1} = 0.5, 1, \Gamma_{i,i+1} = 0.5, \dots, 0.5)$

For benchmarking the algorithms, the following procedure was used:

Benchmarking Algorithms

$noParamsToEstim \leftarrow \{2, \dots, 6\}$
$ss \leftarrow \{500, 1000, 1500, 2000\}$
modelsEachSize $\leftarrow 3$
runsEachModel $\leftarrow 3$
for comb $\in$ noParamsToEstim $\times$ ss $\times$ modelsEachSize **do**
    $model \leftarrow$ generateModel($comb$)
    for r $\in$ runsEachModel **do**
        performRun
    end for
end for

3.2 Gibbs

Figure 3.1 shows the dependency between quality of the chains and model complexity. Interestingly, we observe that quality does *not* decrease as the model complexity (i.e.

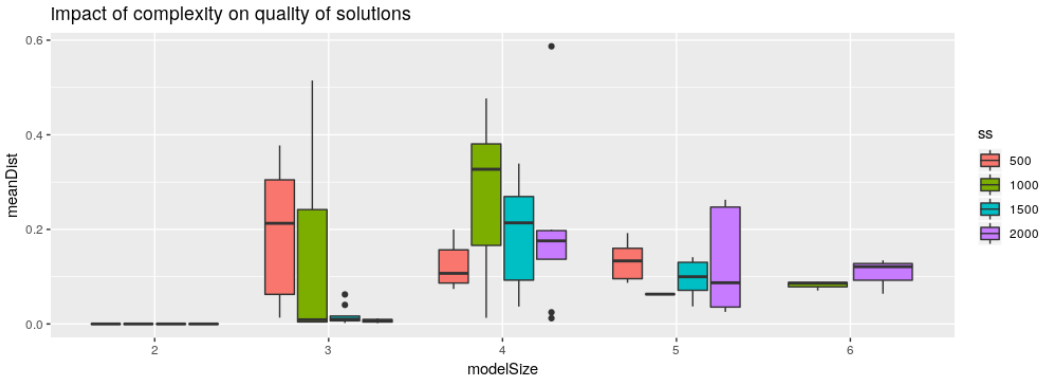


Figure 3.1: MeanDist is defined as the minimum Manhattan distance ($L_1(\mathbb{R})$) between λ of the model sampled from and the mean from the last third of the chain after convergence. Generally speaking, more samples produce solutions of higher quality

parameters to estimate) increases.

Unsurprisingly, the greater the sample size, the closer the estimates are to the original model. Note, however, that sampling size 500 appears to be an exception, as - for those benchmarks - it produces solutions of about equal quality as for higher sample sizes. This can be explained by the fact that for simple models, small sample sizes are sufficient and for more complex models, Gibbs's sampler for small sample sizes simply does often not converge. This is illustrated in figure 3.2. So, *if* the Gibb's sampler converges for a small sample size, its quality is good. However, for more complex models, it does not (sufficiently) converge in the first place. Please note that in figure 3.1, only chains which have converged (i.e. the algorithm aborted before reaching $3,000$ elements) have been considered.

This begs the question whether small sample sizes should be used at all. Figures 3.1 as well as 3.3 provide the relevant information: While a smaller sample sizes yield a stationary distribution faster, when taking into account the empirically obtained convergence rates, the advantages in terms of time spent do not outweigh the inferiority in convergence rates. Note in particular that the algorithm failed to converge for any of the $3 \times 3 = 9$ runs for the full 6 parameter model if a sample size of only 500 was used. Note that this may be due more because of *unidentifiability* of the model than because of a shortcoming of the algorithm or implementation itself. Figure 3.4 shows inconclusive evidence that more samples decrease the number of iterations necessary. Note that increasing sample sizes increase the duration of each iteration (at the magnitude of $\mathcal{O}(T)$). On the other hand, more samples increase *identifiability* of the model, which make each step more informative and could reduce the number of iterations necessary. Hence the time spent until convergence does not directly translate to iterations computed.

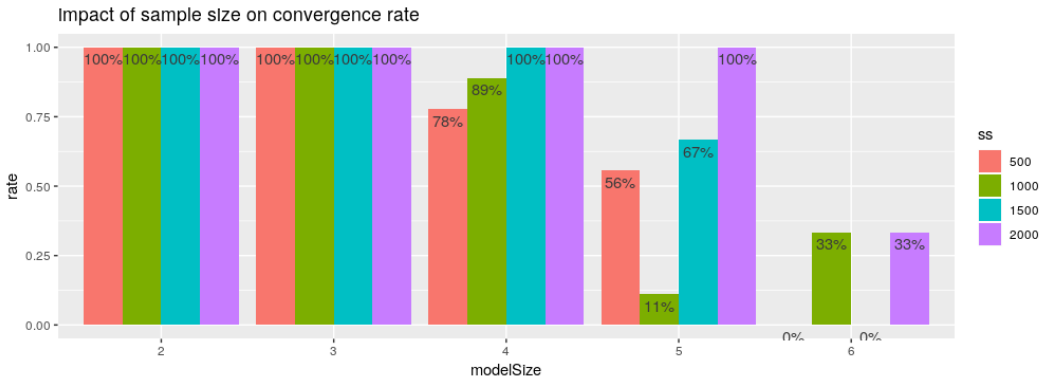


Figure 3.2: Convergence is defined as the max difference inquantiles between the second and third part of the chain falling below the threshold of 0.1 for the quantiles $0.25, 0.5$ and 0.75. More complex models yield lower convergence rates; the cut-off point is $3,000$ samples.

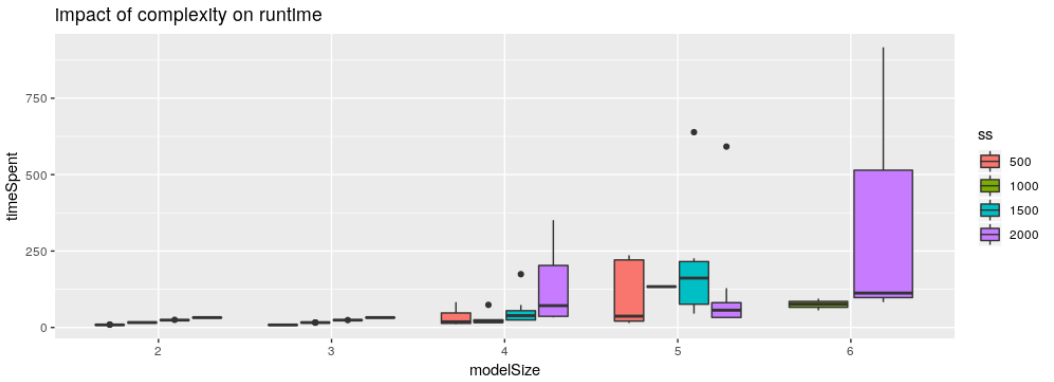


Figure 3.3: The more complex the model, the more time the Gibb's sampler's chain needs until convergence. Missing bars indicate non-convergence of respective chains.

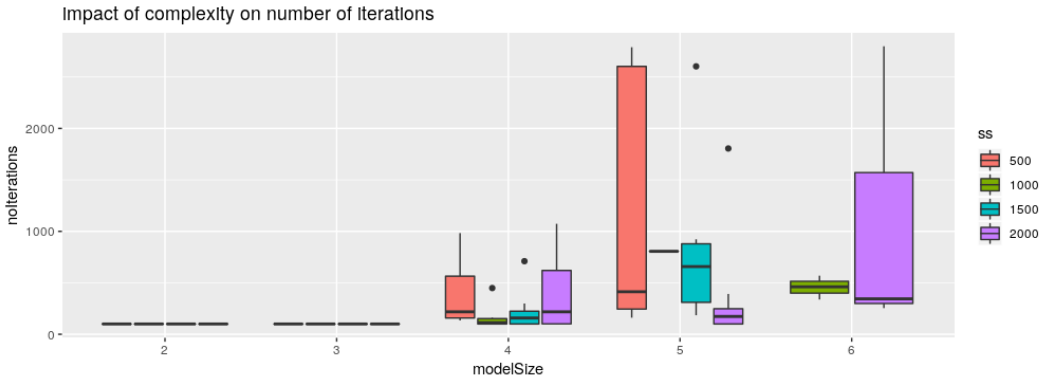


Figure 3.4: More complex models cause longer chains, but more samples also cause longer chains

So while figure 3.1 clearly shows that the quality of the estimation increases with increasing sample size, figure 3.4 also shows that still more iterations are needed.

3.3 Metropolis-Hastings

Figure 3.5 shows that the quality of the solutions decreases when the model complexity increases. Note how this is markedly different from the Gibb's sampler's results. Generally speaking, the graphs exhibit much more variation, which is due to the relatively small sample size of 3 models with each 3 runs. Already judging from this figure, we can see that running several chains with Metropolis Hastings in parallel to decrease variance is a good option.
Figure 3.6 shows that the run time of Metropolis-Hastings is generally lower than of the respective Gibb's sampler. However, more samples seem to have a *negative* effect on the runtime. This suggests that in the setup used in this thesis, the data is not sufficiently informative for the Metropolis-Hastings steps.
Also note that no chains for model complexity 6 are shown. This is because none of the chains have converged within the given limit of iterations $(3,000)$; this fact is corroborated by figure 3.7. All of these effects are not because the chain actually fails to converge towards a stationary distribution; it just does not do so within the time constraint given. This has been verified by manual inspection. That the time limit is quickly reached for complex models becomes evident when examining figure 3.8; When naiively extrapolating the number of iterations depicted therein, we can clearly see that for a model with 6 three parameters, the limit of $3,000$ chain samples is quickly exhausted.

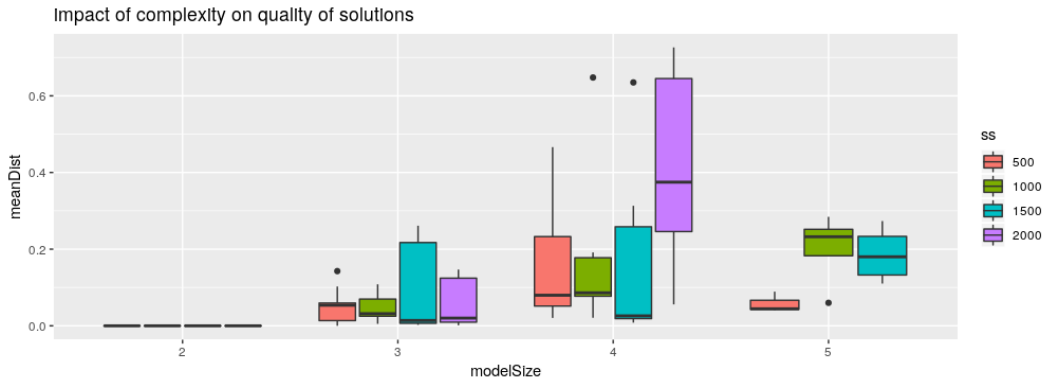


Figure 3.5: Solution quality decreases as model complexity increases; behaviour is highly erratic and model takes very long to converge. No chain converged for the full model given the time constraints.

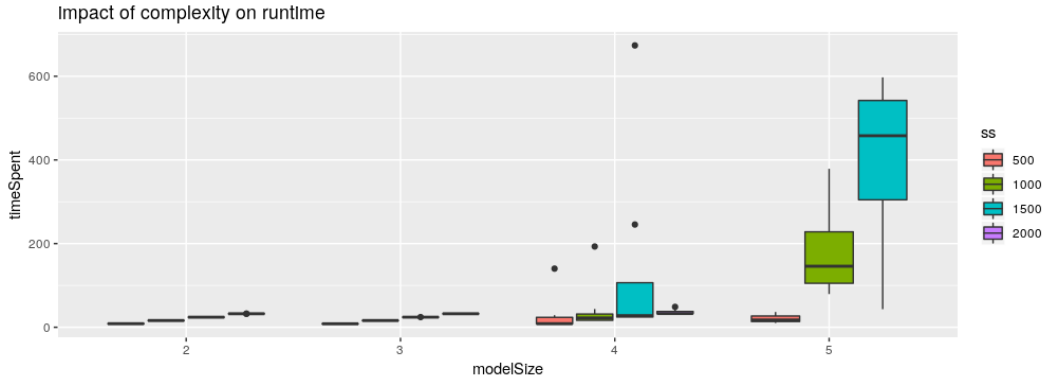


Figure 3.6: Runtime increases dramatically with model complexity and appers to rise exponentially in terms of sample size

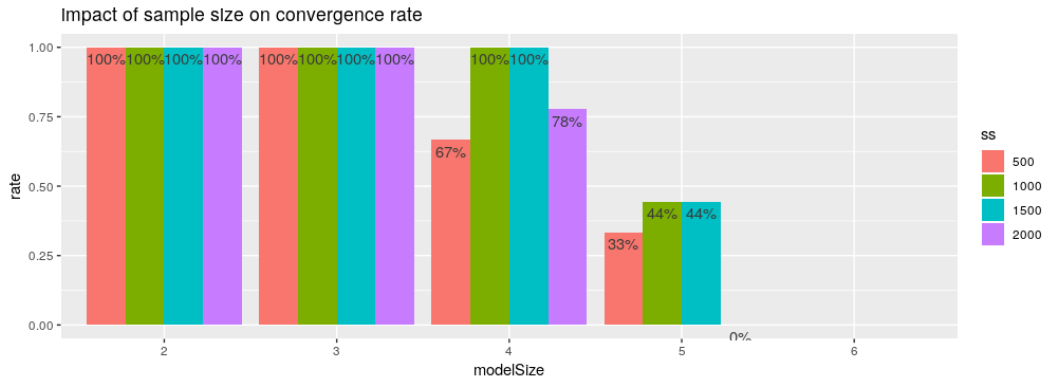


Figure 3.7: Convergence rates decrease dramatically as model complexity increases; this is due to the number of iterations being limited to $3,000$.

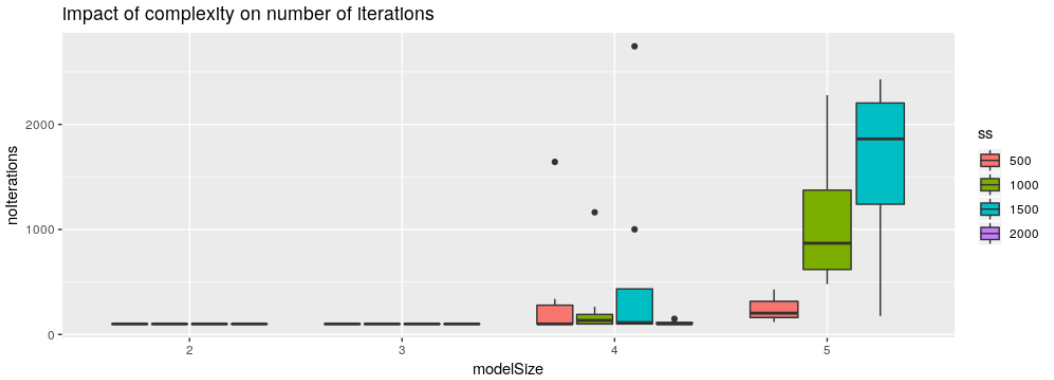


Figure 3.8: The number of iterations needed increases with model complexity; it also increases excessively with the number of samples

Effective Sample Size

The effective sample size has been sampled several times; for both Metropolis-Hastings as well as the Gibb's sampler, the effective sample size was around 10%. Let us shortly break down the total sample size using this number:
Assuming a chain length limit of $3,000$ and convergence after $1,000$ iterations, we extract the last third of the chain, which has about 300 samples. Let us assume that the least probable state - according to the stationary distribution - has a probability of about 0.3 [1]. Hence we expect the state to be assumed about 100 times in those samples; having an effective sample size of 10% then leaves us roughly a mere 10 samples to estimate the state's probability.
This example shows us how an initial sample of $1,000$ samples can shrink drastically to about 1% of it's size for the estimation of individual parameters. Hence it becomes obvious that it is meaningful to let the chain run long after convergence has been reached. Furthermore, this example underlines that our convergence criterion is indeed quite strict, as a high number of conditions (convergence of three quantiles up to 0.01) is used on a relatively small number of samples.
Please note that the effective sample size is *not* a proper proxy to estimate the number of equivalently independent drawn samples used to estimate individual parameters. Hence, this section merely outlines at what magnitudes the different effects might operate and especially which toll they take on our sample sizes.

[1] Note in particular that this applies to the RainModel which is used in the "applyOnRainModel" files for both Metroppolis Hastings as well as Gibb's Sampler.

3.4 Conclusion

This dissertation examined vanilla implementations of Metropolis Hastings and Gibb's samplers on Bernoulli models. It gave a brief account of the underlying theory, described a basic model, elaborated on practical pitfalls for implementation and drew a comparison based on practical simulations.
Altogether, the results obtained herein paint a clear picture; the Gibb's sampler is superior to the Metropolis-Hastings sampler in almost any respect.
The mathematical work required to implement a Gibb's sampler is *much* more extensive than for a Metropolis-Hastings sampler, and so are its requirements. For instance, a prerequisite for the Gibb's sampler is that it is possible to sample from the marginal distributions. This may not always be the case. What is more, the Gibb's sampler requires in-depth knowledge of the underlying model - for instance, it estimates all hidden states at each step. The Metropolis-Hastings sampler, however, is somewhat agnostic of these details. When those prerequisites are fulfilled, however, sections 3.2 and 3.3 clearly show that Gibb's sampler is superior with respect to most criteria.
Last but not least, while the Metropolis-Hastings sampler requires only minimal knowledge of the model it is sampling from, its robustness is much more fragile than Gibb's sampler. Noting that the acceptance rate of the Gibb's sampler is 1, this is hardly surprising. This means that while virtually no model-specific knowledge is required, there is extensive extra work necessary to make Metropolis-Hastings work in general settings. The additional steps introduced, however, are usually not model-specific, but more data-specific. This means that they present a challenge which is specific more to dealing with different types of solution surfaces than with different types of models (and their internal structure). Hence, these challenges can be tackled in a generalised way, whereas Gibb's sampler always requires model-specific implementation.

3.5 Prospect

The dissertation deliberately relied on simple implementations of both Gibb's and Metropolis-Hastings. In practice - and in almost all papers consulted for this dissertation - both samplers were used only on a very select number of models. Then, the parameters of both models are usually manually tweaked to obtain optimal results and problems (and convergence) are diagnosed manually be graphical inspection.
This usually manually applied approach is not viable for a comparison as conducted in this dissertation, yet very relevant for practical use. Case-specific analysis will show whether one or another model can excel in specific areas or under specific

conditions.
In particular, the effect of prior knowledge in the form of priors can be used to aid estimation. The type of prior used depends - of course - on the specific data and model evaluated. Again, model-specific optimisation is in order.

Bibliography

[1] Yves F. Atchadé, Gareth O. Roberts, and Jeffrey S. Rosenthal. Towards optimal scaling of metropolis-coupled markov chain monte carlo. *Statistics and Computing*, 21(4):555–568, Oct 2011. ISSN 1573-1375. doi: 10.1007/s11222-010-9192-1. URL https://doi.org/10.1007/s11222-010-9192-1.

[2] Lothar Breuer. *Introduction to Stochastic Processes*. Lecture Notes, University of Kent.

[3] Oliver Cappe, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. 01 2005. ISBN 978-0-387-28982-3. doi: 10.1007/0-387-28982-8.

[4] Olivier Cappé, Christian P. Robert, and Tobias Rydén. Reversible jump, birth-and-death and more general continuous time markov chain monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 65 (3):679–700, 2003. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3647545.

[5] J. Andrés Christen and Colin Fox. A general purpose sampling algorithm for continuous distributions (the t -walk). *Bayesian Anal.*, 5(2):263–281, 06 2010. doi: 10.1214/10-BA603. URL https://doi.org/10.1214/10-BA603.

[6] Dr Adam M. Johansen. *ST407 Monte Carlo Methods*, volume 4. 2019. Accompanying lecture notes to ST407 Monte Carlo Methods.

[7] Christian P. Robert. Simulation of truncated normal variables. *Statistics and Computing*, pages 121–125, 1995.

[8] Tobias Rydén. Em versus markov chain monte carlo for estimation of hidden markov models: a computational perspective. *Bayesian Anal.*, 3(4):659–688, 12 2008. doi: 10.1214/08-BA326. URL https://doi.org/10.1214/08-BA326.

[9] Steven L Scott. Bayesian methods for hidden markov models. *Journal of the American Statistical Association*, 97(457):337–351, 2002. doi: 10.1198/016214502753479464.

[10] Walter Zucchini and Iain Macdonald. *Hidden Markov Models for Time Series: An Introduction Using R*. 04 2009. ISBN 9781420010893. doi: 10.1201/9781420010893.