Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Bayesian Analysis of Transcriptional Dynamics

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Gonzalo Benegas

Director: Alejo Salles

Codirector: Hernán García

Buenos Aires, 2018

Abstract

The development of multicellular organisms requires precise control of gene expression across time and space. Recently developed experimental techniques are allowing for monitoring live transcriptional activity in the fruit fly embryo, by fluorescently tagging nascent mRNA molecules at specific genes. By extracting fluorescent traces from individual nuclei, transcriptional activity can be quantified across time and space in the developing embryo. Analyzing data from the *eve* gene, we encounter transcription occurs in random bursts, rather than at a predictable rate. We develop a probabilistic model to account for this phenomenon, based on the transition of genes between different ON and OFF transcriptional states. To infer these latent states, we develop a Reversible–jump Markov chain Monte Carlo algorithm, able to handle parameter spaces of varying dimensionality. Our initial results suggest transition rates are modulated across the embryo, and contribute to the establishment of the observed expression pattern. The line of work we have started opens interesting possibilities for further inquiry into the mechanisms of transcriptional regulation in the future.

Keywords: gene regulation, transcriptional bursting, Bayesian Inference, Reversible–jump Markov chain Monte Carlo

Resumen

El desarrollo de organismos multicelulares requiere del preciso control de la expresión genética a lo largo del tiempo y del espacio. Técnicas experimentales recientes permiten monitorear en vivo la actividad transcripcional en en el embrión de mosca, mediante el etiquetado fluorescente de moléculas de ARN naciente en genes específicos. A partir de la extracción de trazas fluorescentes de cada núcleo, se puede cuantificar la actividad transcripcional a lo largo del tiempo y el espacio durante el desarrollo embrionario. Al analizar datos del gen *eve*, encontramos que su transcripción ocurre en ráfagas. Desarrollamos un modelo probabilístico de este fenómeno, basado en la transición del gen entre diferentes estados transcripcionales, activos e inactivos. Para inferir estos estados subyacentes, desarrollamos un algoritmo de Reversible–jump Markov chain Monte Carlo, capaz de soportar espacios de parámetros de dimensión variable. Los resultados preliminares sugieren que las tasas de transición entre estados son moduladas a lo largo del embrión, contribuyendo al establecimiento de los patrones de expresión observados. Esta línea de trabajo abre interesantes posibilidades para seguir investigando los mecanismos de regulación transcripcional en el futuro.

Palabras clave: regulación génica, ráfagas de transcripción, Inferencia Bayesiana, Reversible–jump Markov chain Monte Carlo

Agradecimientos

A Silvina Ponce Dawson y Pablo Turjanski, por su tiempo y disposición para ser jurados de esta tesis.
A mis directores Alejo Salles y Hernán García, por su compromiso y guía en los momentos más lentos y más rápidos a lo largo de este proyecto.
A la Universidad de Buenos Aires, la Facultad de Ciencias Exactas y Naturales y al Departamento de Computación, por formar un ambiente tan enriquecedor para estos años de mi vida.

Outline

Chapters 1 and 2 are introductory. Chapter 1 provides some biological background, and presents the statistical problem of inferring transcriptional dynamics from live imaging data. Chapter 2 introduces Bayesian inference, the statistical framework in which we frame the problem, and Markov chain Monte Carlo, the general class of algorithms we use to perform inference.

In Chapters 3 and 4 we present a new method to solve the problem at hand. In Chapter 3 we define in detail a probabilistic model of transcriptional dynamics, under the Bayesian framework. Chapter 4 presents a Reversible–jump Markov chain Monte Carlo algorithm, designed to infer the parameters of our Bayesian model.

In Chapters 5 and 6 our new method is applied to different datasets. Chapter 5 describes the evaluation of the performance of the algorithm using synthetic data. In Chapter 6, we run the algorithm on experimental data and analyze the results.

Chapter 7 presents the conclusions of the current work and discusses new directions for the future.

Contents

Chapter 1

Biological background

Living organisms are very complex systems, with the remarkable ability to make copies of themselves. Their reproducibility is made possible by the encoding of development and functioning in a simple digital language, DNA, a long chain composed of four nucleotide subunits. The central dogma of molecular biology describes how information flows from DNA to proteins, large versatile biomolecules that carry out most of the functions in the cell (Figure 1.1).

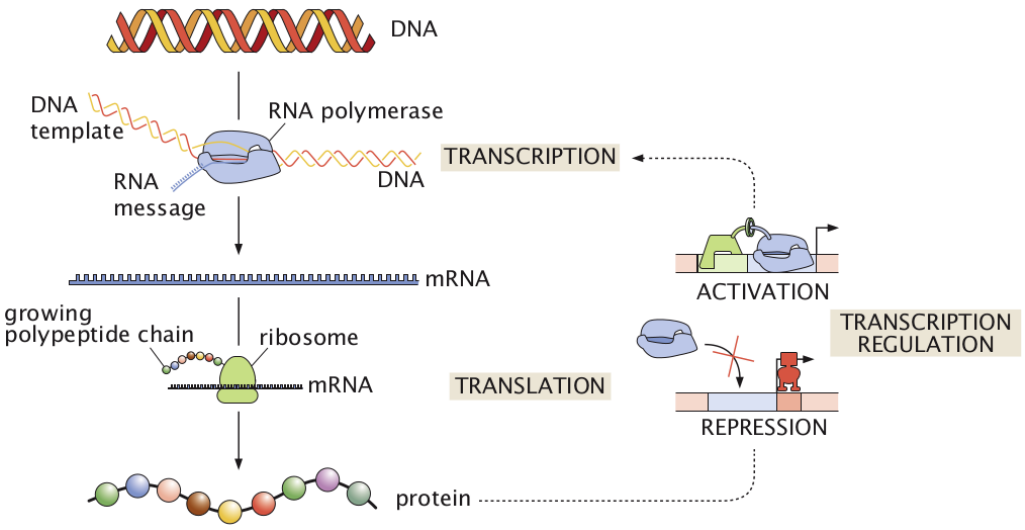


Figure 1.1: The central dogma of molecular biology describes the flow of information from DNA to RNA to protein. Proteins can have diverse roles, including regulating the transcription of genes.

A sequence of DNA, called gene, is transcribed into another nucleotide sequence, messenger RNA (mRNA), by a molecule called RNA polymerase. mRNA is later translated at ribosomes into proteins, sequences of peptides that acquire complex three–dimensional shapes. Proteins can take many roles, for instance, Hemoglobin carries oxygen in red blood cells. Equally as important than to know how to build certain proteins, it is important to know *when* and *where* to build them. Some genes

encode proteins with the sole function of regulating the expression of other genes. Figure 1.2 shows a complex gene regulatory network arising in the development of the sea urchin embryo, resembling a computer circuit.

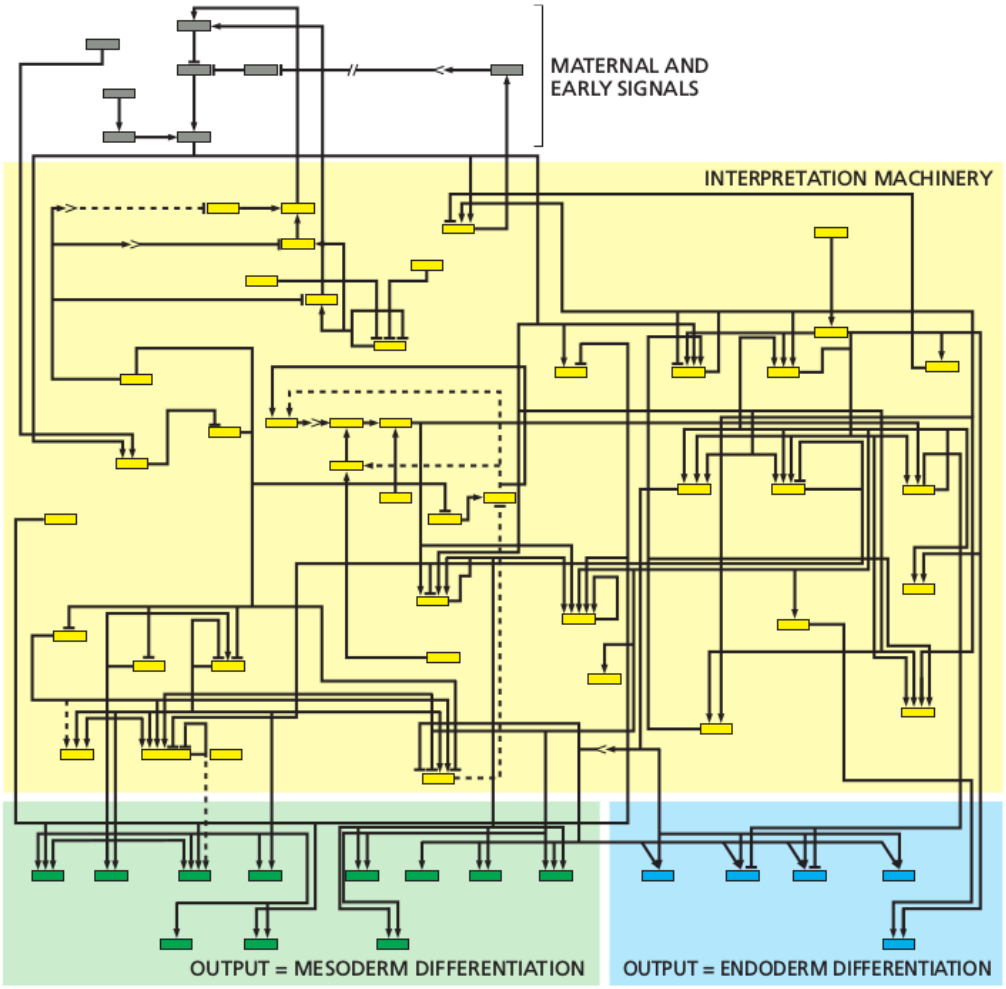


Figure 1.2: Complex gene regulatory network in the developing sea urchin embryo. Each small box represents a gene. Genes in gray are the input of the system, genes in yellow are transcriptional regulators (the logic of the system), and genes in green and blue are outputs. Arrows indicate a positive transcriptional control, while lines ending in bars indicate negative transcriptional control. (Figure from [A+14])

Transcription is the first step of gene expression, and as such, it is where a large part of regulatory logic is focused. Before starting transcription, RNA polymerase must bind to a DNA region called the *promoter*, situated next to the gene. The rate of binding of RNA polymerase at the promoter depends on the interaction of a large number of proteins, called transcription factors (Figure 1.3). The promoter is part of a larger DNA section, the control region for the gene, that specifies which transcription factors can bind to it and control the rate of transcription initiation.

In principle, combinatorial binding of transcription factors allows for multiple rates of transcription.

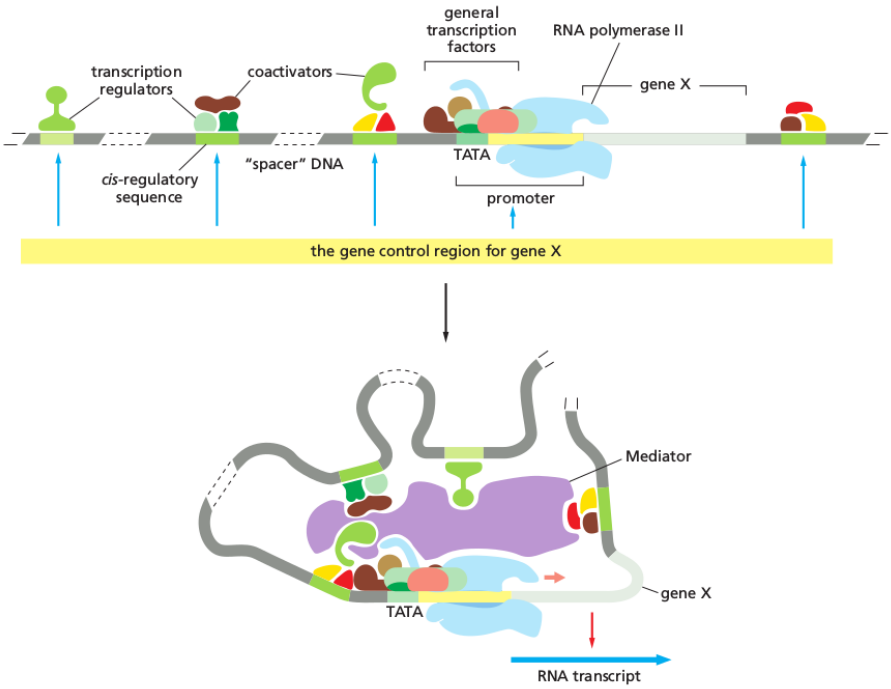


Figure 1.3: Typical control region of an eukaryotic gene. Multiple proteins interact with RNA polymerase and affect its binding to the promoter. The looping of DNA allows proteins bound to distant regions to participate in the regulation of the gene (Figure from [A+14])

1.1 Development of the fruit fly embryo

The developing embryo of the fruit fly *Drosophila melanogaster* is a model system for studying gene expression and its regulation. In the early embryo, multiple nuclei divide rapidly in a common cytoplasm, called synctium. This enables the propagation of regulatory signals via the free diffusion of transcription factors—there is no need for complex cell–to–cell communication.

One of the early signals in the developing embryo is initiated by the deposit of *bicoid* mRNA by the mother, before fertilization, at the anterior end (Figure 1.4). Upon fertilization, this mRNA is translated locally and the protein product diffuses along the embryo, establishing a concentration gradient along the anterior–posterior (AP) axis. Bicoid protein is a transcription factor, and its uneven concentration provides individual nuclei a cue of their spatial position along the AP axis.

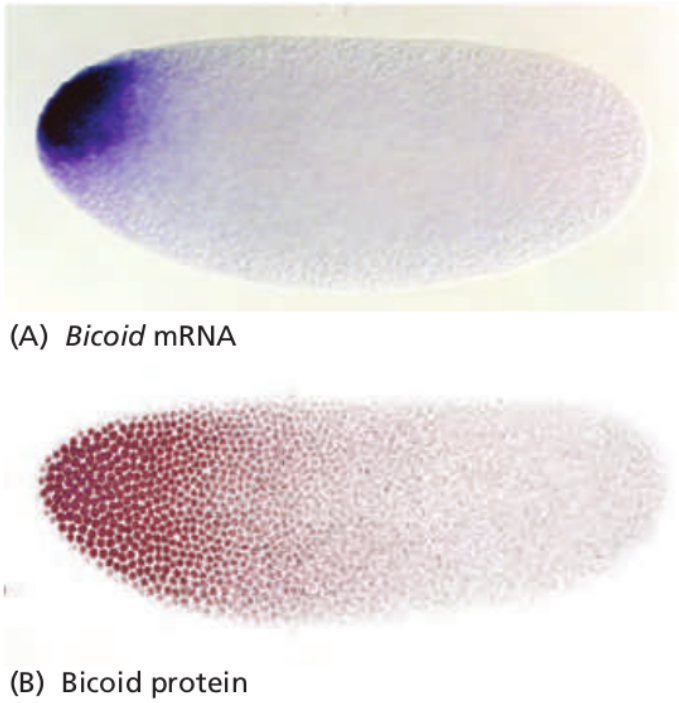


Figure 1.4: (A) *Bicoid* mRNA is deposited by the mother at the anterior end, before fertilization. (B) Upon fertilization, mRNA is translated and Bicoid protein diffuses establishing a gradient along the AP axis. (Figure from [A+14])

The expression pattern of the *even–skipped* (*eve*) gene, forming seven stripes, is one of the most well–studied in developmental biology. In particular, the regulatory elements involved in the formation of the second stripe have been identified (Figure 1.5). The stripe appears in nuclear cycle 14 (i.e., after 14 rounds of cell division). Transcription is confined to the region along the AP axis where two input activators are present at a high concentration and two repressors are present at a low concentration. This striped expression pattern is fundamental to the development of body segments in the fly. Interestingly, this stripe only exists for a few minutes (Figure 1.5 (D)), a discovery made possible by recently developed techniques, described in the next section.

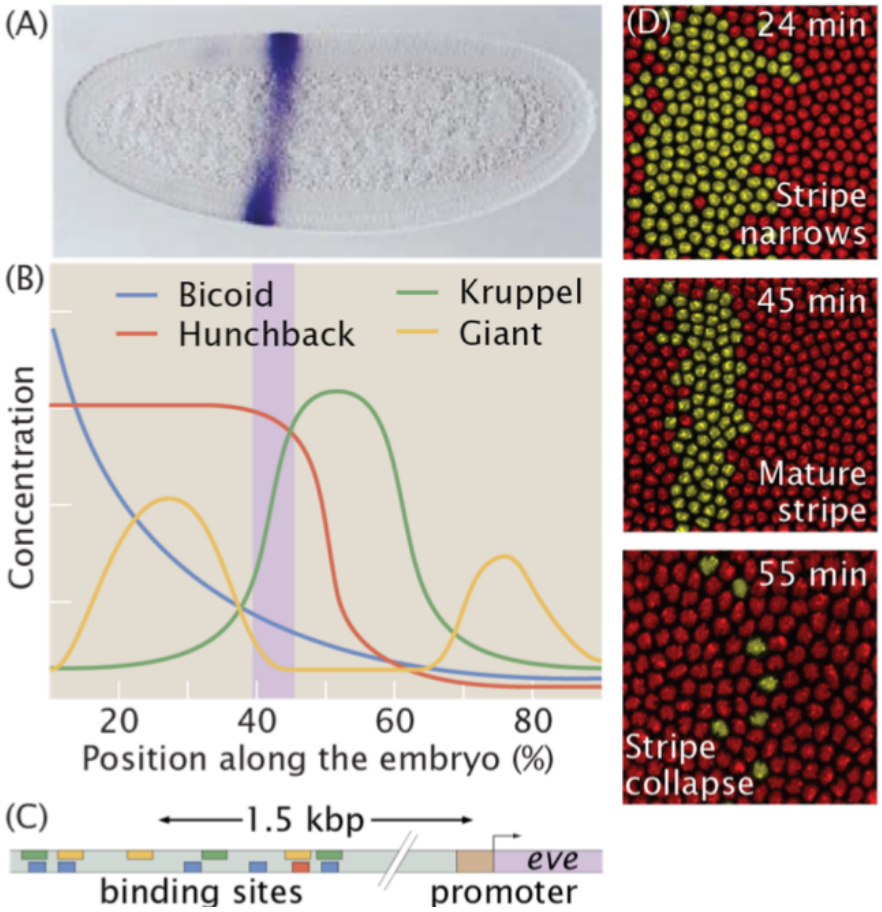


Figure 1.5: (A) Static view of the *eve* stripe 2 expression pattern. (B) Concentration of transcription factors regulating *eve* transcription and defining the second stripe. Bicoid and Hunchback are transcriptional activators, while Giant and Kruppel are transcriptional repressors. (C) DNA region regulating *eve* transcription, containing binding sites for the transcription factors. (D) Dynamic view of the *eve* stripe 2 expression pattern, obtained using the technique described in Section 1.2. Time markings are relative to the start of nuclear cycle 14 (nc 14).

1.2 Live imaging of transcriptional activity

A recently developed method allows for monitoring transcriptional activity in live Drosophila embryos [GTLG13]. This technique is based on fluorescently tagging nascent mRNA molecules at a specific gene, using the MS2-GFP system (Figure 1.6). GFP is dispersed all along the embryo, but there will be localized concentrations at the target genes while they are being actively transcribed (Figure 1.7).

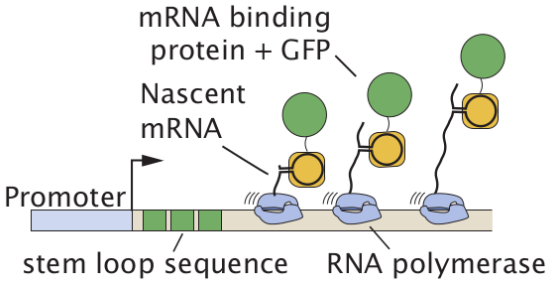


Figure 1.6: A DNA sequence is added at the start of a gene, consisting of 24 repeats of the MS2 stem loop. Each transcribed loop can bind to a GFP (Green Fluorescent Protein) molecule.

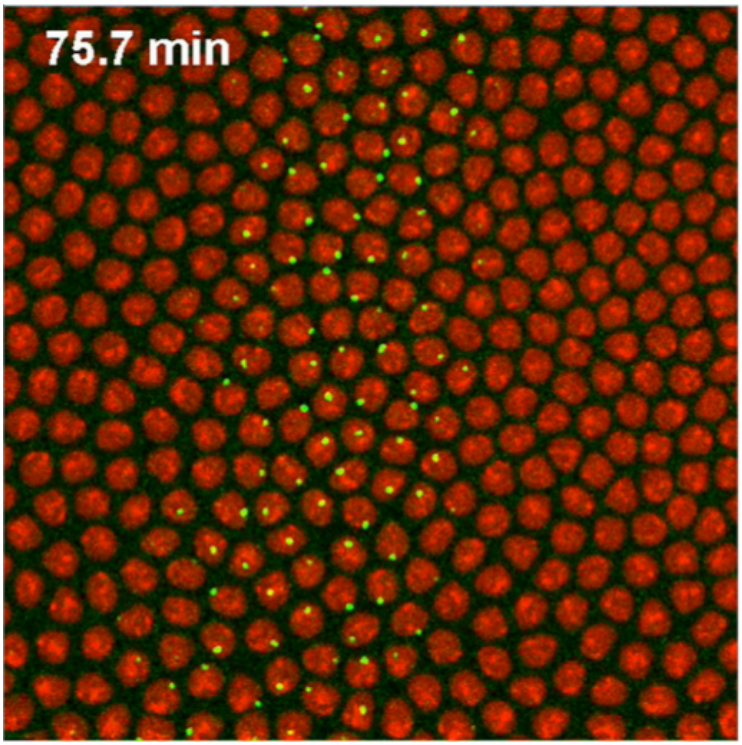


Figure 1.7: Snapshot from movie in [BGE+14], obtained using confocal microscopy. Focus is on *eve* stripe 2, 75.7 min from the start of nuclear cycle 14. Red circles identify individual nuclei, while green spots mark nascent *eve* mRNA.

The intensity of the fluorescence at each individual spot is proportional to the number of mRNA molecules being transcribed, and thus is a way to quantify transcriptional activity across time and space in the embryo. For details about the image processing, see [BGE+14]. It is important to note that to infer the fluorescence produced by nascent mRNA molecules at the gene, it is necessary to subtract the background fluorescence. Estimation of the background fluorescence is the biggest contributor to measurement error. By tracking nuclei and fluorescence spots, time

traces of transcriptional activity for each nucleus can be obtained (Figure 1.8). An additional remark: sometimes two close spots are identified in a nucleus, due to the presence of two copies of the gene, one for each sister chromatid.

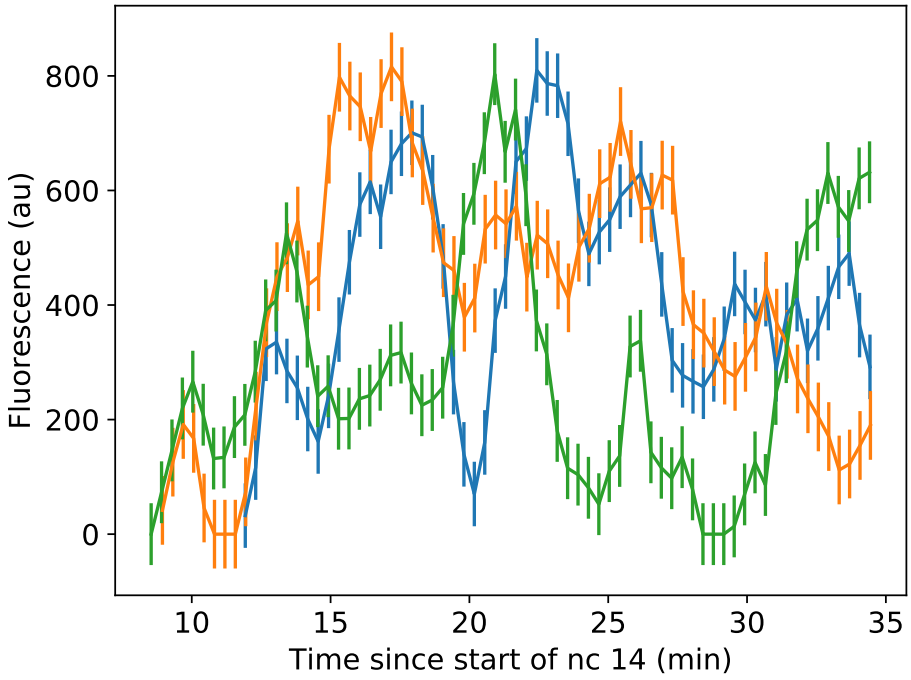


Figure 1.8: Three fluorescent traces extracted from nuclei at the center of *eve* stripe 2, showing heterogeneous transcriptional activity. Time resolution is 22.5 s.

1.3 Transcriptional bursting

Fluorescent traces have revealed a great heterogeneity in transcriptional dynamics among different nuclei—even when they are close to each other, with a similar concentration of transcription factors (Figure 1.8). It seems that transcription occurs not with a predictable rate, but rather in bursts of a random character.

These pulses of fluorescence can be understood with the simple model illustrated in Figure 1.9. Initially, the gene is an OFF state, with no transcription and no fluorescence detected. Later, the gene turns on and polymerase molecules start being loaded at the promoter at a certain rate. Each polymerase that starts transcribing the gene into mRNA will add a constant amount of fluorescence. Once the first polymerase finishes transcription and leaves the gene, fluorescence will stabilize, as the rates of transcription initiation and termination are the same. Once the gene turns off, no more polymerase molecules are being loaded and the ones already loaded start leaving the gene, decreasing the fluorescence.

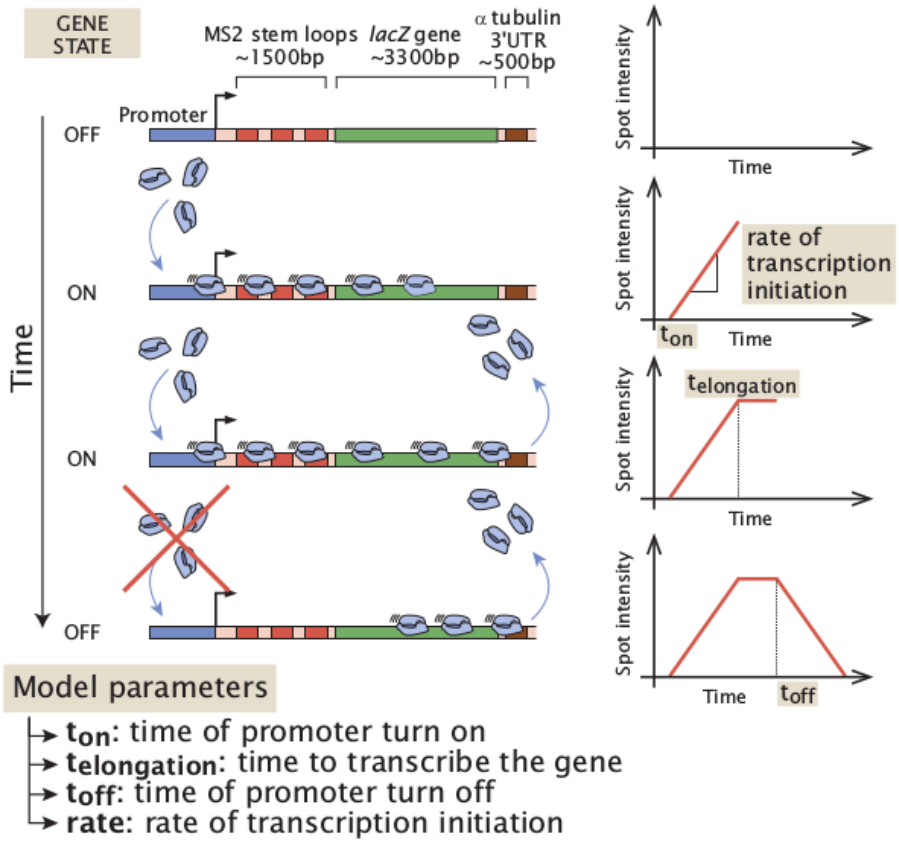


Figure 1.9: Simple model of fluorescence dynamics. First row: The gene is OFF. There are no polymerase molecules and no fluorescence. Second row: The gene turns ON. Polymerase molecules are loaded at the promoter at a constant rate and fluorescence increases linearly as the gene fills up. Third row: The gene is still ON. Polymerase molecules have reached the end of the gene, and they leave at the same rate with which they are being loaded at the promoter. Fluorescence keeps constant. Fourth row: The gene turns OFF. Polymerase molecules already loaded start leaving the gene and fluorescence decreases linearly.

Bursting may result from stochastic transitions between ON and OFF states of the gene (Figure 1.10), and biochemical noise would account for different profiles of transcriptional activity among nuclei. The stochastic switching between different states of the promoter could be caused by the reversible binding and dissociation of transcription factors.

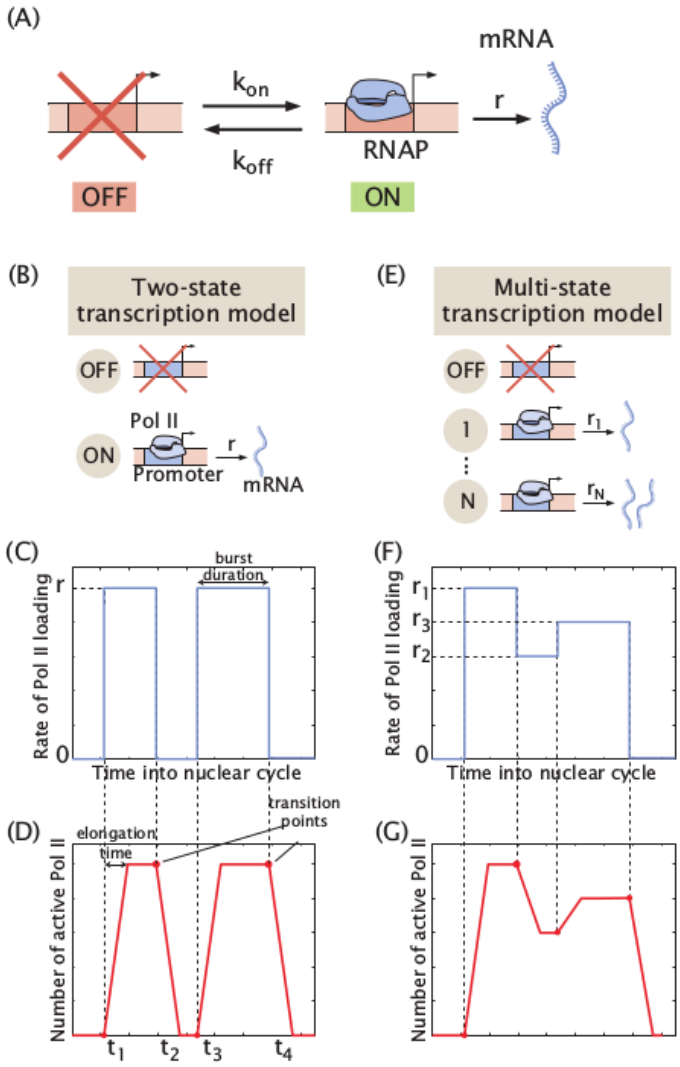


Figure 1.10: Simple model of gene dynamics. (A, B) A gene makes stochastic transitions between an ON and an OFF state. When the gene is ON, it can load polymerase molecules and produce mRNA at a certain rate. (C) Example of a gene ON/OFF state trajectory. (D) Fluorescence dynamics of the trajectory in (C), according to the model described in Figure 1.9. (E, F, G) Gene and fluorescence dynamics corresponding to a more complex model, with multiple discrete states, each with a distinct polymerase loading rate.

Inferring these gene dynamics from the observed fluorescent traces is an interesting statistical problem. Figure 1.11 shows an attempt of fitting a trace by hand, however, given the complexity of the traces and the magnitude of the noise, we need a more reliable method. In the next chapter, we will introduce the statistical framework we will use to tackle this problem.

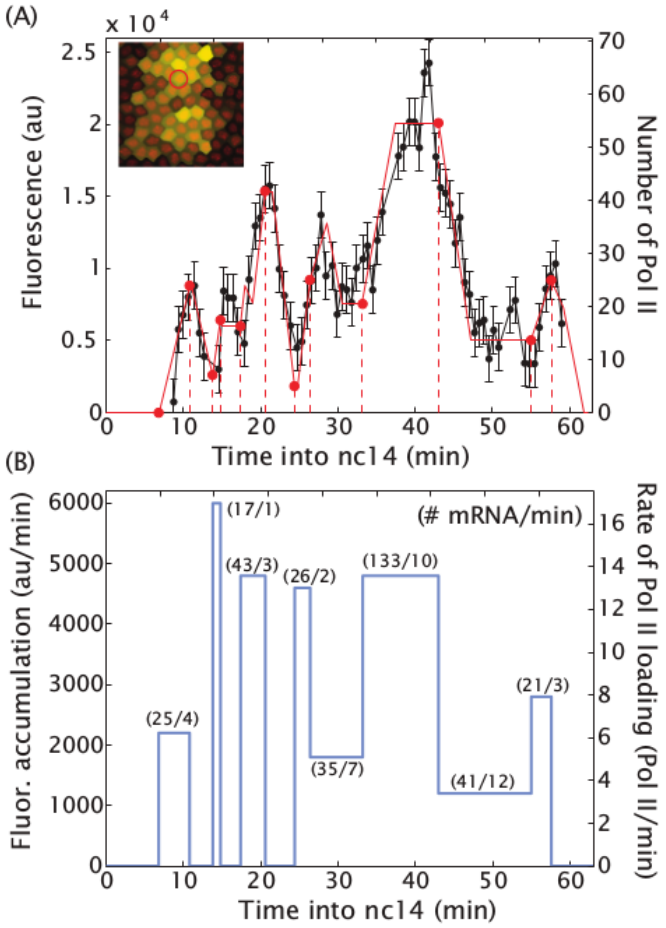


Figure 1.11: Observed fluorescence dynamics and inferred gene dynamics. (A) A fluorescent trace extracted from a nucleus, showing burstiness. (B) A possible gene state trajectory, fit by hand to the observed fluorescence.

Achieving a better understanding of transcriptional dynamics, and how it varies across time and space in the embryo, would cast a light on the underlying molecular mechanisms, as well as on how the information provided by transcription factor concentrations is interpreted by the development system.

Chapter 2

Introduction to Bayesian Inference

A lot of phenomena in our world, where uncertainty is prevalent, are best described by probabilistic models. Statistical inference entails estimating properties of probabilistic processes underlying observed data. The Bayesian approach to inference is to represent our knowledge about unobserved or latent variables with full probability distributions. We aim to present an introduction to Bayesian inference; for a deeper treatment we refer our readers to [GCS+13].

2.1 Simple model: a biased coin

Let's say we pick a coin, possibly *weighted*, such that it has an unknown probability θ of landing heads. We toss it $n = 3$ times and observe $y = 3$ heads. What can we say about θ?

A Bayesian model is based on a joint probability distribution over data y and parameters θ:

$$p(y, \theta) = p(y|\theta)p(\theta) \tag{2.1}$$

The quantity $p(y|\theta)$, known as the *likelihood function*, is the probability that the data y was generated with parameters θ—in this case according to the binomial distribution:

$$p(y|\theta) = Bi(y; n, \theta) = \binom{3}{3}\theta^3(1-\theta)^{3-3} = \binom{3}{3}\theta^3(1-\theta)^{3-3} = \theta^3 \tag{2.2}$$

The remaining factor $p(\theta)$, called the *prior distribution*, expresses our initial beliefs about θ, before seeing any data. For example, we can choose a $Beta(2, 2)$ distribution (see fig. 2.2(a)), encoding our general expectation of coins to be fair. The model is summarized in Figure 2.1.

Figure 2.1: Graphical model notation for the coin example. Nodes represent random variables (white: latent, shaded: observed), and an arrow denotes conditional dependence between them. In this case, the number of heads depends probabilistically on the weighting of the coin.

Note that the direction of the arrow posits how the data is generated. Inference entails the flow of information in the reverse direction: going from the observed leaves to the latent roots of the graph. Bayesian inference is centered around the following formula, using Bayes' rule to calculate the *posterior distribution* $p(\theta|y)$:

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} = \frac{p(y|\theta)p(\theta)}{p(y)} \tag{2.3}$$

$p(y)$, called the *marginal likelihood*, is a normalization factor:

$$p(y) = \int p(y|\theta)p(\theta)d\theta \tag{2.4}$$

For many inference problems, $p(y)$ be taken as a constant, since data y is fixed. However, it plays an important role in the problem of model selection.

We have the following form:

$$\underbrace{p(\theta|y)}_{\text{posterior}} \propto \underbrace{p(y|\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} \tag{2.5}$$

The formula can be seen as a way to update our initial beliefs about θ after observing new data, weighing the likelihood with the prior (fig. 2.2).

The fact that the result of the inference depends on an arbitrary choice of prior distribution is often a critic towards Bayesian statistics. However, it should be acknowledged that all inference rests on assumptions—the choice of likelihood function is one of them. The prior distribution is an explicit and mathematically consistent way of placing assumptions on how parameters are generated—it can be appropriate in many problems, but we should be aware of it, as with any other assumption.

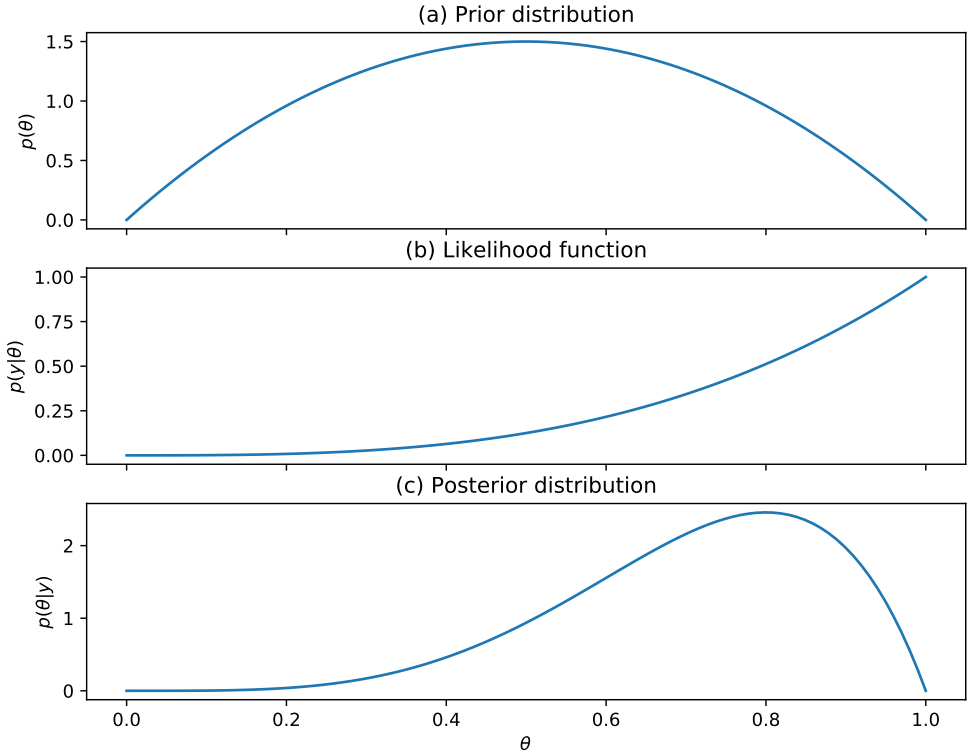


Figure 2.2: Prior distribution, likelihood function and posterior distribution in the coin example

2.2 Sampling methods

Now, let us consider evaluating the posterior expectation of θ:

$$E[\theta|y] = \int \theta \; p(\theta|y) \; d\theta \tag{2.6}$$

or, in general, of a function of θ:

$$E[h(\theta)|y] = \int h(\theta) \; p(\theta|y) \; d\theta \tag{2.7}$$

In most real–world problems, this integral is analytically intractable, but several approximation methods have been developed. The simplest method of approximation consists in computing the value of the integrand over a fixed set of points θ^s, such as grid on the parameter space:

$$E[h(\theta)|y] \approx \sum_{s=1}^{S} h(\theta^s) \; p(\theta^s|y) \tag{2.8}$$

However, this method is not suitable in high dimensions, as the number of evaluations required grows exponentially. This phenomenon is known as the *curse of dimensionality*. An alternative approach is based on obtaining random samples θ^s from the posterior distribution $p(\theta|y)$:

$$E[h(\theta)|y] \approx \frac{1}{S}\sum_{s=1}^{S} h(\theta^s) \tag{2.9}$$

This method was introduced in a 1953 paper by Metropolis et al. [MRR+53], as a statistical mechanics application, with $\exp(-E/kT)$ being the Boltzmann factor:

> "Instead of choosing configurations randomly, then weighting them with $\exp(-E/kT)$, we choose configurations with a probability $\exp(-E/kT)$ and weight them evenly."

2.3 Markov chain Monte Carlo

Monte Carlo algorithms rely on the use of random samples. For many problems, they can result in simpler solutions than those provided by deterministic methods. For example, a Monte Carlo method to estimate the constant π is the following:

- Generate random points in the unit square.
- Count the number of points falling inside a circle inscribed in the square.
- The fraction of points falling inside the circle is then an approximation of $\pi/4$, the ratio of the area of the circle to that of the square.

Markov chain Monte Carlo (MCMC) is a general class of algorithms for sampling from a probability distribution, which has proved useful in high-dimensional settings. MCMC simulates values θ^s forming a Markov chain. That is, θ^s is generated iteratively, based on a transition probability P that, given the last value θ^{s-1}, is independent of the past:

$$P(\theta^s|\theta^1, ..., \theta^{s-1}) = P(\theta^s|\theta^{s-1}) \tag{2.10}$$

The key property is that the stationary distribution of the chain matches the objective distribution π. MCMC algorithms can satisfy this property by ensuring:

- Detailed balance:
$$\pi(x)P(x'|x) = \pi(x')P(x|x') \tag{2.11}$$
- Ergodicity:
Every state must be aperiodic—it should not be visited after a fixed number of steps—and positive recurrent—the expected steps to return to the same state should be finite.

These conditions are not hard to satisfy, and thus MCMC methods have been applied in a wide variety of settings.

The first values in the chain are discarded, in what is known as the *burn–in* period, as they are not representative of the stationary distribution (fig. 2.3 (a)). Even after reaching convergence, values from the chain are correlated; to obtain independent samples the chain must be *thinned*, that is, only every kth value is kept.

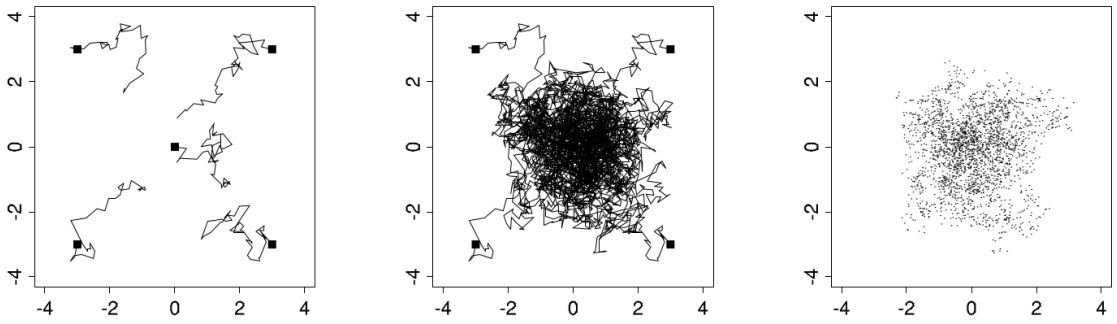


Figure 2.3: Five MCMC chains to sample from the bivariate unit normal distribution (figure from [GCS+13]). Initial values are chosen randomly. (a) The chains are far from convergence. (b) The chains are close to convergence. (c) The values from the second halves of the chains represent (correlated) samples from the target distribution

Assessing convergence

Figure 2.4 illustrates some of the challenges of assessing the convergence of MCMC chains. On the left, each sequence looks stable by itself, however by looking at them together it is clear that they have not converged to a common distribution—they have not *mixed*. On the right, the sequences cover the same distribution—they have mixed—but they lack *stationarity*.

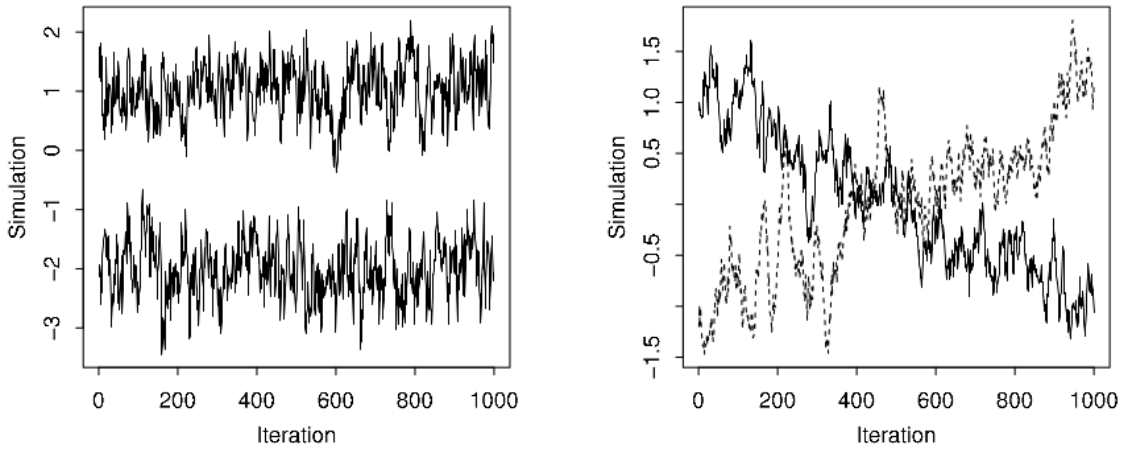


Figure 2.4: Assessing convergence of MCMC chains. (A) On the left, the sequences lack mixing. (B) On the right, the sequences lack stationarity. (Figure from [GCS+13])

The Gelman–Rubin convergence diagnostic [GR92] is based on analyzing variance both between and within sequences. A *potential scale reduction factor* is calculated for each scalar quantity of interest, such as the value of the parameters, which estimates how much the variance might decrease with additional iterations. It approaches 1 as the number of iterations goes towards infinity; values less than 1.1 are considered enough for practical uses, as a rule of thumb.

2.4 Metropolis-Hastings

Metropolis–Hastings is a widely used MCMC algorithm, first proposed by Metropolis et al. in 1953 [MRR+53] and generalized by Hastings in 1970 [Has70]. It is a random walk based on the following:

- The ability to calculate the target density $\pi(\theta)$ at any point θ, up to a multiplicative factor
- A *proposal distribution* $Q(\theta^*|\theta)$ that proposes a transition to a new state θ^* given the current state θ
- The acceptance or rejection of the proposal, with an acceptance probability $\alpha(\theta \to \theta^*)$

$\alpha(\theta \to \theta^*)$ is chosen such that the condition of detailed balance is satisfied:

$$\pi(\theta)P(\theta^*|\theta) = \pi(\theta^*)P(\theta|\theta^*) \tag{2.12}$$

The transition probability $P(\theta^*|\theta)$ is decomposed into making a proposal $Q(\theta^*|\theta)$ and accepting it, with probability $\alpha(\theta \to \theta^*)$:

$$\begin{aligned} \pi(\theta)\ Q(\theta^*|\theta)\ \alpha(\theta \to \theta^*) &= \pi(\theta^*)\ Q(\theta|\theta^*)\ \alpha(\theta^* \to \theta) \Rightarrow \\ \frac{\alpha(\theta \to \theta^*)}{\alpha(\theta^* \to \theta)} &= \frac{\pi(\theta^*)}{\pi(\theta)}\ \frac{Q(\theta|\theta^*)}{Q(\theta^*|\theta)} \end{aligned} \tag{2.13}$$

It is easy to see that taking $\alpha(\theta \to \theta^*) = \min\left(1, \frac{\pi(\theta^*)}{\pi(\theta)}\frac{Q(\theta|\theta^*)}{Q(\theta^*|\theta)}\right)$ satisfies the equation.

Algorithm 1: Metropolis–Hastings

Choose an arbitrary θ^0, such that $\pi(\theta^0) > 0$;
for $i = 1, \ldots$ **do**
  Sample $\theta^* \sim Q(\theta^*|\theta^{i-1})$;
  $\theta^t \leftarrow \begin{cases} \theta^* & \text{with probability } \min\left(1, \frac{\pi(\theta^*)}{\pi(\theta^{i-1})}\frac{Q(\theta^{i-1}|\theta^*)}{Q(\theta^*|\theta^{i-1})}\right) \\ \theta^{i-1} & \text{otherwise} \end{cases}$
end

It is important to remark that, when a proposal is rejected, the chain stays in the same state, and that state counts as a (repeated) sample.

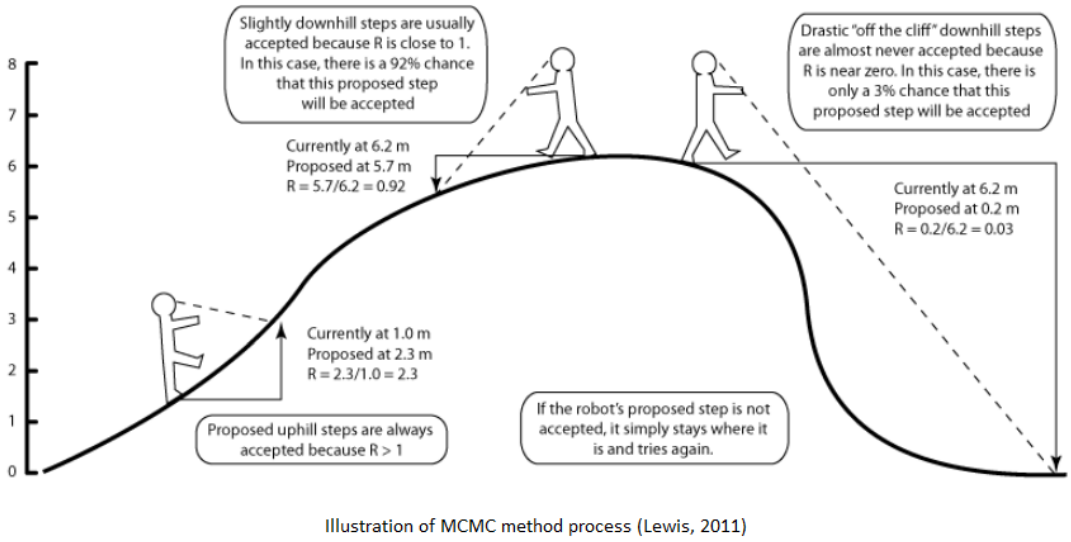


Figure 2.5: Illustration of Metropolis–Hastings as a random walker in a probability density [Rob]. The proposal distribution is assumed symmetrical and the acceptance probability depends solely on the target density ratio R

As seen in Figure 2.5, a jump to a value with a higher probability is always accepted; a jump to a value with a lower probability is sometimes accepted. In this light, it can be seen as an optimization procedure. Also, it is intuitive that the chain will spend more time in more probable states.

Some considerations for choosing a good proposal distribution:

- It should be easy to sample from $Q(\theta^*|\theta)$.
- It should be easy to compute the acceptance probability.
- Each jump should cover enough distance so that the random walk does not move too slowly.
- Transitions should not be rejected too often, or too much time would be wasted standing still.

For real–valued parameters, a common choice for the proposal distribution is a Gaussian centered at the current value, with a relatively small variance. Most proposed jumps would be of a local character, aiming to achieve a high acceptance; but occasionally bigger, riskier jumps would be proposed, which would enable a better exploration of the parameter space.

2.5 Reversible jump MCMC

Reversible jump Markov chain Monte Carlo (RJMCMC) is an extension of Metropolis–Hastings that allows moving between parameter spaces of varying dimensionality, introduced by Green in 1995 [Gre95]. The paper takes the case of a time series of coal mining accidents in the UK, introduced by Raftery & Akman [RA86]. The data can be seen in Figure 2.6.

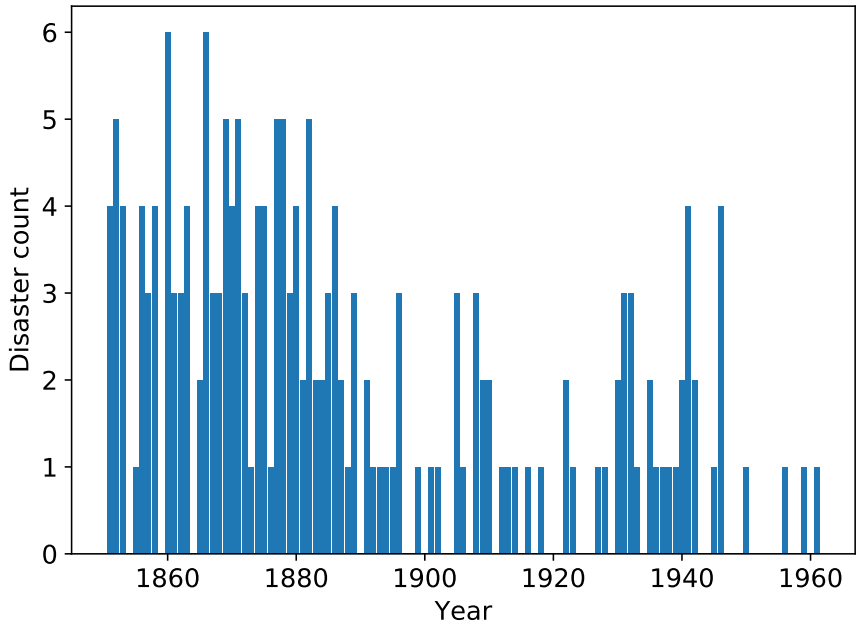


Figure 2.6: Coal mining disasters in the UK, 1851–1962.

The basic model assumes that the number of disasters in a given year has a Poisson distribution with a certain rate, and the goal of the analysis is to understand if this underlying rate may have changed through the years, for example due to safety regulations. Raftery & Akman explore the case where the rate function has exactly one changepoint. Green analyzes the more complex case where the rate function can be any step function, with an unknown number of changepoints.

To perform inference in this multiple changepoint model, Green introduces RJMCMC, a generalization of MCMC to parameter spaces of varying dimensionality. The general parameter space can be thought as the union of several parameter subspaces θ_k, each under a model k with dimensionality d_k. In the multiple changepoints example, the space is the union of models with one changepoint $(1, (t_1))$, models with two changepoints $(2, (t_1, t_2))$, etc. Green introduces transdimensional jumps: the addition of a new changepoint at a random location ("birth" proposal) and the random removal of a changepoint ("death" proposal).

To propose a jump from a state (k, θ_k) to a state (k^*, θ_{k^*}), and ensure detailed balance even if the dimensions are different, a *dimension–matching* step is introduced. θ_k and θ_{k^*} are padded with auxiliary variables u and u^*, respectively, such that $d_k + \dim(u) = d_{k^*} + \dim(u*)$; and a deterministic, bijective function $g_{k,k^*}(\theta_k, u) = (\theta_{k^*}, u^*)$ is defined. Note that either of $\dim(u)$ or $\dim(u^*)$ can be zero.

Let's illustrate the ("rather obscure", in the words of the author) dimension–matching condition with a simple example. We are modeling a mixture of unit–variance gaussians $\mathcal{N}(\mu, 1)$, such that the number of components is not fixed. A model with n components would have parameters $\mu_1, \cdots, \mu_n$. Let's consider a jump from state $(1, (\mu))$ to state $(2, (\mu_1, \mu_2))$. In some context, a reasonable move could be to "split" the component into two components centered around the previous one, with some separation. The idea would be to sample u from some distribution, and set $\mu_1 = \mu - u/2,\ \mu_2 = \mu + u/2$:

$$g_{1,2}(\mu, u) = (\mu - u/2,\ \mu + u/2) \tag{2.14}$$

The reverse move would be to "merge" the two components:

$$g_{2,1}(\mu_1, \mu_2) = ((\mu_1 + \mu_2)/2,\ \mu_2 - \mu_1) \tag{2.15}$$

Each RJMCMC iteration would be as follows:

Algorithm 2: RJMCMC step

Given current state (k, θ_k), propose a move to model k^* with probability J_{k,k^*};
Generate auxiliary random variable $u \sim J(u|k, k^*, \theta_k)$;
Determine the proposed new parameters, $(\theta_{k^*}, u^*) = g_{k,k^*}(\theta_k, u)$;
Calculate the ratio $r = \frac{\pi(k^*,\theta_{k^*})}{\pi(k,\theta_k)} \frac{J_{k^*,k} J(u^*|k^*,k,\theta_{k^*})}{J_{k,k^*} J(u|k,k^*,\theta_k)} \left| \frac{\partial(\theta_{k^*},u^*)}{\partial(\theta_k,u)} \right| =$
(target density ratio) $\times$ (proposal ratio) $\times$ (Jacobian);
Accept the move with probability $\min(1, r)$;

The last term in the acceptance ratio is the Jacobian of the transformation g_{k,k^*}.

In the previous example, the Jacobian factor of the split move is:

$$\left|\frac{\partial(\mu-u/2,\ \mu+u/2)}{\partial(\mu,u)}\right| = \begin{vmatrix} 1 & -1/2 \\ 1 & 1/2 \end{vmatrix} = 1 \tag{2.16}$$

RJMCMC will allow us to analyze fluorescent traces from individual nuclei in the developing fly, and infer the underlying trajectory of promoter states, with its unknown number of changepoints.

Chapter 3

Probabilistic model of transcriptional dynamics

The goal of the present work is to analyze fluorescent traces of transcriptional activity from the early fruit fly embryo, and extract the underlying transcriptional dynamics. In this chapter, we present a hierarchical Bayesian model describing the probabilistic relationship between the observed fluorescence and the changes in polymerase loading rate at the promoter.

3.1 Transcriptional bursting

It is assumed that the gene promoter switches stochastically between S discrete states (we will use $S = 2, 3, 4$), each with a characteristic polymerase loading rate r_i. Each state could reflect a different occupancy of transcription factor binding sites. We assume transcriptional bursting takes place in a transcription time window $[t_{start}, t_{end}]$. The higher–order process that determines the length of the transcription time window is not currently modeled.

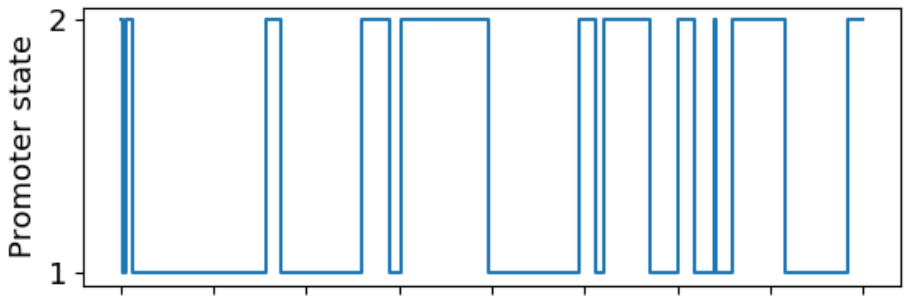
p shall denote the trajectory of the promoter state, with $p(t)$ the state at time t, taking values in $\{1, \ldots, S\}$. Between t_{start} and t_{end}, p is modeled as a continuous–time Markov chain. A continuous–time Markov chain can be seen as the limit of a discrete–time Markov chain as the time step becomes smaller. The chain is characterized by an initial probability distribution π and a transition rate matrix k, with $k_{i,j}$ $(i \neq j)$ the rate, in events per minute, of a transition from state i to state j.

An intuitive way to simulate a continuous–time Markov chain is the following:

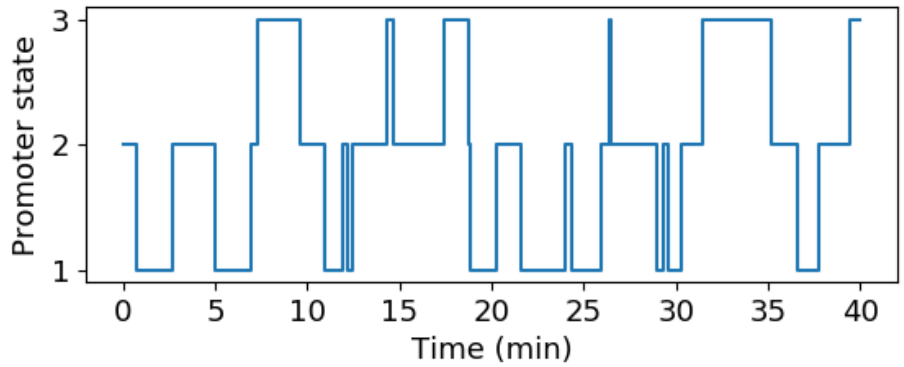
Algorithm 3: Generate a continuous–time Markov chain given t_{start} , t_{end}, π and k

$T \leftarrow t_{start}$; `// initial time`
$X \leftarrow$ sample from π; `// initial state`
while $T < t_{end}$ **do**
  For each non–zero rate $k_{X,Y}$, generate an exponential t_Y with rate $k_{X,Y}$;
  Pick the smallest, t_Z;
  $T \leftarrow T + t_Z$; `// update time`
  $X \leftarrow Z$; `// jump to the new state`
end

We introduce the additional constraint that the promoter state can only jump to adjacent states, that is, $|i - j| > 1 \Rightarrow k_{i,j} = 0$. This is known as a birth–death process. These assumptions should be revisited, but for the moment the reduced degrees of freedom increase the performance of the algorithm as well as the interpretability of the results. Two simulated promoter trajectories are depicted in Figure 3.1, corresponding to different transition rate matrices.



(a) $S = 2,\ k_{1,2} = k_{2,1} = 0.5$



(b) $S = 3,\ k_{1,2} = 1,\ k_{2,1} = k_{2,3} = 0.5,\ k_{3,2} = 1$

Figure 3.1: Promoter trajectories simulated according to different transition rate matrices.

Probabilities

We define minimum and maximum transition rates, $k_{min} = 0.01$ events/m and $k_{max} = 10$ events/m. These are an estimation, and future work should better accommodate biological and physical constraints on the rates. Transition rate matrix elements are given a uniform prior distribution:

$$\begin{aligned} k_{i,j} &\sim \mathcal{U}(k_{min}, k_{max}) \\ P(k) = \prod_{|i-j|=1} P(k_{i,j}) &= \prod_{|i-j|=1} \frac{1}{k_{max} - k_{min}} \end{aligned} \tag{3.1}$$

Similarly, we define constraints for the polymerase loading rate, $r_{min} = 0$ pol/m, $r_{max} = 50$ pol/m. Polymerase loading rates for each state are given a uniform prior distribution:

$$\begin{aligned} r_i &\sim \mathcal{U}(r_{min}, r_{max}) \\ P(r) = \prod_i P(r_i) &= \prod_i \frac{1}{r_{max} - r_{min}} \end{aligned} \tag{3.2}$$

To calculate the likelihood of a continuous–time Markov chain realization, it is enough to know the time spent in each state (T_i) and the number of transitions between each pair of states ($Q_{i,j}$)—these are the *sufficient statistics* [GM95]. The likelihood of a certain promoter trajectory, given a transition rate matrix, can be calculated in the following way:

$$P(p|k) = \prod_{|i-j|=1} k_{i,j}^{Q_{i,j}} e^{-k_{i,j} T_i} \tag{3.3}$$

3.2 Fluorescence

By f_{pol} we shall denote the fluorescence of a nascent mRNA molecule with 24 MS2 loops attached to GFP. f_{pol} has been estimated at 20 a.u. of the microscope, albeit with 20-30% error bounds. In this work we will freely use that value to have some intuition, in the sense that 100 a.u. of fluorescence could correspond to 5 mRNA molecules. Since we currently analyze relative counts of mRNA, rather than absolute counts, we are not affected by the error in the estimation.

The elongation time e_t of the *eve* gene (including the MS2 sequence) has previously been estimated as 3 minutes [GTLG13]. The time spent transcribing the MS2 sequence at the start of the gene (l_t) can be estimated as 0.6 minutes, taking into account the ratio of the length of the MS2 sequence to the length the whole gene. The fluorescent profile of a single mRNA as it is transcribed is illustrated in Figure 3.2. As each of the 24 MS2 loops are being transcribed, fluorescence increases linearly while GFP binds to each of them. After the elongation time, the mRNA molecule leaves the gene and fades into the background.

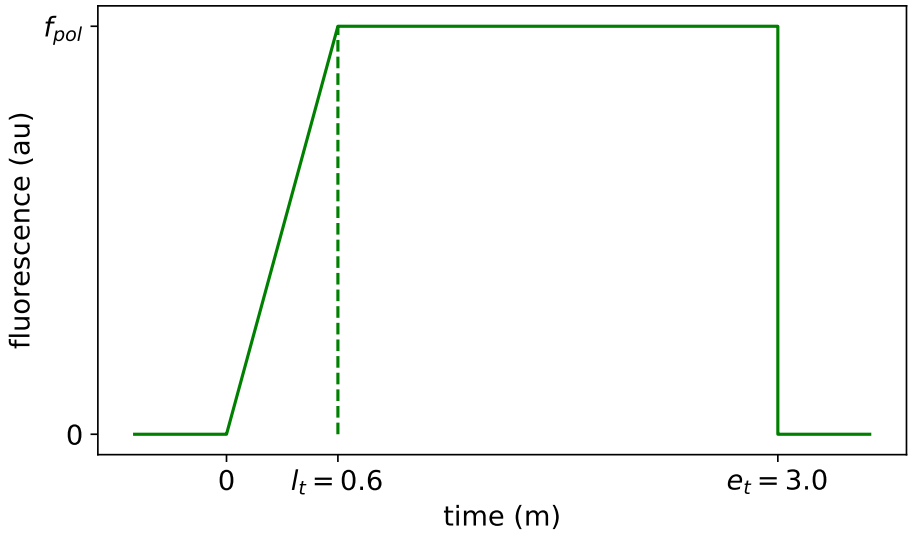


Figure 3.2: Fluorescent profile of a single mRNA as it is transcribed.

The fluorescence observed at the spot at time t, $f(t)$, is the sum of the fluorescence of mRNA molecules currently being transcribed. Thus, the signal $f(t)$ will have memory of the mRNAs that have started being transcribed up to three minutes before $(t - e_t)$, as they will still be at the gene. The count of mRNAs at the gene is equal to the integral of the polymerase loading rate over the last three minutes. The mRNAs that have started being transcribed in the last $l_t = 0.6$ minutes do not have full fluorescence as the MS2 loops are still being transcribed. Given the history of polymerase loading rates $r(t)$, $f(t)$ is calculated in the following way:

$$f(t) = \int_{t-e_t}^{t} r(t')\ f_{pol}\ \min\left(1, \frac{t-t'}{l_t}\right) dt' \tag{3.4}$$

It is, on general terms, the integral of the polymerase loading rate in the past 3 minutes (e_t), times the fluorescence associated with one polymerase. The factor $\min\left(1, \frac{t-t'}{l_t}\right)$ is added to contemplate recent mRNAs that do not yet have full fluorescence (the linear increase section in Figure 3.2).

Figure 3.3 illustrates the dynamics of the fluorescence given a simple trajectory of the polymerase loading rate, while Figure 3.4 shows a more complex case.

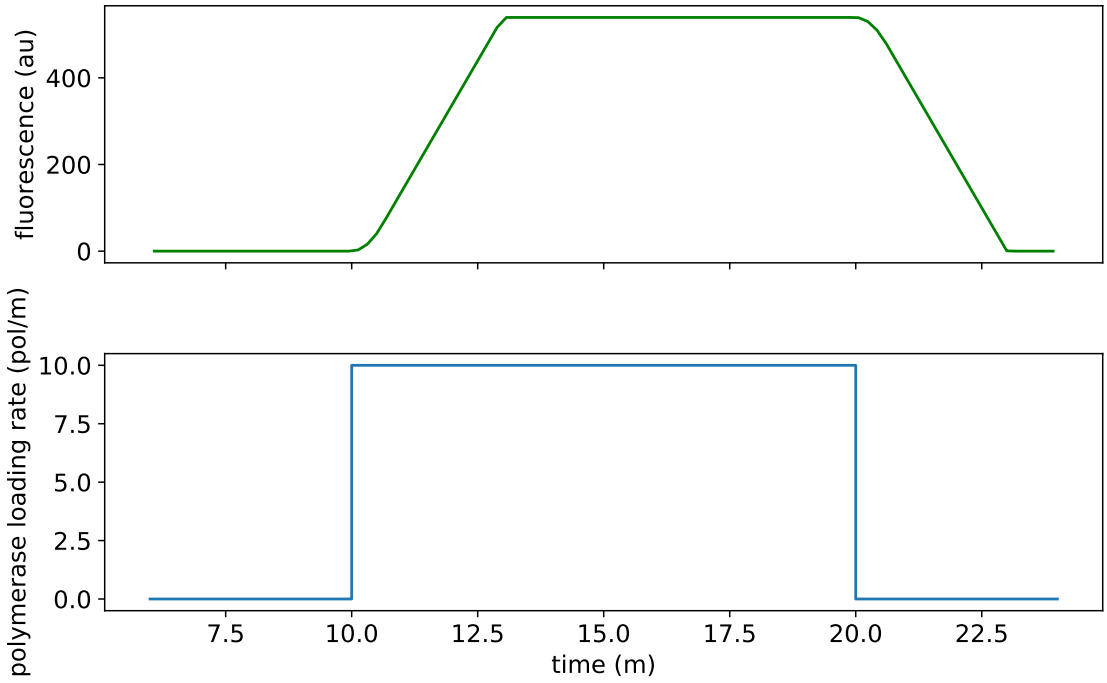


Figure 3.3: Fluorescence observed given a simple trajectory of the polymerase loading rate. As polymerases start being loaded at the promoter at time 10 min., fluorescence starts increasing linearly as the gene is being progressively filled with polymerase molecules. After the elongation time of three minutes, polymerase molecules start being released from the gene. Up to time 20 min., the rate at which polymerases enter and leave the gene is equal, so the fluorescence remains constant. After time 20 min., no polymerases are being loaded anymore and fluorescence starts decreasing to zero, as polymerase molecules already on the gene start being released.

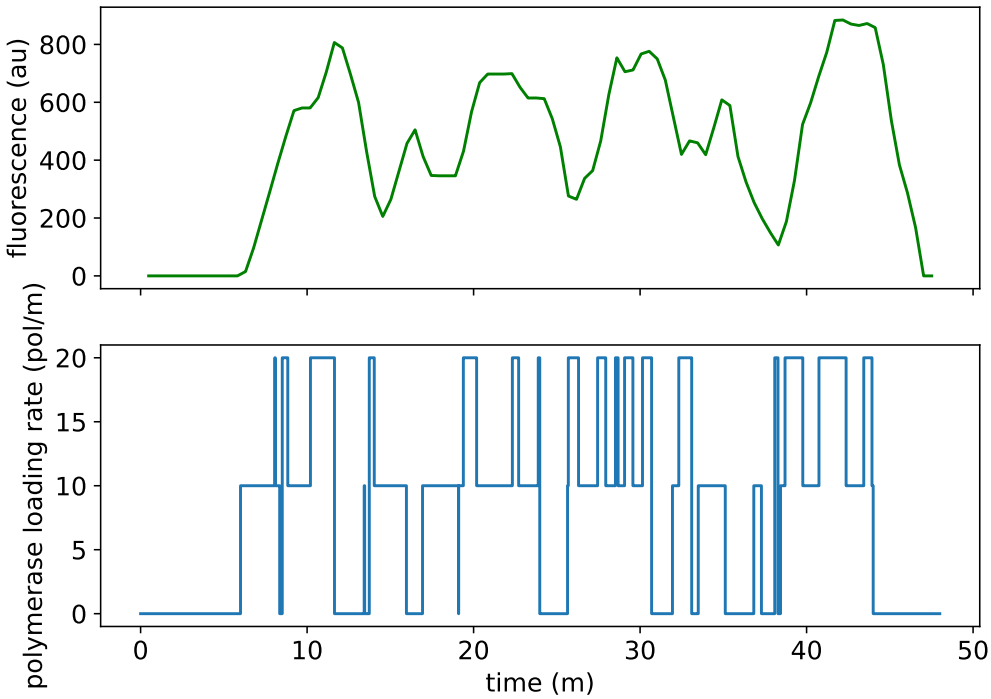


Figure 3.4: Fluorescence observed given a complex trajectory of the polymerase loading rate.

Probability

The experimental data consists of n measurements of the fluorescence at specific times: $(t_i, f_i),\ i = 1, \ldots, n$. Equation 3.4 shows how to calculate the fluorescence expected given the promoter state trajectory p and the loading rates for each state r, $f(t_i, p, r)$. The observational error, $f_i - f(t_i, p, r)$, is assumed to be gaussian with zero mean and unknown variance σ^2:

$$f_i \sim \mathcal{N}(f(t_i, p, r), \sigma^2) \tag{3.5}$$

σ is assigned a uniform prior distribution between $\sigma_{min} = 10$ a.u. and $\sigma_{max} = 400$ a.u.:

$$\begin{aligned} \sigma &\sim \mathcal{U}(\sigma_{min}, \sigma_{max}) \\ P(\sigma) &= \frac{1}{\sigma_{max} - \sigma_{min}} \end{aligned} \tag{3.6}$$

As noted earlier, this error is predominantly determined by the error in the estimation of the background fluorescence. A more detailed model of the error could be developed in the future.

The probability of the observed fluorescent trace, given the promoter state tra-

jectory and the polymerase loading rates for each state, is thus:

$$P(f|p,r,\sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(f_i - f(t_i,p,r))^2\right] \tag{3.7}$$

3.3 Complete model

So far, the joint distribution over the random variables in our model factorizes as follows (see Figure 3.5 for a graphical representation):

$$P(k,r,\sigma,p,f) = \underbrace{P(k)}_{\text{Prior on } k} \underbrace{P(r)}_{\text{Prior on } r} \underbrace{P(\sigma)}_{\text{Prior on } \sigma} \underbrace{P(p|k)}_{\substack{\text{Likelihood of } p \\ \text{given } k}} \underbrace{P(f|p,r,\sigma)}_{\substack{\text{Likelihood of } f \\ \text{given } p,\ r \text{ and } \sigma}} \tag{3.8}$$

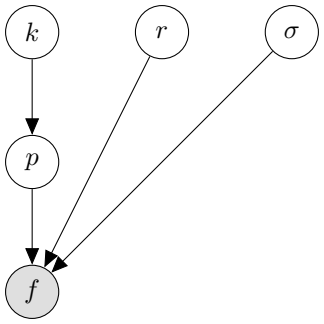


Figure 3.5: Graphical representation of the model

Parameter sharing

An additional assumption is that transcriptional processes of nuclei at the same position in the anterior–posterior (AP) axis share the same parameters k and r. That is, each nucleus goes through a different promoter state trajectory and thus has a different fluorescent profile (as seen in Figure 1.8), but that these are stochastic processes governed by the same parameters k, r (σ is also assumed to be shared). The reason we group together nuclei at the same AP position is that they are exposed to the same concentration of transcription factors, which we assume are the underlying parameters of the system.

The model is modified by grouping a number M of different nuclei such that

they share k, r and σ (see the graphical representation in Figure 3.6):

$$
\begin{aligned}
P(k, r, \sigma, p, f) &= P(k)P(r)P(\sigma)\prod_{i=1}^{M} P(p_i|k)P(f_i|p_i, r, \sigma) \\
P(k) &= \prod_{|i-j|=1} P(k_{i,j}) = \prod_{|i-j|=1} \frac{1}{k_{max} - k_{min}} \\
P(r) &= \prod_{i} P(r_i) = \prod_{i} \frac{1}{r_{max} - r_{min}} \\
P(\sigma) &= \frac{1}{\sigma_{max} - \sigma_{min}} \\
P(p|k) &= \prod_{|i-j|=1} k_{i,j}^{Q_{i,j}} e^{-k_{i,j}T_i} \\
P(f|p, r, \sigma) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(f_i - f(t_i, p, r))^2\right]
\end{aligned}
\tag{3.9}
$$

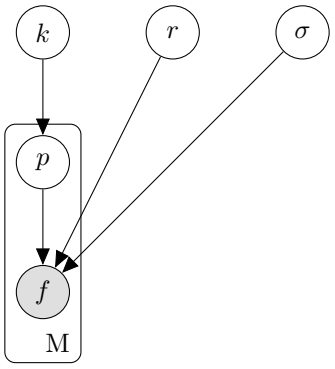


Figure 3.6: Graphical representation of the model. The plate surrounding variables p and f indicates that these variables are repeated M times.

The assumption we have introduced increases the statistical power and the information available to infer the underlying rates k and r.

With the model in hand, we need an inference algorithm to move from the observed fluorescence to the latent transcriptional dynamics. Software packages such as Stan [CGH+17] have excellent implementations of inference algorithms supporting a large class of models, however, support is scarce for dimension–varying parameters, such as the promoter trajectory. Hence, we set to develop a new inference algorithm.

Chapter 4

Inference algorithm

We have developed a Reversible–jump Markov chain Monte Carlo (RJMCMC) algorithm, that allows us to sample from the posterior distribution $P(k, r, \sigma, p|f)$, the distribution of our parameters of interest conditioned on the data f. That is, it generates N samples $(k^i, r^i, \sigma^i, p^i)$, that can be used to approximate quantities of interest, such as:

$$\begin{aligned} E[k|f] &\approx \overline{k^i} = \frac{1}{N}\sum_{i=1}^{N} k^i \\ Var[k|f] &\approx \frac{1}{N-1}\sum_{i=1}^{N}(k^i - \overline{k^i})^2 \end{aligned} \tag{4.1}$$

The algorithm can be outlined in the following way:

Algorithm 4: Sampler

Start from an initial state;
for *i = 1, ...* **do**
  Choose a move from $\{r, \sigma, k, t, \text{Birth}, \text{Death}\}$;
  Propose a new state with the move;
  Compute the acceptance probability;
  Accept or reject the proposal;
end

In general, for a move from a state (k, r, σ, p) to a state (k', r', σ', p'), the ratio used in the acceptance probability is:

$$\begin{aligned} &\text{(likelihood ratio)} \times \text{(prior ratio)} \times \text{(proposal ratio)} \times \text{(Jacobian)} = \\ &\frac{\prod_{i=1}^{M} P(f_i|p_i', r', \sigma')}{\prod_{i=1}^{M} P(f_i|p_i, r, \sigma)} \times \frac{P(k')P(r')P(\sigma')\prod_{i=1}^{M} P(p_i'|k')}{P(k)P(r)P(\sigma)\prod_{i=1}^{M} P(p_i|k)} \times \text{(proposal ratio)} \times \text{(Jacobian)} \end{aligned} \tag{4.2}$$

It should be noted that for any given move only some of the parameters are updated, so most of these factors do not need to be calculated again. Also, the Jacobian factor is only relevant in the Birth and Death moves, which are trans–dimensional.

Representation of the promoter trajectory

The promoter trajectory is represented as a tuple (cps, hs), with cps an ordered vector of changepoint times and hs a vector of promoter states, one for each segment.

Truncated normal distribution

We shall make use of the truncated normal distribution, $\mathcal{N}_a^b(\mu, \sigma)$, which has the same shape as normal distribution $\mathcal{N}(\mu, \sigma)$, but has support $[a, b]$ and a rescaling factor (see Figure 4.1 for an example).

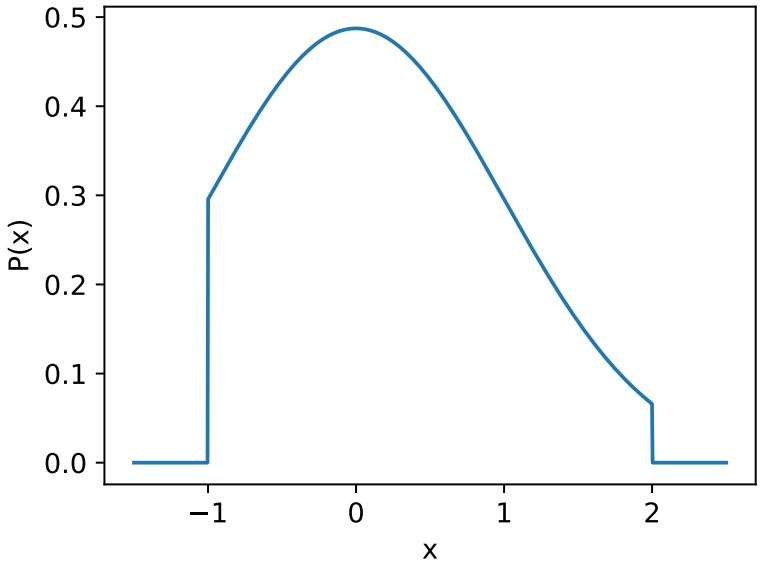


Figure 4.1: A standard normal truncated at -1 and 2.

4.1 r move

One of the polymerase loading rates r_i is updated (see Figure 4.2 for an example). The new r_i' is sampled from a truncated normal distribution, centered at the current value, with a fixed standard deviation $\sigma_r = 1.25$ pol/min. The boundaries are either r_{i-1}, r_{i+1}, r_{min} or r_{max}.

Algorithm 5: r move

$i \sim \mathcal{U}(\{1, \ldots, S\})$;
$a \leftarrow$ if $i > 1$ then r_{i-1} else r_{min};
$b \leftarrow$ if $i < S$ then r_{i+1} else r_{max};
$r_i' \sim \mathcal{N}_a^b(r_i, \sigma_r)$;

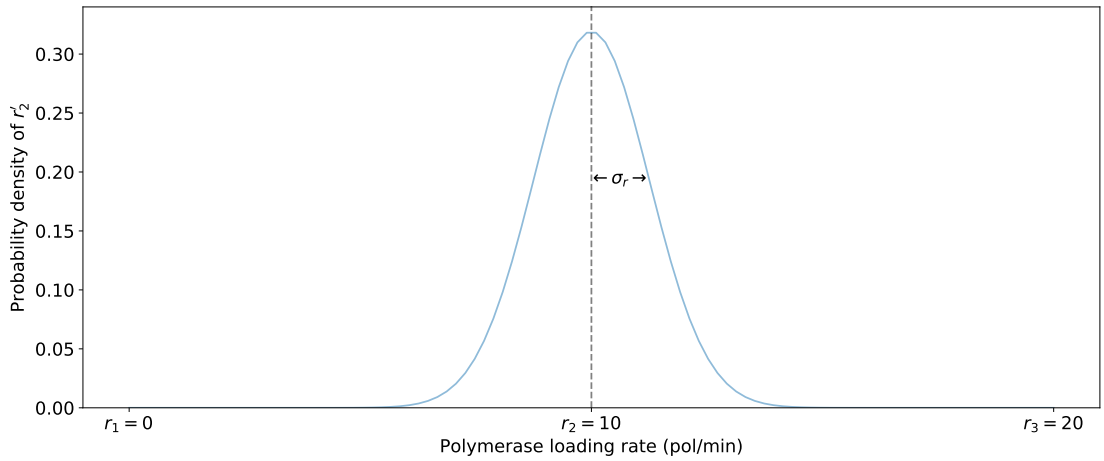


Figure 4.2: A move is proposed for r_2. r_2' is sampled from a normal distribution centered at r_2, with standard deviation σ_r, truncated at r_1 and r_3.

The choice of σ_r is important. If it is too small, exploration of the space will be slow. If it is too large, the move will have a high rejection rate.

The probability of the proposal is:

$$Q(k', r', \sigma', p' | k, r, \sigma, p) = \underbrace{P(r \text{ move})}_{\substack{\text{choosing the} \\ r \text{ move}}} \underbrace{\frac{1}{S}}_{\substack{\text{choosing one rate} \\ \text{to update}}} \underbrace{\mathcal{N}_a^b(r_i'; r_i, \sigma_r)}_{\substack{\text{proposing a} \\ \text{new rate}}} \tag{4.3}$$

To obtain the acceptance probability, the likelihood of each fluorescent trace must be recalculated, given the change in r:

$$\alpha(k, r, \sigma, p \to k', r', \sigma', p') = \min\left(1, \prod_{i=1}^{M} \left(\frac{P(f_i | p_i, r', \sigma)}{P(f_i | p_i, r, \sigma)}\right) \frac{\mathcal{N}_a^b(r_i; r_i', \sigma_r)}{\mathcal{N}_a^b(r_i'; r_i, \sigma_r)}\right) \tag{4.4}$$

4.2 σ move

σ, the estimated error in the fluorescence measurements, is updated. The new value is sampled from a truncated normal distribution centered at the old value, with standard deviation 5 a.u. The limits are σ_{min} and σ_{max}.

Algorithm 6: σ move

$\sigma' \sim \mathcal{N}_{\sigma_{min}}^{\sigma_{max}}(\sigma, 5)$;

The probability of the proposal is:

$$Q(k', r', \sigma', p' | k, r, \sigma, p) = P(\sigma \text{ move})\, \mathcal{N}_{\sigma_{min}}^{\sigma_{max}}(\sigma'; \sigma, 5) \tag{4.5}$$

To obtain the acceptance probability, the likelihood of each fluorescent trace must be recalculated, given the change in σ:

$$\alpha(k, r, \sigma, p \to k', r', \sigma', p') = \min\left(1, \prod_{i=1}^{M} \left(\frac{P(f_i | p_i, r, \sigma')}{P(f_i | p_i, r, \sigma)}\right) \frac{\mathcal{N}_{\sigma_{min}}^{\sigma_{\max}}(\sigma; \sigma', 5)}{\mathcal{N}_{\sigma_{min}}^{\sigma_{\max}}(\sigma'; \sigma, 5)}\right) \tag{4.6}$$

4.3 k move

A transition rate is chosen at random and updated, using a normal distribution centered at the current value, with standard deviation $\sigma_k = 0.05$ events/min, truncated at k_{min} and k_{max}.

Algorithm 7: k move

$i \sim \mathcal{U}(\{1, \ldots, S\})$;
$options_j = \{1, \ldots, S\} \cap \{i-1, i+1\}$;
$j \sim \mathcal{U}(options_j)$;
$k'_{i,j} \sim \mathcal{N}_{k_{min}}^{k_{max}}(k_{i,j}, \sigma_k)$;

The probability of the proposal is:

$$Q(k', r', \sigma', p' | k, r, \sigma, p) = P(k \text{ move}) \, \frac{1}{S} \, \frac{1}{options_j} \, \mathcal{N}_{k_{min}}^{k_{max}}(k'_{i,j}; k_{i,j}, \sigma_k) \tag{4.7}$$

To obtain the acceptance probability, the prior of each promoter trajectory must be recalculated, given the change in k:

$$\alpha(k, r, \sigma, p \to k', r', \sigma', p') = \min\left(1, \prod_{z=1}^{M} \left(\frac{P(p_z|k')}{P(p_z|k)}\right) \frac{\mathcal{N}_{k_{min}}^{k_{max}}(k_{i,j}; k'_{i,j}, \sigma_k)}{\mathcal{N}_{k_{min}}^{k_{max}}(k'_{i,j}; k_{i,j}, \sigma_k)}\right) \tag{4.8}$$

4.4 t move

One nucleus is selected at random. Then, the position of one changepoint in the promoter trajectory is updated, using a truncated normal distribution centered at the current value and with standard deviation $\sigma_t = 0.6$ min. The boundaries are the previous and next changepoint positions.

Algorithm 8: t move

$m \sim \mathcal{U}(\{1, \ldots, M\})$;
$cps, hs \leftarrow p_m$;
$i \sim \mathcal{U}(\{1, \ldots, |cps|\})$;
$a,\ t,\ b \leftarrow cps_{i-1},\ cps_i,\ cps_{i+1}$;
$t' \sim \mathcal{N}_a^b(t, \sigma_t)$;

The probability of the proposal is:

$$Q(k', r', p' | k, r, p) = P(t \text{ move}) \, \frac{1}{M} \, \frac{1}{|cps|} \, \mathcal{N}_a^b(t'; t, \sigma_t) \tag{4.9}$$

To obtain the acceptance probability is, the prior and likelihood of only one nucleus must be calculated, given the change in one of its changepoints:

$$\alpha(k, r, \sigma, p \to k', r', \sigma', p') = \min\left(1, \frac{P(f_i|p'_i, r, \sigma)}{P(f_i|p_i, r, \sigma)} \, \frac{P(p'_i|k)}{P(p_i|k)} \, \frac{\mathcal{N}_a^b(t; t', \sigma_t)}{\mathcal{N}_a^b(t'; t, \sigma_t)}\right) \tag{4.10}$$

4.5 Birth move

The Birth move, together with its reverse, the Death move, are the critical transdimensional jumps in which we add or remove changepoints from the promoter trajectory. They were designed to do small local alterations to the promoter trajectory, in order to increase the chance of their acceptance.

The Birth move is illustrated in Figure 4.3. We replace a step in the middle of the trajectory, going from h_a to h_b, by three steps h_1, h_2 and h_3, adding two changepoints t_1 and t_2. It can be wondered, why add two changepoints and not one, which is simpler? The reason is that if we only added one changepoint we could not in general get from h_a to h_b satisfying the constraint that the value of consecutive steps should differ by one.

However, the first and last steps of the trajectory are only constrained from one side (for example, if we were modifying the first step there would be no constraint from the left). In this cases, we use a very similar move that adds only one changepoint. Here we will describe the more frequent case of choosing a step in the middle of the trajectory, and adding two changepoints.

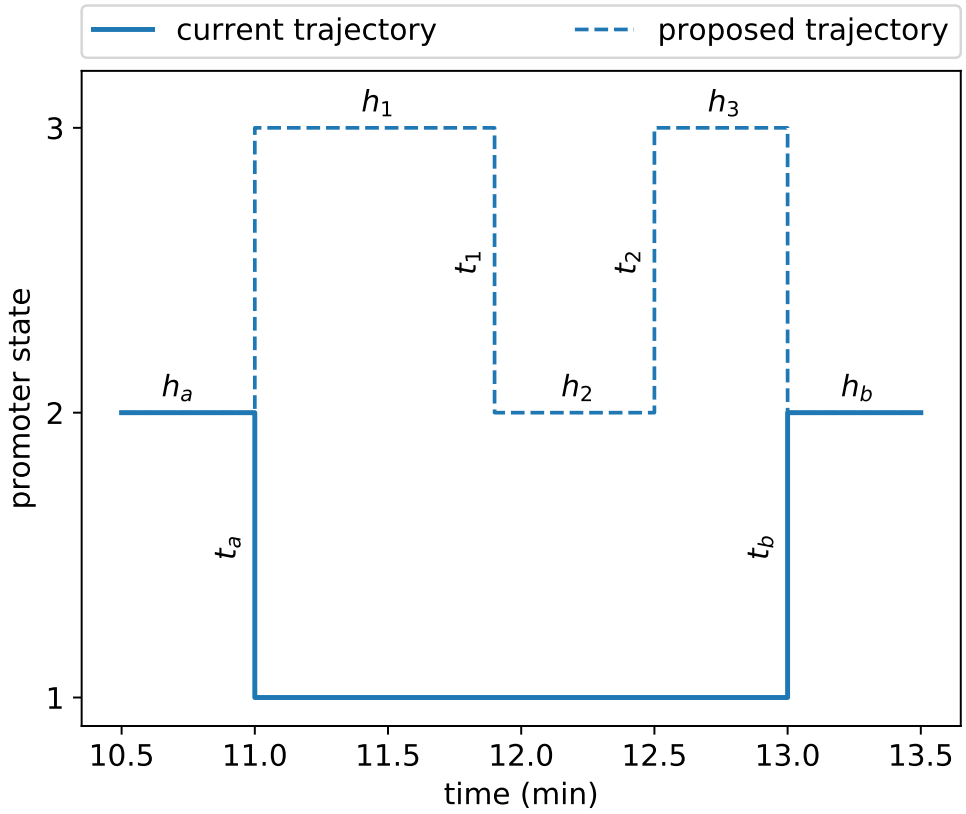


Figure 4.3: Illustration of the Birth move. One promoter trajectory is selected at random and two changepoints are added at some position. Instead of having one step between h_a and h_b, there will be three new steps h_1, h_2, h_3, sepparated by two new changepoints t_1 and t_2.

Algorithm 9: Birth move

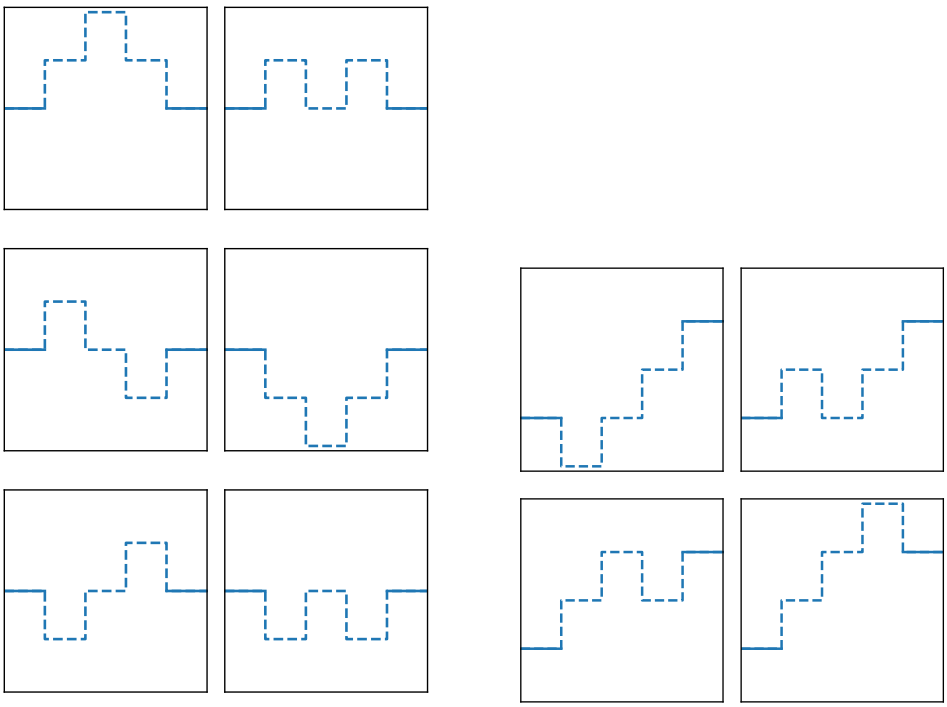
// choose a promoter trajectory to update:
$m \sim \mathcal{U}(\{1, \ldots, M\})$;
$cps, hs \leftarrow p_m$;
// choose a position to insert the new changepoints:
$i \sim \mathcal{U}(\{2, \ldots, |cps| - 1\})$;
$t_a,\ t_b \leftarrow cps_{i-1},\ cps_i$;
// sample two uniform changepoints, in order:
$t_1,\ t_2 \sim \text{sort}([\mathcal{U}(t_a, t_b), \mathcal{U}(t_a, t_b)])$;
$h_a,\ h_b \leftarrow hs_{i-2},\ hs_i$;
// choose the three new values from all possible paths from h_a to h_b:
$(h_1, h_2, h_3) \sim \mathcal{U}(Paths_{Birth}(h_a, h_b))$;
// insert the two new changepoints:
$cps' \leftarrow \left[cps_0, \ldots, cps_{i-1}, t_1, t_2, cps_i, \ldots, cps_{|cps|}\right]$;
// insert the three new values:
$hs' \leftarrow \left[hs_1, \ldots, hs_{i-2}, h_1, h_2, h_3, hs_{i-1}, \ldots, hs_{|hs|}\right]$;

$Paths_{Birth}(h_a, h_b)$ is the set of all paths of three steps from h_a to h_b, satisfying two conditions:

- Values should be between 1 and S.
- Consecutive values should differ by 1.

Since in the original trajectory there is one step between h_a y h_b, there are three cases:

- $h_a = h_b$: There are potentially 6 paths (Figure 4.4a). Some of the paths may be filtered out if values go out of the range $[1, S]$.
- $h_b = h_a + 2$: There are potentially 4 paths (Figure 4.4b). Some of the paths may be filtered out if values go out of the range $[1, S]$.
- $h_b = h_a - 2$: Symmetrical to the previous case.



(a) $h_a = h_b$ (b) $h_b = h_a + 2$

Figure 4.4: Illustration of the potential paths between h_a and h_b using three steps. (a) When $h_a = h_b$, there are 6 possibilities. (b) When $h_b = h_a + 2$, there are 4 possibilities.

The probability of the proposal is:

$$Q(k', r', \sigma', p' | k, r, \sigma, p) = \underbrace{P(\text{Birth move})}_{\substack{\text{choosing} \\ \text{the move}}} \underbrace{\frac{1}{M}}_{\substack{\text{choosing one} \\ \text{trajectory}}} \underbrace{\frac{1}{|cps| - 2}}_{\substack{\text{choosing a} \\ \text{(central) step}}} \underbrace{\frac{2}{(t_b - t_a)^2}}_{\substack{\text{choosing two} \\ \text{sorted points}}} \underbrace{\frac{1}{|Paths_{Birth}(h_a, h_b)|}}_{\text{choosing a path}} \tag{4.11}$$

4.6 Death move

The Death move is the reverse of the Birth move: two changepoints are removed from a promoter trajectory (see Figure 4.5 for an example).

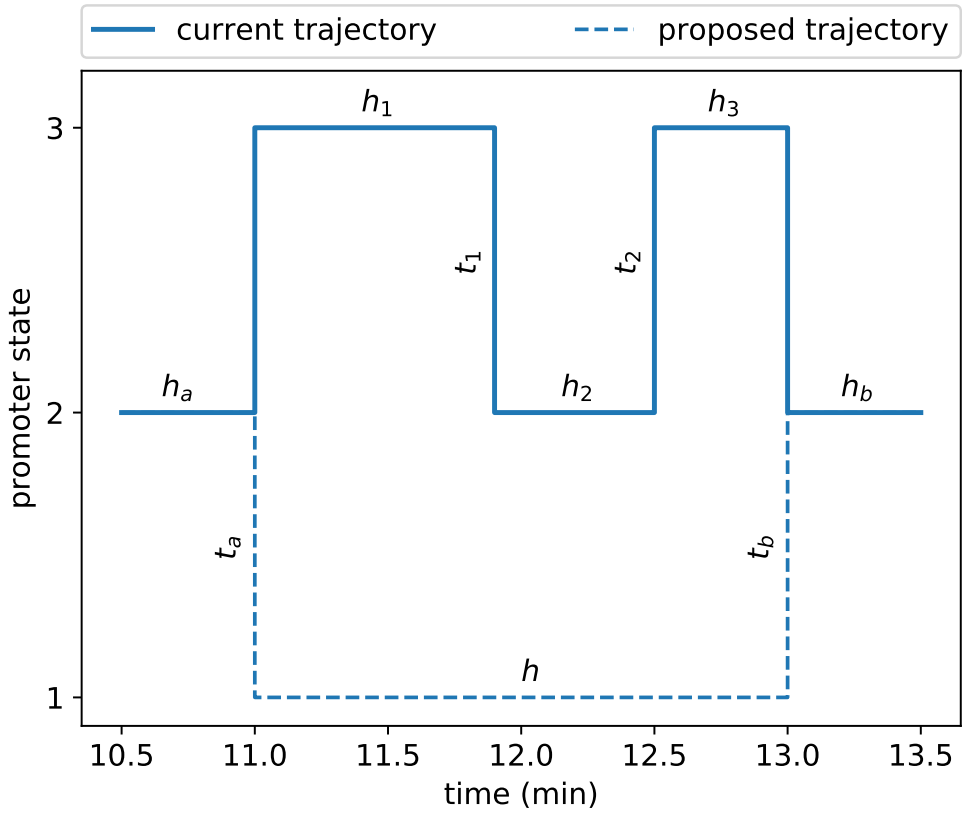


Figure 4.5: Illustration of the Death move. One promoter trajectory is selected at random and two changepoints are removed. Instead of having three steps h_1, h_2, h_3 between h_a and h_b, there will be one new step h.

Algorithm 10: Death move

`// choose a promoter trajectory to update:`
$m \sim \mathcal{U}(\{1, \ldots, M\})$;
$cps, hs \leftarrow p_m$;
`// choose the position from where to delete the changepoints:`
$i \sim \mathcal{U}(\{2, \ldots, |cps| - 3\})$;
`// choose a new value from all possible one-step paths from` h_a `to` h_b`:`
$h \sim \mathcal{U}(Paths_{Death}(h_a, h_b))$;
`// delete two changepoints:`
$cps' \leftarrow \left[cps_0, \ldots, cps_{i-1}, cps_{i+2}, \ldots, cps_{|cps|}\right]$;
`// replace three values by one:`
$hs' \leftarrow \left[hs_1, \ldots, hs_{i-2}, h, hs_{i+2}, \ldots, hs_{|hs|}\right]$;

$Paths_{Death}(h_a, h_b)$ is the set of all one–step paths from h_a and h_b. Since originally there were three steps from h_a to h_b, we need to consider three cases:

- $|h_b - h_a| = 0$: there are at most 2 one–step paths from h_a to h_b: going through $h_a + 1$ or going through $h_a - 1$.

- $|h_b - h_a| = 2$: there is only one path: going through $(h_a + h_b)/2$.

- $|h_b - h_a| = 4$: there are no possibilities of going in one step from h_a to h_b. However, since we will be using models of up to four states, we will not encounter this case.

The probability of the proposal is:

$$Q(k', r', \sigma', p'|k, r, \sigma, p) = P(\text{Death move}) \ \frac{1}{M} \ \frac{1}{|cps| - 4} \ \frac{1}{|Paths_{Death}(h_a, h_b)|} \quad (4.12)$$

At the moment of calculating acceptance probabilities, each Birth move should be considered together with its reverse Death move. Let us suppose a Birth move takes us from a state (k, r, σ, p) to a state (k', r', σ', p) (the corresponding Death Move would take us from (k', r', σ', p') to (k, r, σ, p)). Then the ratio used in the acceptance probability $\alpha(k, r, \sigma, p \to k', r', \sigma', p')$ is:

$$\begin{aligned}(\text{likelihood ratio}) \times (\text{prior ratio}) \times (\text{proposal ratio}) \times (\text{Jacobian}) = \\ \frac{P(f_i|p_i', r, \sigma)}{P(f_i|p_i, r, \sigma)} \times \frac{P(p_i'|k)}{P(p_i|k)} \times \frac{(t_b - t_a)^2}{2} \frac{|Paths_{Birth}|}{|Paths_{Death}|} \times 1\end{aligned} \quad (4.13)$$

The Jacobian factor equals one since the Birth and Death moves do not modify the values of existing parameters [FJSW09].

The ratio used in the acceptance probability of the Death move is the exact opposite of this.

The final summary of the algorithm is the following:

Algorithm 11: Sampler

$k, r, \sigma, p \leftarrow$ initial values;
for *iter = 1, ...* **do**
  Move $\sim$ P(moves);
  $k', r', \sigma', p' \sim \text{Move}(k, r, \sigma, p)$;
  switch *Move* **do**
    case *r move* **do**
      $R = \prod_{i=1}^{M} \left(\frac{P(f_i|p_i,r',\sigma)}{P(f_i|p_i,r,\sigma)} \right) \frac{\mathcal{N}_a^b(r_i;r_i',\sigma_r)}{\mathcal{N}_a^b(r_i';r_i,\sigma_r)}$
    end
    case *σ move* **do**
      $R = \prod_{i=1}^{M} \left(\frac{P(f_i|p_i,r,\sigma')}{P(f_i|p_i,r,\sigma)} \right) \frac{\mathcal{N}_{\sigma_{min}}^{\sigma_{\max}}(\sigma;\sigma',5)}{\mathcal{N}_{\sigma_{min}}^{\sigma_{\max}}(\sigma';\sigma,5)}$
    end
    case *k move* **do**
      $R = \prod_{z=1}^{M} \left(\frac{P(p_z|k')}{P(p_z|k)} \right) \frac{\mathcal{N}_{k_{min}}^{k_{max}}(k_{i,j};k_{i,j}',\sigma_k)}{\mathcal{N}_{k_{min}}^{k_{max}}(k_{i,j}';k_{i,j},\sigma_k)}$
    end
    case *t move* **do**
      $R = \frac{P(f_i|p_i',r,\sigma)}{P(f_i|p_i,r,\sigma)} \frac{P(p_i'|k)}{P(p_i|k)} \frac{\mathcal{N}_a^b(t;t',\sigma_t)}{\mathcal{N}_a^b(t';t,\sigma_t)}$
    end
    case *Birth move* **do**
      $R = \frac{P(f_i|p_i',r,\sigma)}{P(f_i|p_i,r,\sigma)} \frac{P(p_i'|k)}{P(p_i|k)} \frac{(t_b - t_a)^2}{2} \frac{|Paths_{Birth}|}{|Paths_{Death}|}$
    end
    case *Death move* **do**
      $R = \frac{P(f_i|p_i,r,\sigma)}{P(f_i|p_i',r,\sigma)} \frac{P(p_i|k)}{P(p_i'|k)} \frac{2}{(t_b - t_a)^2} \frac{|Paths_{Death}|}{|Paths_{Birth}|}$
    end
  end
  Accept Move with probability $\min(1, R)$;
end

Chapter 5

Synthetic data experiments

We proceed to evaluate the performance of the algorithm in a variety of synthetic data experiments. Unless otherwise noted, the following constants were used in the generation of synthetic data, to simulate experimental conditions:

- Time resolution of the measurements: 20 sec.
- Magnitude of the error in the fluorescence: 55 a.u.
- Length of the traces: 40 min.
- Number of traces: 100.

5.1 Convergence to the stationary distribution

We first analyzed the convergence of the Markov chain to the stationary distribution, as iterations proceeded. Convergence was evaluated using the Gelman–Rubin indicator, described in Chapter 2. In Figure 5.1, we plot the convergence indicator for each of the fixed–dimensional parameters and for the logarithm of the posterior probability, against the number of iterations. As described before, as a rule of thumb a value less than 1.1 is enough for practical purposes.

In this case, $5 \cdot 10^8$ iterations would have been enough to have approximate convergence. Even if the Gelman–Rubin statistic for r_2 does not seem to behave as nicely as for other parameters, it is within acceptable values. More iterations always increase accuracy, but of course a trade–off is made with computational cost. We settled for using $4 \cdot 10^9$ iterations for the rest of the synthetic and real data situations encountered in this work. This was enough to achieve approximate convergence, except for one case, described in 5.5, where the Gelman–Rubin statistic suggested more iterations would have been beneficial.

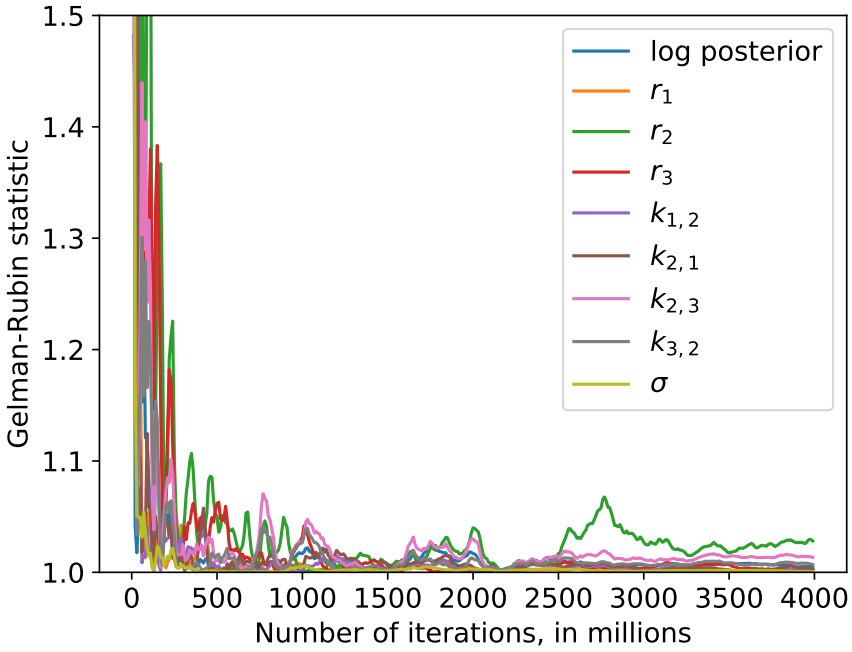


Figure 5.1: Monitoring convergence in relation to the number of iterations. Data was generated with the following parameters: three states, $k_{1,2} = k_{2,1} = k_{2,3} = k_{3,2} = 1$ event/min, $r = [0, 12.5, 25]$ pol/min. Two MCMC chains were simulated, up to $4 \cdot 10^9$ iterations each. While evaluating the Gelman–Rubin indicator, the first half of the chain was discarded (the *burn–in* period). The Gelman–Rubin indicator is plotted for each of the fixed–dimensional parameters and for the logarithm of the posterior density.

The number of iterations needed increased in the course of this project, as we extended the model—increasing the running time of the algorithm as well and making development more cumbersome. One of the reasons for the large number of iterations required is that in one iteration maybe only one changepoint of one of the ≈100 nuclei trajectories is updated. It could be interesting to seek variants of the algorithm that allow updating all nuclei trajectories in parallel.

We also suspect the large number of iterations required has to do with the algorithm spending too much time exploring changepoints very close to each other, which do not affect the fluorescence much and have a small effect on the inference result. An attempt to remediate this would be to forbid the formation of changepoints too close to each other, however, the implementation of a proposal distribution would turn awkward. A more natural approach would be to move to a discrete model: the promoter trajectory would be represented with a fixed number of steps of a fixed length. Disallowing changepoints very close to each other would have the added benefit of helping protect against overfitting.

5.2 Fitting the fluorescence

Figure 5.2 illustrates the algorithm fitting the fluorescence as iterations proceed. The algorithm starts with a constant promoter trajectory and fluorescence (Figure 5.2a). After 1000 iterations, it has added two changepoints to the promoter trajectory, which increases the fit of the fluorescence by some amount. An observation: it may seem slow to need 1000 iterations to reach this state; however, the algorithm is concurrently updating the promoter trajectories of another 100 nuclei. After a million iterations, sufficient changepoints have been added to the promoter trajectory to achieve a reasonable fit of the fluorescence.

It is not clear from the figure whether the original promoter trajectory has been recovered correctly. The inferred promoter trajectory may differ from the original, especially in short segments between changepoints close to each other, that have a negligible effect on the observed fluorescence. In section 5.3, we will discuss how well the original number of changepoints is inferred.

While we can say the chain has already reached a region of high posterior probability, many more iterations will be needed to ensure all the space of high probability is explored.

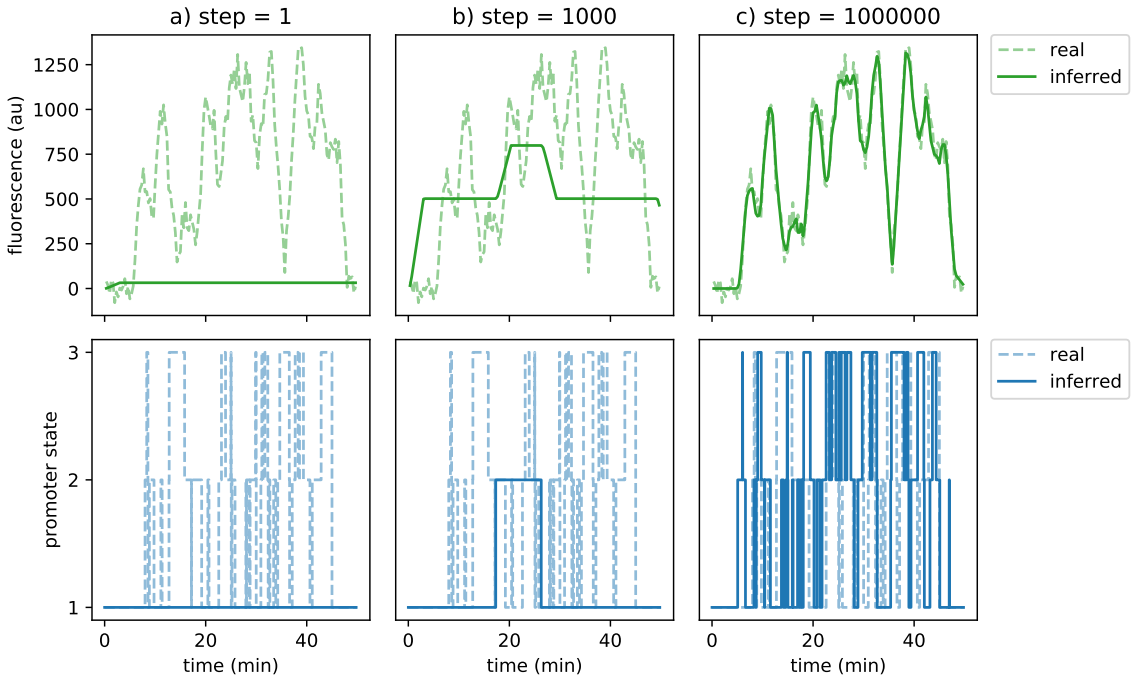
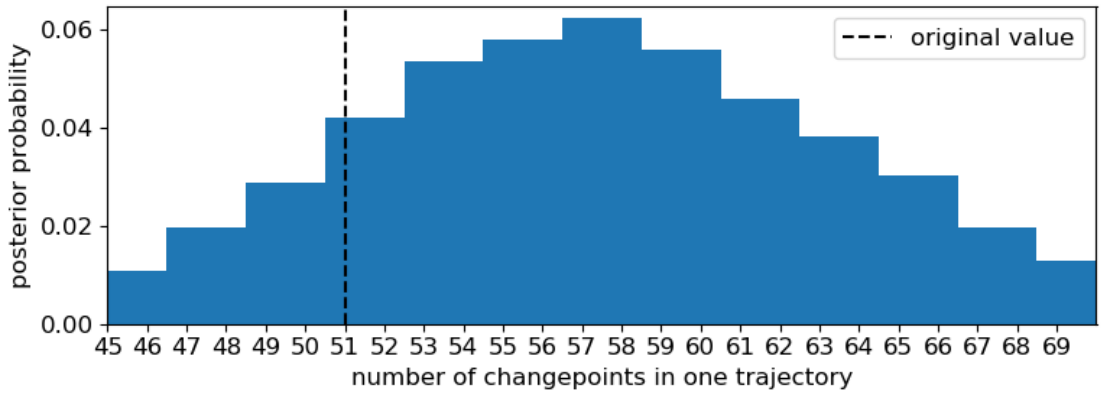


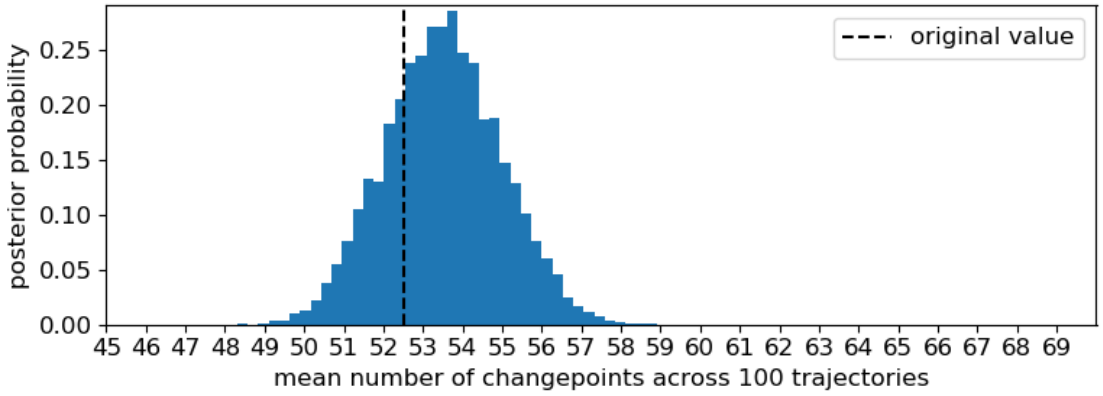
Figure 5.2: Some steps of the algorithm fitting the fluorescence. Data was generated with the following parameters: three states, $k_{1,2} = k_{2,1} = k_{2,3} = k_{3,2} = 1$ event/min, $r = [0, 12.5, 25]$ pol/min. Below, comparison between the original promoter trajectory, used to generate the fluorescence, and the value of the promoter trajectory at the current step of the algorithm. Above, comparison between the original fluorescence and the fluorescence expected given the value of the promoter trajectory at the current step.

5.3 Inferring the number of changepoints

We let iterations proceed in the previous experiment, and evaluated the inference of the number of changepoints in the original trajectory. Figure 5.3a shows that there is a lot of uncertainty in the predicted number of changepoints of that single promoter trajectory. As stated earlier, changepoints very close to each other are nearly impossible to detect, as they do not affect the observed fluorescence much. However, when we look at the average number of changepoints inferred across 100 promoter trajectories, we see that the algorithm performs quite well (Figure 5.3b). Uncertainty has been reduced a lot, and the average difference between the inferred number of changepoints and the original value is not more than one.



(a) Posterior distribution of the number of changepoints in a single promoter trajectory, compared to the original number in the trajectory that was used to generate the fluorescent data.

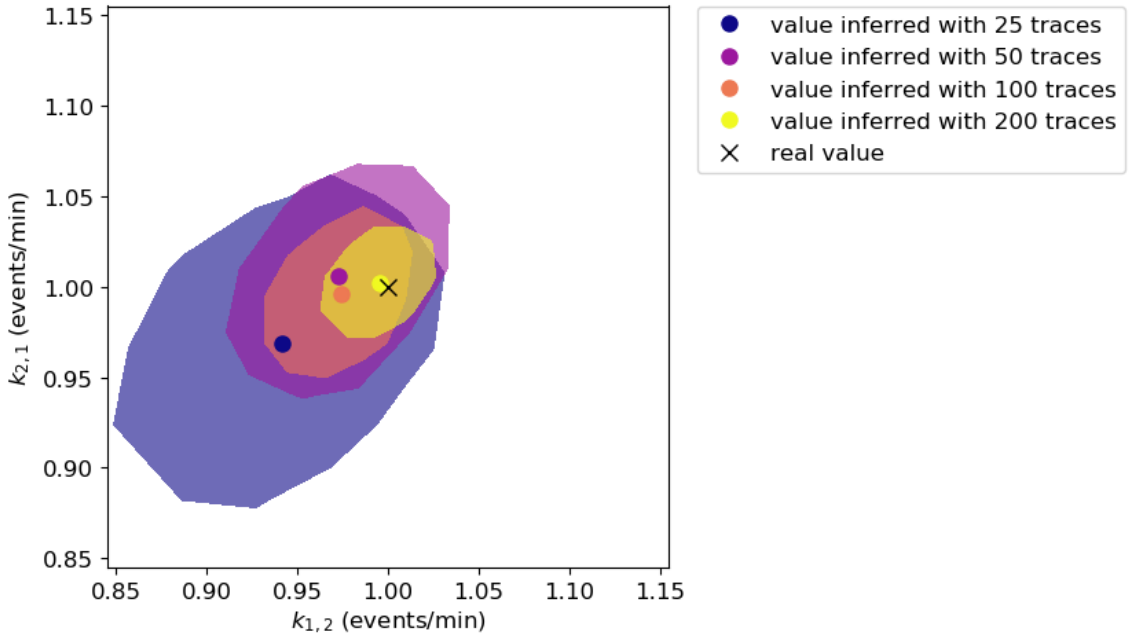


(b) Posterior distribution of the mean number of changepoints across all promoter trajectories, compared to the original value in the trajectories that were used to generate the fluorescent data.

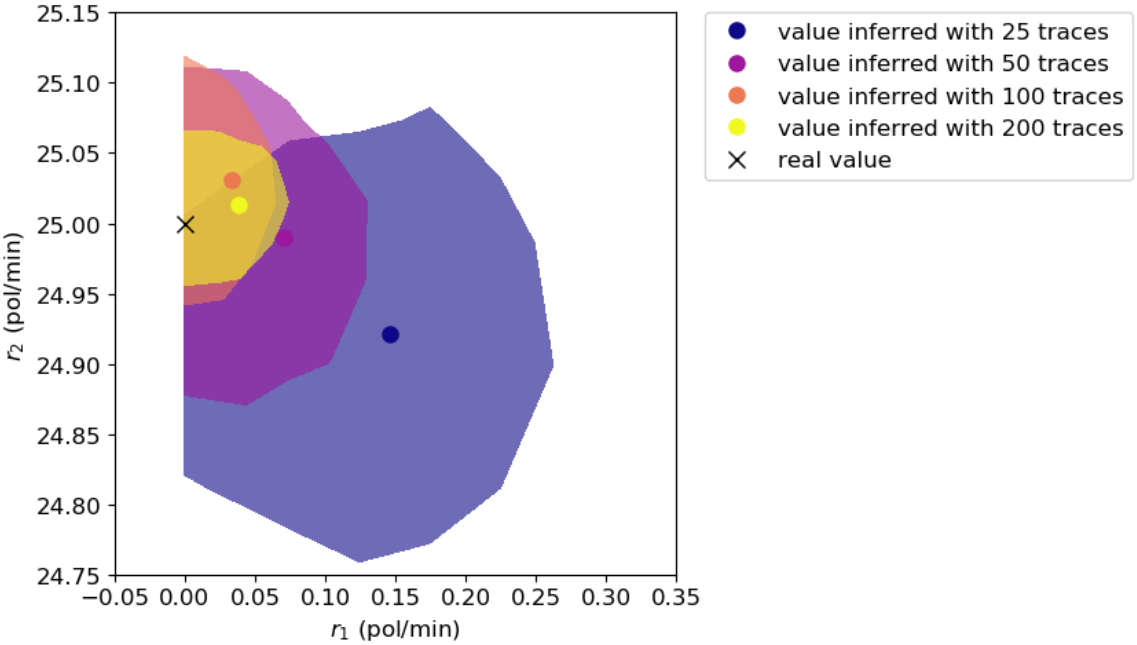
Figure 5.3: Comparison of the inference of the correct number of changepoints when (a) considering a single trace vs. (b) taking the average of the 100 trajectories.

5.4 Statistical performance and number of traces

We sought to investigate the effect of the number of traces available on the statistical performance of the algorithm. We generated 200 traces and compared the result of the inference when only using a subset of the traces (Figure 5.4). As the number of traces increases, it is clear how the prediction gets closer to the real value, and at the same time the uncertainty in the prediction is reduced. With only 25 traces, results were already within a close margin of the real value, but is should be noted that this is the two–state model, the simplest one, chosen here for ease of visualization. With more states and parameters to infer, the advantage of having more traces becomes clearer.



(a) Results of the inference of transition rates.

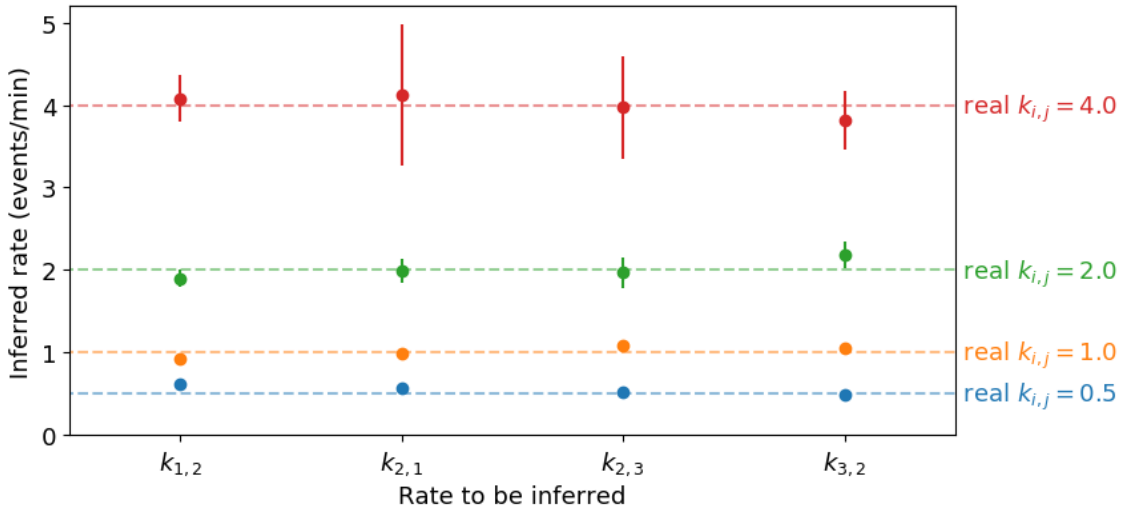


(b) Results of the inference of polymerase loading rates.

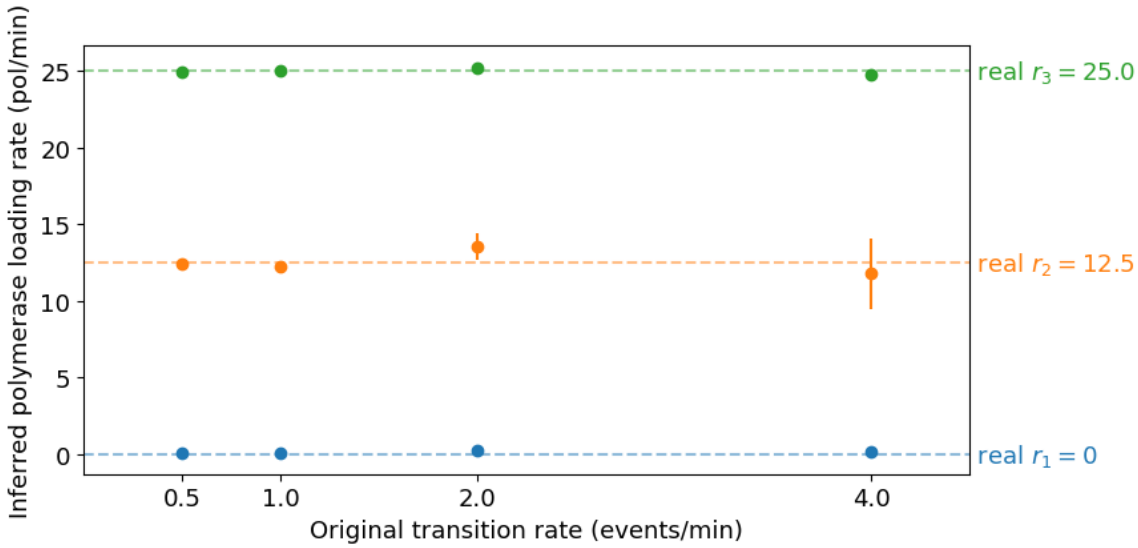
Figure 5.4: Statistical performance in relation to the number of traces. 200 traces were generated with the following parameters: two states, $k_{1,2} = k_{2,1} = 1$ event/min. Then, inference was performed by using only a subset of the traces. Results are shown for (a) transition rates and (b) polymerase loading rates. For each inference, the dot marks the mean of the posterior distribution and the shaded area denotes the one standard deviation credible region.

5.5 Statistical performance and transition rates

Next, we studied how the algorithm behaves as we make the system move faster, by increasing transition rates. For simplicity and clarity of exposition, we have chosen all transition rates to be equal, however, we have not seen any difference in behaviour when rates are different from each other. The results can be seen in Figure 5.5. The predictions keep close to the original values, even if the uncertainty in the predictions increases. In the case of the rates being equal to 4.0 events/min, uncertainty increases a lot. Inference inevitably becomes more difficult as we increase the transition rates, since we have a fixed time resolution in the measurements. In fact, in the case of the highest rates (4.0 events/min) the Gelman–Rubin statistic indicated the MCMC chain had not reached convergence, suggesting more iterations should be made to achieve a greater accuracy, but what would make the running time less practical. Still, we can see the prediction is quite good.



(a) Inference of transition rates. Moving upwards, we increase the transition rates with which we generate the data.



(b) Inference of polymerase loading rates. Moving from left to right, we increase the transition rates with which we generate the data.

Figure 5.5: Statistical performance in relation to transition rates. Four synthetic datasets were generated, with three states and all the rates equal to 0.5, 1, 2 and 4 events/min. Results are shown for (a) transition rates and (b) polymerase loading rates. For the result of each inference, the mean and the one standard deviation credible interval is plotted, against the dashed line that represents the original value.

5.6 Model selection

Besides trying to assess the fit of parameters of a model to the data, we tried to assess the fit of different models to the data. This is the problem of model selection, an inference at another level, requiring a different set of computations from those used to infer the parameters.

The principled Bayesian approach to model selection involves calculating the marginal likelihood (see Equation 2.4) for each model, but this is generally intractable. Another approach for model selection, that has proved very useful in practice, is cross–validation [AC+10]. However, it is not always applicable to unsupervised settings like this one (our data does not include traces labeled with their underlying parameters). An approach we thought could work was using the Watanabe–Akaike information criterion (WAIC) [Wat10], an information criterion specifically designed for MCMC chains. The WAIC score is calculated for each model, and the lower the score, the more the model is favored by the data.

We generated synthetic data with one model, and then calculated the WAIC score for the results obtained by running the algorithm assuming each of the models. We would have expected the model used to generate the data to achieve the lowest WAIC score, but we find inconsistent results, that should be further analyzed. For example, in Table 5.1 we display the scores for each model when data is generated by the three–state model. We can see the four–state model is chosen as the most plausible model given the data, as it achieves the lowest WAIC score (the difference between scores is what matters, not the absolute values).

Two–state model	Three-state model	Four–state model
170086	168790	168584

Table 5.1: WAIC scores of the three models, for data generated under the three–state model. The lower the score the more plausible the model. The difference between scores is what matters, not the absolute values.

We think the problem could be an ill–posed one, since given enough changepoints, any model can approximate any of the others quite well.

Chapter 6

Analysis of experimental data

Fluorescent traces of transcriptional activity of the *eve* gene during nuclear cycle 14 were obtained from 9 fruit fly embryos, using the technique described in Chapter 1. Time resolution of the measurements was 22.5 seconds, and trace length varied from 5 to 35 minutes. Nuclei were grouped into five bins along the anterior–posterior (AP) axis: $\{-2, -1, 0, 1, 2\}$, as shown in Figure 6.1. The rationale for grouping nuclei according to their AP position is that transcription factors governing *eve* expression are modulated along the AP axis (Figure 1.5(B)).

We proceed to analyze the fluorescent traces. Armed with a new tool for investigating transcriptional bursting, we will try to learn more about its role in transcriptional regulation in the developing embryo.

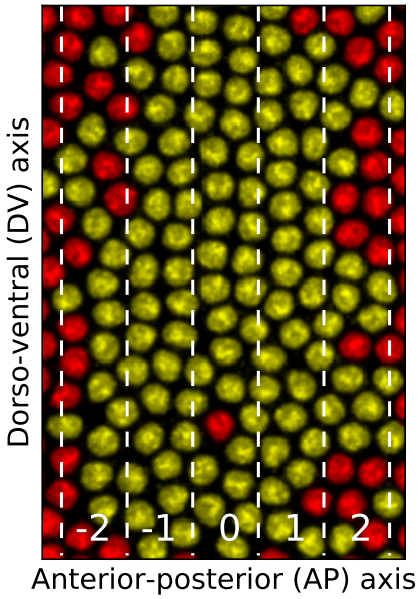


Figure 6.1: Grouping of nuclei into five bins along the anterior–posterior axis. 0 corresponds to the center of *eve* stripe 2 ($\approx 40\%$ of the AP length), while -2 corresponds to the most anterior section of the stripe and 2 to the most posterior section.

6.1 Preliminary analysis

Before jumping into the inference of transcriptional bursting dynamics, we can make useful observations just by looking at averaged properties of the traces. Figure 6.2 shows the modulation of different statistics of the traces along the AP axis. Total mRNA produced in a nucleus is proportional to the accumulated fluorescence of the trace. There is a substantial modulation in the amount of mRNA produced—nuclei at the center of the stripe produce on average twice as much mRNA than nuclei at the borders of the stripe.

We may wonder if the cause of this difference in transcription levels could be in transcriptional bursting dynamics—at the center of the stripe the promoter could spend more time in active states, for example. However, we must first acknowledge the importance of the transcription time window: the total time a nucleus engages in transcriptional bursting. The transcription time window is also modulated along the AP axis: bursting lasts an average of 24 min. on the center of the stripe and around 16 min. on the borders. Thus, the modulation of the transcription time window is an important factor determining total mRNA produced. If we divide total mRNA by the transcription time window, we get the amount of mRNA produced per minute, which is modulated less sharply than total mRNA produced. This is the quantity which we want to characterize by changes in the rates of transcriptional bursting along the AP axis.

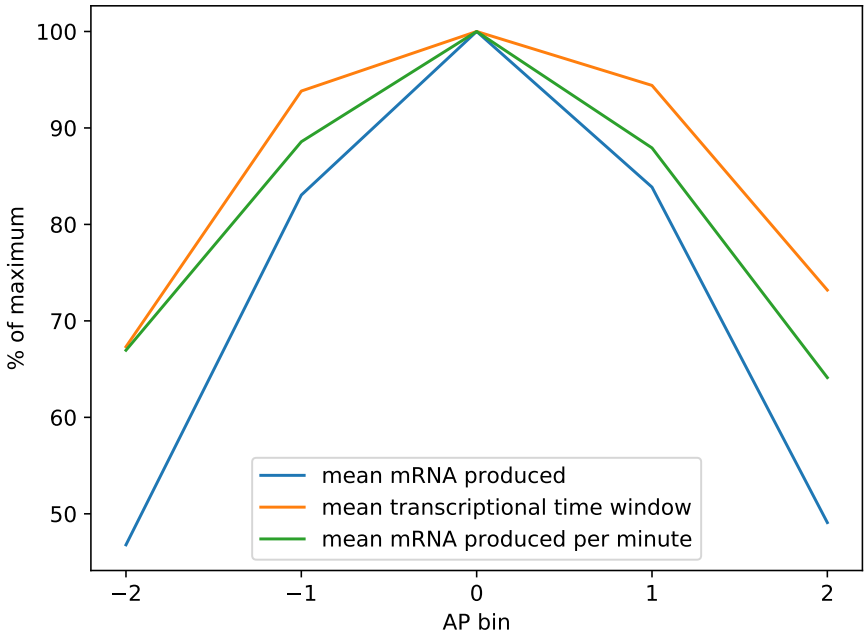


Figure 6.2: Modulation of different quantities along the AP axis. Each variable is plotted as a percentage of the maximum.

6.2 Transcriptional bursting

We ran the inference algorithm on the five AP bins and under three models: a two–state model, a three–state model and a four–state model. Figure 6.3 shows a typical trace fitted by the three models. The different models approximate the fluorescence quite similarly, even if they have vastly different promoter trajectories. In particular, none of the models can recover certain features of the fluorescence, such as very sharp peaks. A certain possibility is that they are just image processing artifacts. Ongoing work should clarify which discrepancies could be attributed to the image processing, the model, or the inference algorithm.

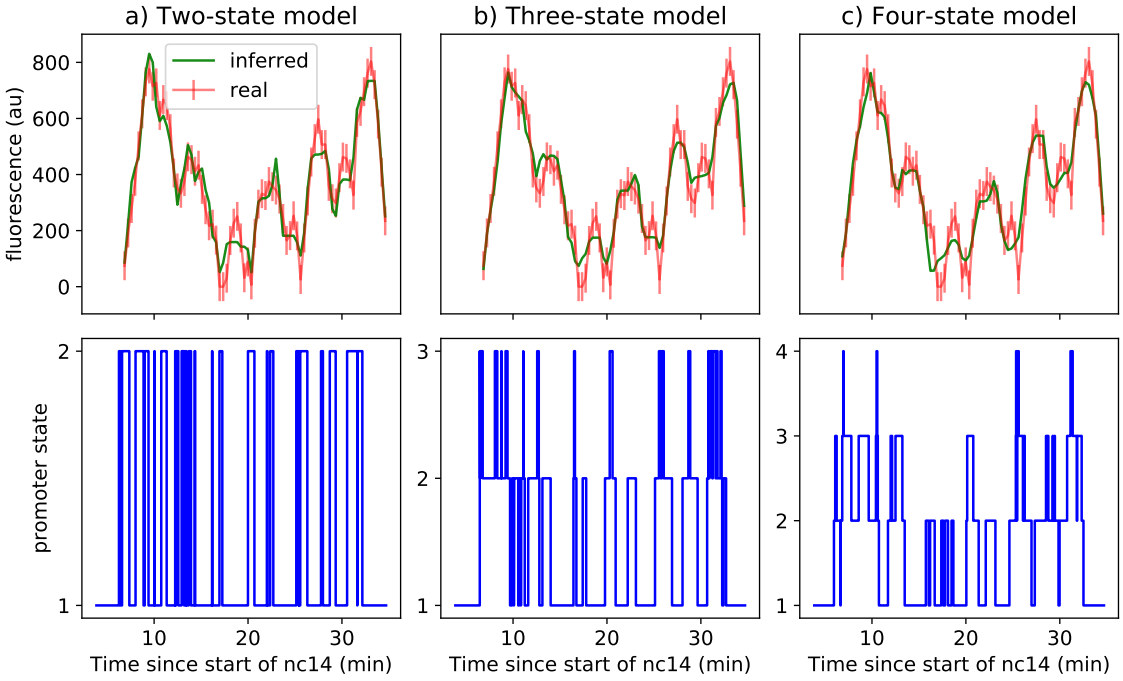


Figure 6.3: Typical fluorescent trace fitted to the three models. On the bottom, a sample from the posterior distribution of the promoter trajectory. On top, a comparison between the simulated and the observed fluorescence.

Transition rates

Figure 6.4 shows the inferred transition rates. All rates show some modulation along the AP axis, with different different degrees of symmetry with respect to the center of the stripe. The high error bars of $k_{3,2}$ in the three–state model, and particularly $k_{4,3}$ in the four–state model, stem in great part from the low occupancy of the highest state in those models (see Figure 6.6). It remains to be seen whether the high value of $k_{4,3}$ is physically realistic.

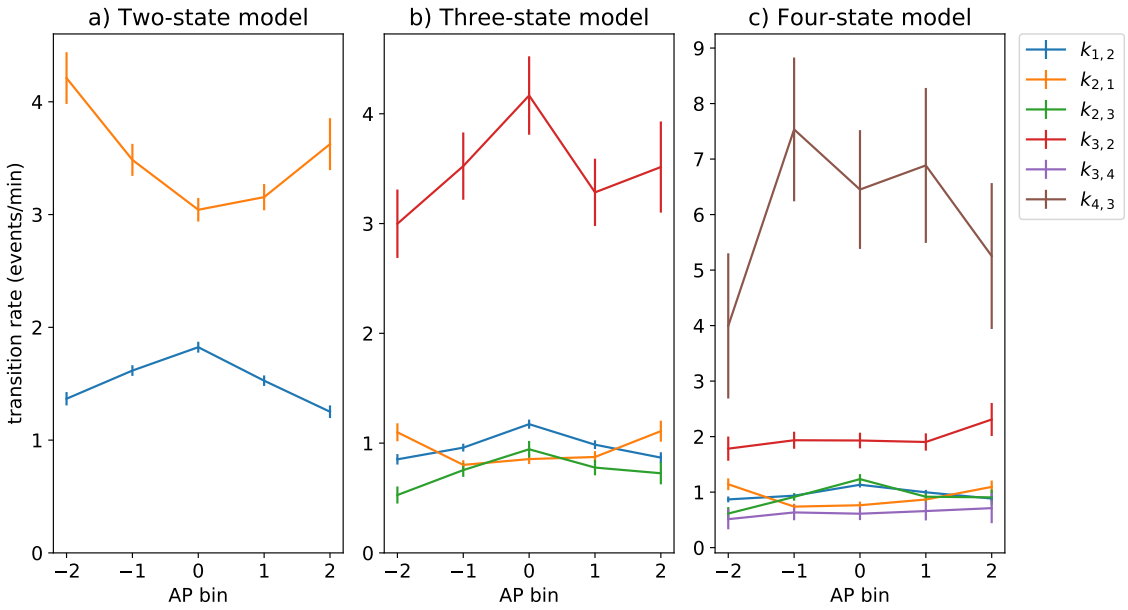


Figure 6.4: Inferred transition rates across AP bins and models.

Polymerase loading rates

Inferred polymerase loading rates are represented in Figure 6.5. In comparison to transition rates, polymerase loading rates are relatively stable along the AP axis. Also worth noting is that r_1, the loading rate of the lowest state, is consistently zero across the three models. Again, there is a lot of variance in r_4 due to the low occupancy of the fourth state.

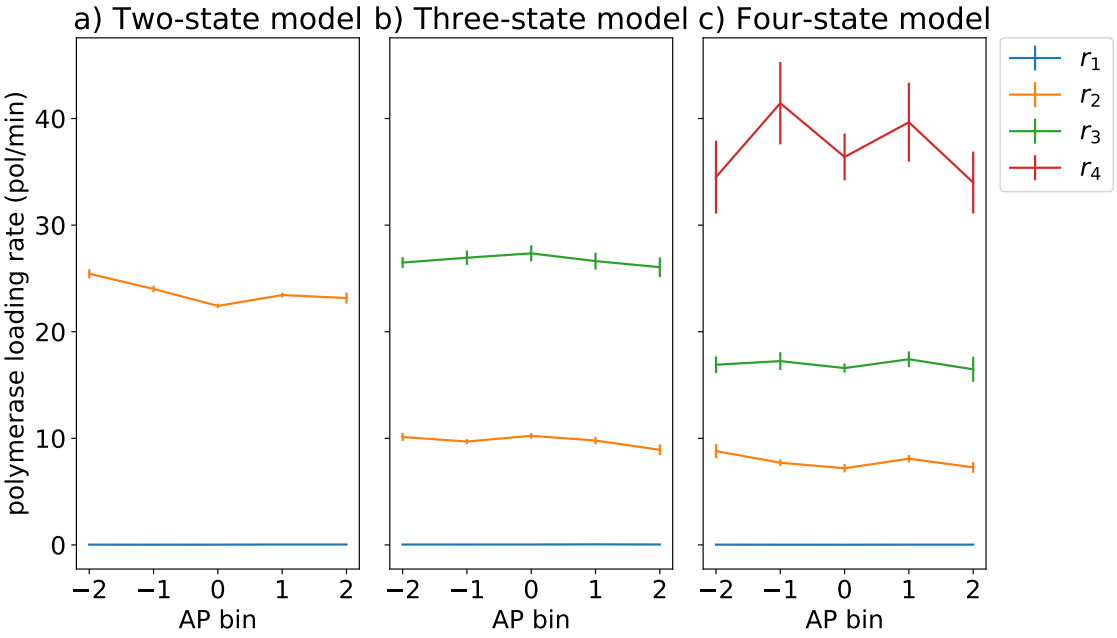


Figure 6.5: Inferred polymerase loading rates across AP bins and models.

State occupancy

The percentage of time spent at each state is depicted in Figure 6.6. All models predict a lower occupancy of the lowest state (of nearly zero transcriptional activity) in the center of the stripe, as well as higher occupancy of the highest states. The modulation achieved is quite symmetrical, even if the transition rates, which ultimately determine the occupancy, do not show total symmetry. In the four–state model, the occupancy of the highest state is around 2%, which raises doubts about its biological significance. This fourth state might not be a real state, but rather an evidence of overfitting.

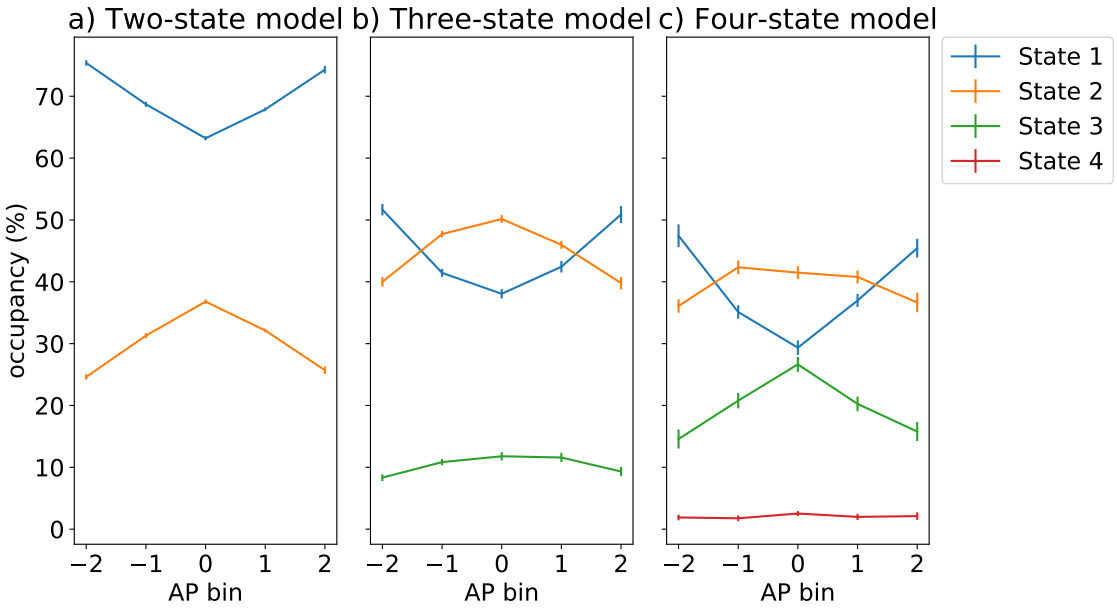


Figure 6.6: Inferred occupancy of the different states, across AP bins and models.

Fluorescence error

Figure 6.7 shows the inferred fluorescence error. The result in the three models is similar; there is a consistent ordering, with the two-state model inferring lower errors, but the difference is very small. Fluorescence error decreases towards the borders of the stripe; it would be interesting to see how this connects to the variance in background fluorescence, the main contributor to measurement error.

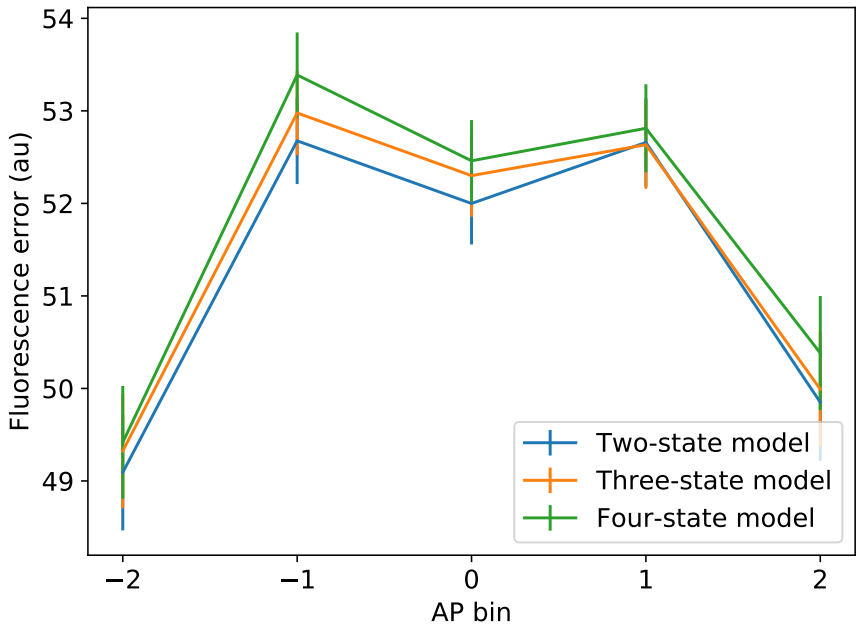


Figure 6.7: Inferred fluorescence error, across AP bins and models.

A window into time–varying rates

Our model assumes rates are constant in time. However, we can get a preview on time–modulation of transition rates by sliding a small time window and counting the transitions that occur in that window. This is not a valid way of inferring rates, since we do this analysis *after* fitting the model that assumes constant rates. However, some conclusions can be drawn. If rates were constant in time, inferred local rates for each time window could oscillate but would statistically move around the true rate. However, our results suggest many of the rates are indeed modulated across time (Figure 6.8). It would be interesting to compare these trends to how the concentration of transcription factors varies in time.

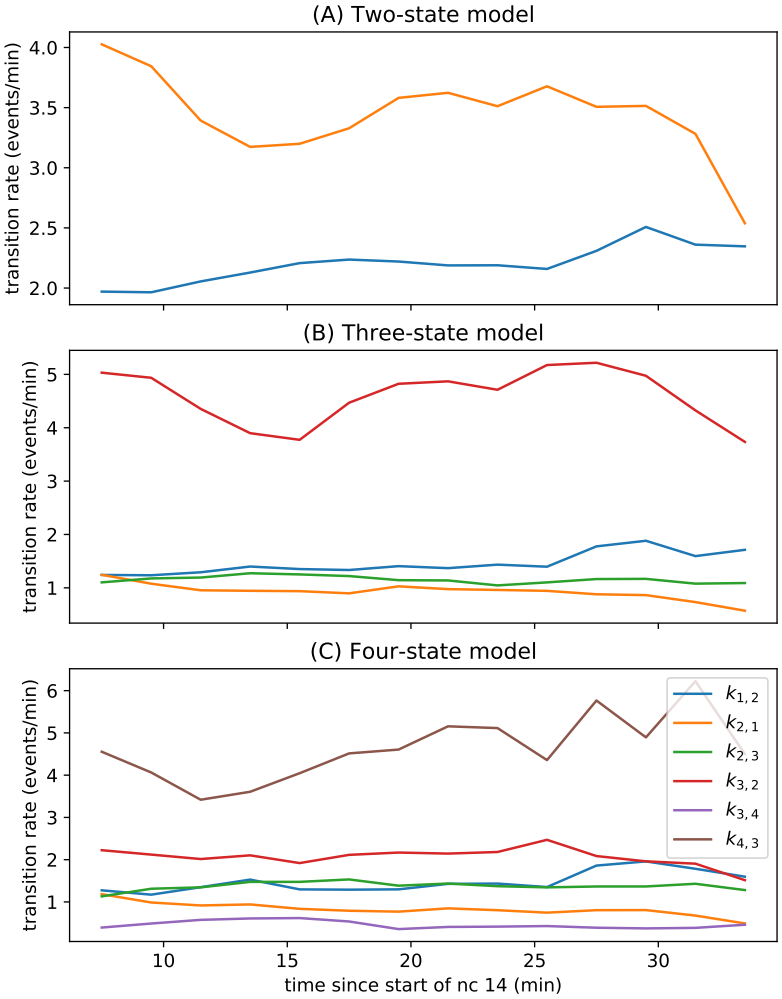


Figure 6.8: Preliminary view of time–varying transition rates. A time window of 3 minutes was slid every 2 minutes, and transition rates were estimated for each window.

6.3 Final remarks

We have seen how control of mRNA production in the second stripe of the *eve* gene can be achieved by the joint modulation of the transcription time window and bursting dynamics—primarily transition rates, apparently. We have presented three models, but further work is needed to assess the plausibility of each one of them.

A very interesting direction would be trying to find functional relationships between bursting rates and the concentration of transcription factors. New technology is being developed to allow real–time tracking of the concentration of transcription factors. If we could adapt the algorithm to infer time–varying rates, and match these with the concentration of transcription factors, we could ask a lot of new questions, for example:

- Does the concentration of Kruppel modulate the transition rate from the ON to the OFF state?
- What molecular mechanism may be behind the repression by Kruppel?
- May certain rates going to infinity/zero be determining the transcription time window?

Chapter 7

Conclusion

In this thesis, we sought to develop an algorithm to infer transcriptional dynamics from single–molecule fluorescence traces. We made some first steps tackling novel experimental data from the fruit fly embryo, with the challenge of having no proven solutions at the time of our development.

We were able to see through seemingly chaotic fluorescent traces and find the underlying regularities of the system, about which we can now think more clearly. Our work showcases the important role computation can have in biological research, confronting the complexity and noise of experimental data, and extracting meaningful information that can be interpreted by theorists.

The method

Our model describes the transition of the gene promoter between different ON and OFF states, and the loading of polymerase molecules at a constant rate within each state. A more detailed model would look at the location of each individual polymerase loading event. However, it is not clear this inference would be possible given the noise present in the current experimental setup. Also, we do not yet know how much elongation time could vary—our current approach, which assumes it to be constant, could become unfeasible. It would be very helpful to better understand physical and biological limits of the rates, to constrain the quite–flexible promoter trajectory representation we have.

We have chosen a continuous changepoint representation of the promoter trajectory, and the RJMCMC algorithm we have developed has been quite flexible and performant to date. Still, moving to a discretized promoter trajectory, represented by a fixed number of steps, could confer some advantages:

- It would prevent the proliferation of super–small steps, that waste exploration time and are prone to overfitting.

- Having a fixed number of parameters would ease development in general, for example, extending the model with time–varying rates.

- A fixed number of parameters would give access to many other inference algorithms and software libraries.

It remains to be seen if a discretization could be found that is both flexible enough to represent the dynamics of the system and tractable.

Results

We analyzed fluorescent traces of transcriptional activity of the *eve* gene in the fruit fly embryo, during the formation of the second stripe of expression. Our results suggest that transcriptional control in the second stripe is achieved by the joint modulation across the AP axis of the transcriptional time window and of the transition rates governing bursting dynamics. Analysis of results also points out that transition rates are probably changing in time as well.

We have not yet devised a testable criterion for comparing different models, but the low occupancy of the highest states raises doubts about their physical existence.

Future work

A natural next step would be to extend the algorithm to allow for time–varying transition and polymerase loading rates, while keeping statistical accuracy and computation time reasonable. It could be necessary to collect more data.

Once we can infer how rates vary across time and space, and we measure the concentration of transcription factors, we can start asking questions about what the functional relationship between them could be. For example, does the concentration of Kruppel determine the transition rate from the ON to the OFF state? Understanding these functional relationships would also cast a light on how repression or activation may be implemented at the molecular level.

Finally, we could have a computational model, but would we be able to experimentally test it? We should devise a way to perturb the concentration of transcription factors, and check how well our predictions hold.

Bibliography

[A+14] B Alberts et al. *Molecular Biology of the Cell, Sixth Edition*. Garland Sci, New York, 2014.

[AC+10] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[BGE+14] Jacques P Bothma, Hernan G Garcia, Emilia Esposito, Gavin Schlissel, Thomas Gregor, and Michael Levine. Dynamic regulation of eve stripe 2 expression reveals transcriptional bursts in living drosophila embryos. *Proceedings of the National Academy of Sciences*, 111(29):10598–10603, 2014.

[CGH+17] Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32, 2017.

[FJSW09] Emily Fox, Michael I Jordan, Erik B Sudderth, and Alan S Willsky. Sharing features among dynamical systems with beta processes. In *Advances in Neural Information Processing Systems*, pages 549–557, 2009.

[GCS+13] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013.

[GM95] Peter Guttorp and Vladimir N Minin. *Stochastic modeling of scientific data*. CRC Press, 1995.

[GR92] Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statist. Sci.*, 7(4):457–472, 11 1992.

[Gre95] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[GTLG13] Hernan G Garcia, Mikhail Tikhonov, Albert Lin, and Thomas Gregor. Quantitative imaging of transcription in living drosophila embryos links polymerase activity to patterning. *Current biology*, 23(21):2140–2145, 2013.

[Has70] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[RA86] Adrian Elmes Raftery and VE Akman. Bayesian analysis of a poisson process with a change-point. *Biometrika*, pages 85–89, 1986.

[Rob] https://commons.wikimedia.org/wiki/File:Robot_metaphor.png.

[Wat10] Sumio Watanabe. Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11(Dec):3571–3594, 2010.