HMM project: Application to Genetics

Léo HOUAIRI, Yao Pacome KOUAME, Théo LORTHIOS and Sonali PATEKAR

January 2023

1 Introduction

This project is based on the research article **"Parseq: reconstruction of microbial transcription landscape from RNA-Seq read counts using state-space models"** (Mirauta et al., 2014) [1]. The goal of the paper is to develop a way to reconstruct the transcription level accross the genom based on the read counts produced by modern RNA-sequencing techniques. Shortly, the transcriptome is the set of all RNA molecules in a cell at a given time. It cannot be observed directly, but RNA-Seq allows the observation of read counts, a proxy of the transcription level. Appendix D provides more details about RNA-Seq techniques and the line of work developed in the paper.

An interesting way of modelling this problem is to see the genome as a space and to use a state-space model. The transcription level at each point of the genome is modeled as a Markov chain, and an emission law links the oberserved values (or read counts) to this transcription level. In practical settings, only the read counts are observed and the goal is to recover the unerlying transcription level. However, in this project, we worked on synthetic data and could thus access the hidden variable.

The model developed in the paper tries to account for many sources of varibility. This might lead to good performances, but also induces a complex model, which complicated our implementation.

This report is organized as follows: section 2 presents the model developped in the paper and implemented during this project, section 3 discusses data generation and the manual construction of our Feynman-Kac, section 4 presents the results obtained using the bootstrap filter, section 5 presents our attempt at performing bayesian inference using the PMMH algorithm and section 6 provides a summary of our work and how it could be continued.

Our code can be found at: https://github.com/TheoAlegretti/HMMC-project.
We relied heavily on the **particle** package, see: https://github.com/nchopin/particles. We warmly thank Nicolas Chopin for his assistance throughout the project.

2 The model

We stated in the introduction that the model involves two variables, but there are three:

- $(y_t)_{t>0}$ is the sequence of observations, the read counts.
- $(u_t)_{t>0}$ is the underlying expression level.
- $(s_t)_{t>0}$ is a local scaling variable, that also follows a Markov kernel and influences the emission law.

So if we write X_t the hidden variable, it is actually a vector: $X_t = (u_t, s_t)$.

Following the notations in the supplementary materials of the article, we denote $\mathcal{D}(...)$ the distribution $\mathcal{D}$, and by $\mathcal{D}(x; ...)$ the density of the distribution $\mathcal{D}$ at point x. See appendix A for the conventions we use regarding the different distributions.

[1] Some parts of the report, especially the equations, were directly copied from the paper and its supplementary information.

2.1 The Markov chain for the hidden variable(s)

We start with the simple kernel of s_t. It writes:

$$k_s(s_{t+1}|s_t) = \alpha_s \times \delta_{s_t}(s_{t+1}) + (1-\alpha_s) \times \Gamma(s_{t+1}; shape = \kappa_s, scale = \kappa_s)$$

This kernel simply implies that with probability α_s, $s_{t+1} = s_t$, and with probability $1-\alpha_s$, s_t is sampled from a certain gamma distribution.

The kernel of u_t is much more complicated. It writes :

$$\begin{aligned} k_u(u_{t+1}|u_t) = \mathbb{1}_{\{u_t=0\}} \times \Big[&(1-\eta) \times \delta_0(u_{t+1}) + \eta \times \mathcal{E}(u_{t+1};\zeta)\Big] \\ + \mathbb{1}_{\{u_t>0\}} \times \Big[&\alpha \times \delta_{u_t}(u_{t+1}) + \beta \times \mathcal{E}(u_{t+1};\zeta) + \beta_0 \times \delta_0(u_{t+1}) \\ &+ \gamma_u \mathbb{1}_{\{u_{t+1}>u_t\}} \left(u_t + \mathcal{E}(Z; \frac{\lambda_u}{u_t})\right) + \gamma_d \mathbb{1}_{\{u_{t+1}<u_t\}} \left(u_t - \mathcal{E}(Z; \frac{\lambda_d}{u_t})\right) \Big] \end{aligned}$$

With $\alpha + \beta + \beta_0 + \gamma_u + \gamma_d = 1$; also, u_t is not allowed to take negative values.

This kernel is complicated, let's understand it step by step. First, note that this kernel is divided in two parts: it acts differently depending on whether $u_t = 0$ or $u_t > 0$. In each case, different "moves" are possible and have a certain probability of happening.

If $u_t = 0$, there are two possibilities:

- With probability $(1-\eta)$, $u_{t+1} = 0$.
- With probability η, u_{t+1} is drawn from an exponential.

If $u_t > 0$, there are five possibilities:

- With probability α, the transcription level does not change and $u_{t+1} = u_t$.
- With probability β, u_{t+1} is drawn from an exponential law. This models a **shift**, a big change in the level of transcription.
- With probability β_0, $u_{t+1} = 0$.
- With probability γ_u, the chain experiences a slight raise: $u_{t+1} = u_t + Z$, where $Z \sim \mathcal{E}(\frac{\lambda_d}{u_t})$[2] This models an upward **drift** (little change).
- With probability γ_d, the chain experiences a slight decrease: $u_{t+1} = u_t - Z$, where $Z \sim \mathcal{E}(\frac{\lambda_d}{u_t})$. This models a downard **drift** (little change).

2.2 The emission law

In this model, the emission law - that is, the density of y_t - can be written in two different forms. Both were useful in our work.

We used the first one to generate the observations. It involves two intermediary variables: $a_t \sim \Gamma(\kappa, \theta)$ (the amplification coefficient) and $x_t \sim \mathcal{P}(\frac{u_t s_t}{\kappa\theta})$ (the number of molecules). In this case, the density of y_t writes:

$$e(y_t|x_t, a_t) = (1-\epsilon_b-\epsilon_0) \times \mathcal{P}(y_t; x_t \times a_t) + \epsilon_b \times \mathcal{P}_{-\{0\}}(y_t; a_t) + \epsilon_0 \times \mathcal{U}(y_t; 0...b)$$

The second expression does not depend on those two intermediary variables but depends directly on u_t and s_t. We used it for the computation of the likelihood. It writes:

$$e(y_t|u_t, s_t) = (1-\epsilon_b-\epsilon_0) \times \sum_{x_t=0}^{\infty} \mathcal{P}(x_t; \frac{u_t s_t}{\kappa\theta}) \times \mathcal{NB}(y_t; \kappa, \frac{x_t\theta}{x_t\theta+1}) + \epsilon_b \times \mathcal{NB}_{-\{0\}}(y_t; \kappa, \frac{\theta}{\theta+1}) + \epsilon_0 \times \mathcal{U}(y_t; 0...b)$$

[2]For this exponential law and the next one, we decided of a parametrization that seemed to make sense, even if it might differ from the one used in the paper. From our point of view, the modelling of the drifts was not explained clearly.

Similarly to the Markov transition kernel of the hidden variable, both of these densities are mixtures of densities which are assigned certain probabilities. The authors chosed to use such model in order to account for the fact that the observations y_t can stem from different things, some of them being artifacts. Shortly, they can depend on the underlying transcription level, be background noise or be outliers.

2.3 Initial laws

We did not find initial distributions in the paper nor in the supplementary materials. For both data generation and the bootstrap filter, we chose to intialize u_t and s_t at 0 (the distribution is a Dirac). In both cases, the chain will rapidly jump from zero to a positive value: it happens with probability η for u_t (0.05 in our implementation) and probability $1 - \alpha_s$ for s_t (0.47 in our implementation).

3 Data generation and Feynman-Kac definition

The model we analyze is complicated, especially because of the presence of indicator function. Due to this, we could not define a **SSM** using the package particles. Instead, we used a set of functions to generate data ourselves and defined manually the Feynman-Kac model (see the notebook Manual definition of Feynman-Kac).

We relied primarily on 3 functions. Function **stepX** takes as input X_t and some parameters and returns a sample from X_{t+1} (given X_t); **sampleY** takes as input X_t and some parameters and returns a sample from Y_t (given X_t).

Function **generate_data** first initializes both X_t and Y_t, then iteratively updates X_t (function **stepX**) and samples Y_t (function **sampleY**) and therefore generates synthetic data.

Finally, we defined ourselves the Feynman-Kac model associated with the bootstrap filter. Its method **M** updates the values of X_t given its precendent value (function **stepX**) for all N particles. Its method **logG** computes the loglikelihood of an observation and involves twice the usage of the sum-log-exp trick. The computation of the loglikehillod involves an infinite sum, obviously not computable, we truncated this sum at the arbitrary value of 10.

Figure 1 presents the data that we generated using this model and on which the rest of the project is centered; it has a length of 1000. The data in the paper is much longer, but we kept it short to reduce computation time. The red lines indicates u_t, the underlying transcription level we wish to recover from the read counts y_t (the observations). There are some extreme values of y_t that were not present in the figure of the paper and are probably due to the tail of the distributions in the emission model. Appendix B presents the value of each of the parameter that we used and a figure with only the transcription level u_t is available in appendix C.

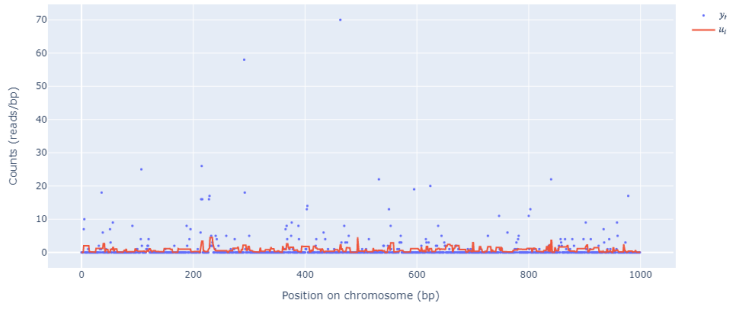

Figure 1: The expression level (u_t) and the observations (y_t) in the generated data

4 Results obtained using the bootstrap filter

After the manual definition of our Feynman-Kac model, we were then able to run bootstrap filters. We ran 10 of them, using 100 particles. Figure 2 presents our results. The red curve is the true u_t, which is available to us because of the synthetic nature of our data. The blue curve is, for each data point, the mean of the 10 boostrap filters; we've also computed the standard deviations. The light blue area gives an idea of the uncertainty: the top of the hull is the mean plus one standard deviation, the bottom is the mean minus one standard deviation (for each data point; this is to say that figure is based on 1000 means and 1000 standard deviations).

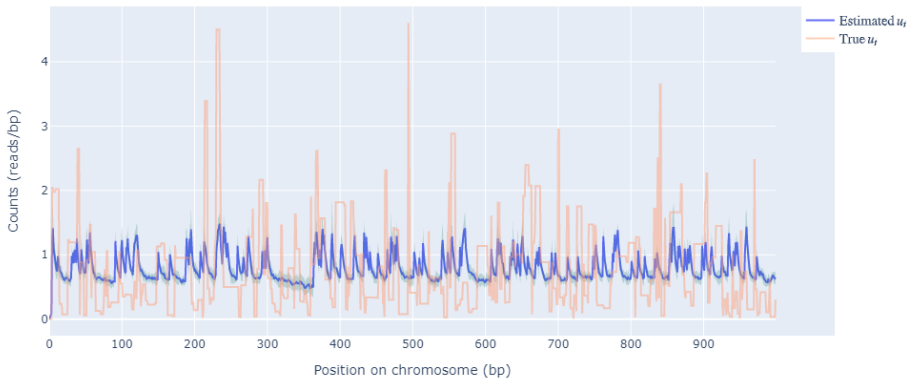

Figure 2: Comparison of the output of the bootstrap filter with the real data

The u_t obtained using the bootstrap filters depicts much less variation than the true one. It does not follow the huge variations of the true u_t. All in all, this result is pretty disappointing.

This weak performance can be due to the nature of the bootstrap filter but perphaps also to the nature of the data, that experiments powerful and short spikes. It is possible that our modification of the original parameter's value, that we chose to do to help the bayesian inference, had a role in this poor performance (see appendix B for more information on this front). The data presented in the paper, as well as data we generated earlier in the project (not shown), were much smoother.

5 Bayesian inference using PMMH

In total, our model involves a staggering **16 parameters**. But we did not try to perform bayesian inference on all of them.

First, the paper states that some parameters of the emission model (κ, θ, κ_s, α_s) can, in real settings, be estimated without PMMH, on some portions of the real data. Therefore, it seemed logical not to estimate them. Also, the parameter b is set, in real settings, to be the maximum value taken by y_t, so it does not need to be estimated either. Also, the paper explains that it is difficult to estimate at the same time the frequences (γ_u, γ_d) and the amplitutdes (λ_u, λ_d) of the drifts. They chose to fix the λ's and only to estimate the γ's and we do the same.

At this point, we are left with 9 parameters. But 7 of them are probabilities, that sum to one in two different groups. Because of that, they cannot take arbitrary values, and their prior is a Dirichlet distribution. This distribution is not implemented yet in the particle package, and we chose not to work on them.

Finally, we used the PMMH algorithm only on two parameters: η and ζ [3]. The parameter η is the probability that u_t jumps from a value of zero to a positive one. The paramater ζ is the rate of two exponentials: u_t is drawn from such exponential: (i) with probability η if $u_{t-1} = 0$, (ii) with probability β if $u_{t-1} > 0$.

We used the same priors than the paper:

$$\eta \sim Beta(shape_1 = 1, shape_2 = 100), \quad \zeta \sim Exponential(rate = 1)$$

Due to lack of time, we were mainly able to run one long PMMH algorithm, with 1000 iterations but only 20 particles (it took about 6 hours to run). Figure 3 presents the MCMC traces of the two parameters. The light red lines represent the true value of each parameter ($\eta = 0.05$ and $\zeta = 1.18$).

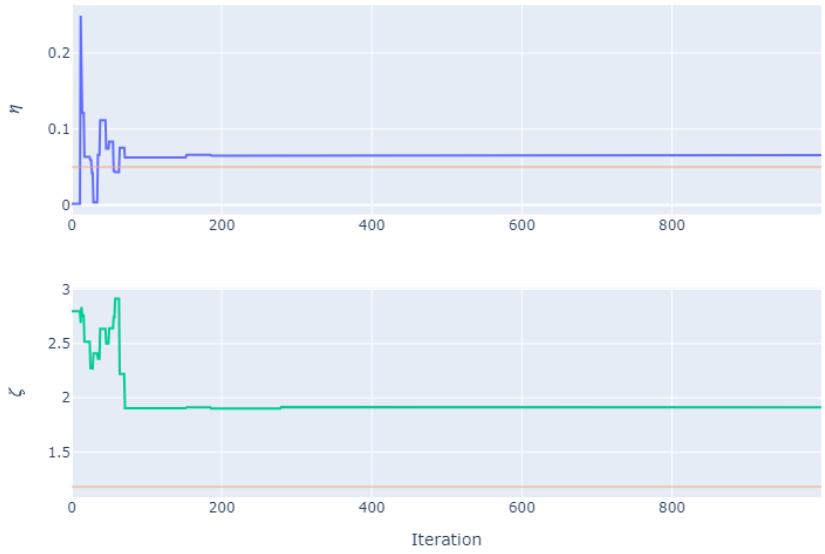

Figure 3: MCMC traces obtained using the PMMH algorithm

The behavior of the traces is pretty odd. During the first few iterations, the value of both parameters does, but then it stabilizes and does not move anymore. On the whole simulation, the mean of η is 0.065, which pretty close to its actual value; the mean of ζ is 1.959, further from its real value. Clearly, this chain is not satisfatory. It should vary much more so as to explore the whole posterior distribution of the two parameters. This would probably require much more particles, at the cost of diminishing the total number of iterations.

We did not have the time to tune the parameters of the PMMH algorithm, so as to obtain good performance. However, in an effort to estimate how inaccurate our results were, we ran 10 PMMH with the same number of particle than during this long simulation (20) but for only 100 iteration (it took about 7 hours to run). This allowed us to access the a posteriori log likelihood for each of them and to compute its variance. The variance of the log likelihood estimation is 25.6; ideally, it should be much smaller than one, so this value shows that the PMMH algorithm is not correctly tuned.

To sum up, we were able to use the PMMH algorithm, but not to calibrate it and obtain good performances. The first thing that would have been necessary it to increase the number of particles (and probably

[3] We realised later that, considering we were not trying to estimate γ_u and γ_d, we could have tried to perform bayesian inference also on λ_u and λ_d.

to decrease the number of iterations so that the computing time would remain acceptable), until the variance of the log likelihood was small enough.

6 Conclusion

Mirauta et al., 2014 presented a novel way to analyze RNA-Seq results in order to reconstruct the transcription result based on read counts. The complicated nature of the state space model they propose allows to account for multiple sources of artifactual variability in the data. They were able to use Particle Gibbs to estimate parameter's values and showed that their method performed well on real data.

Our project involved three steps: (i) generating synthethic data using the model, (ii) implementing the bootstrap filter based on a manually defined Feynman-Kac and (iii) performing bayesian inference using the PMMH algorithm. After some wanderings, we managed to implement data generation and the Feynman-Kac. We were also able to run both the bootstrap filter and the PMMH algorithm, but their performance is deceptive. Our project is incomplete in the sense that we did not have the time to tune up both algorithms so as to obtain good performances.

Finally, the model involves many parameters acting as probabilities, and therefore associated with a Dirichlet prior. An interesting continuation for the project would be to work on those parameters - possibly generating a Dirichlet based on Gamma distributions.

References

Mirauta, B., Nicolas, P., & Richard, H. (2014). Parseq: Reconstruction of microbial transcription landscape from rna-seq read counts using state-space models. *Bioinformatics*, *30*(10), 1409–1416.

A Distributions

This section is largely copied from the section **1.2 Notations** in the supplementary materials of the paper.

- $\mathcal{P}(\lambda)$ stands for the Poisson distribution with mean λ.
- $\Gamma(a, b)$ stands for the Gamma distribution with **shape** κ and **scale** θ. This implies being careful because some packages use a rate instead of a scale.
- $\mathcal{E}(\lambda)$ stands for the exponential distribution with **rate** λ.
- $\mathcal{U}(a...b)$ represents the uniform distribution with minimal value a and maximal value b.
- $\mathcal{NB}(r, p)$ stands for the Negative Binomial distribution characterized by size r and probability p. Its mean is $\frac{rp}{1-p}$.
- δ_a stands for the Dirac distribution at point a.

B Parameter's values

We have worked on synthethic data and thus needed to choose values of the parameters that "made sense", that would generate somewhat realistic data. We began with values we found in the supplementary materials: the mean value of each parameter, estimated for *S. cerevisiae* (table S2, page 17).

Because we performed bayesian inference on η and ζ and used a limited length of data (1000), we wanted our data to involve as much information as possible. So, we changed the value of two parameters, η and β. Indeed, η controls the "jump" of u_t from zero to non-zero values; increasing it led to more "jumps". On another hand, β is the probability of u_t beeing drawn from $\mathcal{E}(\zeta)$ if $u_{t-1} > 0$, so increasing the value of β led to more of those exponentials and it seemed to us that it would had information about the parameter of this exponential.

The disadvantage of this approach is that it led to generating a u_t much less smooth than the one in the paper, due to the decreased value of α and the increased value of β. Earlier data generation (not shown), using the original parameter's values from the paper, allowed the generation of such smoother data. We did not use them because we feared such data would not contain enough information about η and ζ to achieve good bayesian inference.

In the end, the values we used (for both generating the data and for the bootstrap filter) are the following:

Parameter	Value
κ	0.61
θ	1.93
ϵ_b	6.7e-4
ϵ_0	2e-6
α_s	0.53
κ_s	0.31
α	0.72
γ_u	0.011
γ_d	0.013
β_0	6e-4
β	8e-4+0.25
η	0.05
ζ	1.18
λ_u	5.0
λ_d	5.0
b	10

C Another look at the generated data

Figure 4 allows to inspect u_t more closely, since the presence of y_t leads to a change in the scale.

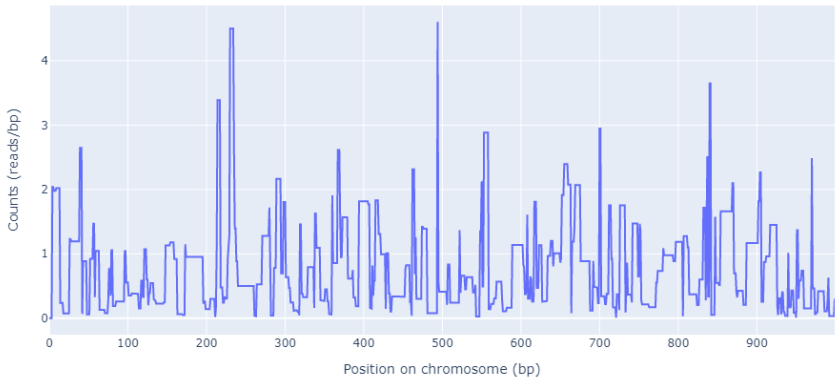

Figure 4: The expression level u_t in generated data

D More on the paper and RNA-Seq

RNA–seq is a laboratory method used to determine the exact sequence (order) of the building blocks that make up all RNA molecules in a cell. RNA-Seq allows researchers to detect both known and novel features in a single essay, enabling the identification of transcript isoforms, gene fusions, single nucleotide variants, and other features without the limitation of prior knowledge. The research paper on which we worked broadly discusses the development of a statistical approach to estimate the local transcription levels and to identify transcript borders.

The research is focused on developing a computational approach for analyzing gene expression data obtained from RNA-Seq experiments. RNA-Seq is a widely used technique for characterizing whole-genome transcriptional profiles, and it involves the high-throughput sequencing of RNAs to generate millions of sequence reads (a read is basically a short sequence, reprensenting a piece of genome). These reads can be aligned to a reference genome and used to estimate gene expression levels and identify transcriptionally active regions. However, there are still limitations to the accuracy and completeness of transcriptome annotations for many organisms, and there is a need for new computational approaches that can help to extract more information from RNA-Seq datasets.

One approach for identifying transcript structures is based on read assembling, which involves aligning reads to the genome and constructing fragments based on paired-end or fragment-length information. These fragments can then be used to infer isoform structure and estimate expression levels. However, this approach can be limited by the sequencing depth and technical biases, and it may not be able to identify overlapping transcripts or incomplete termination.

The paper discusses a probabilistic model of RNA-Seq count data that takes into account transcription level variation as well as protocol-induced biases. This model includes a state-space model to describe transcription level variations and a new emission model that captures read count variance and autocorrelation. The approach is intended to improve the accuracy of transcription level estimation and the identification of transcript borders.

This approach is built on previous work on segmenting genomic data into regions of piecewise constant expression, which has been motivated by the analysis of comparative genomic hybridization and transcription tiling array data. One of the challenges in this context is identifying the correct number of breakpoints and assessing uncertainty in their positions. To address these issues, the authors developed a probabilistic model based on hidden Markov models with continuous state space (also known as state-space models) that can be used to reconstruct local transcription levels, call transcribed regions, and identify coverage breakpoints. This model incorporates the full dynamics of the transcription signal, which may allow for a more accurate and comprehensive analysis of transcriptional data.