5

# Bayesian Inference of Gene Expression

Víctor Jiménez-Jiménez[1,2*] • Carlos Martí-Gómez[1,3*] • Miguel Ángel del Pozo[2] • Enrique Lara-Pezzi[3] • Fátima Sánchez-Cabo[1]

[1]Bioinformatics Unit, Centro Nacional de Investigaciones Cardiovasculares (CNIC), Madrid, Spain; [2]Mechanoadaptation & Caveolae Biology Laboratory, Centro Nacional de Investigaciones Cardiovasculares (CNIC), Madrid, Spain; [3]Molecular Regulation of Heart Failure Lab, Centro Nacional de Investigaciones Cardiovasculares (CNIC), Madrid, Spain

**Author for Correspondence:** Fátima Sánchez-Cabo, Bioinformatics Unit, Centro Nacional de Investigaciones Cardiovasculares (CNIC), Madrid, Spain. Email: fscabo@cnic.es

Doi: https://doi.org/10.36255/exonpublications.bioinformatics.2021.ch5

**Abstract:** Omics techniques have changed the way we depict the molecular features of a cell. The integrative and quantitative analysis of omics data raises unprecedented expectations for understanding biological systems on a global scale. However, its inherently noisy nature, together with limited knowledge of potential sources of variation impacting health and disease, require the use of proper mathematical and computational methods for its analysis and integration. Bayesian inference of probabilistic models allows propagation of the uncertainty from the experimental data to our beliefs of the model parameters, allowing us to appropriately answer complex biological questions. In this chapter, we build probabilistic models of gene expression from RNA-seq data and make inference about their parameters using Bayesian methods. We present models of increasing complexity, from the quantification of a single gene expression to differential gene expression for a whole transcriptome, comparing them to the available tools for analysis of gene expression data. We provide *Stan* scripts that introduce the reader into the implementation of Bayesian statistics for omics data. The rationale that

* These authors have contributed equally.

In: *Bioinformatics*. Nakaya HI (Editor). Exon Publications, Brisbane, Australia.
ISBN: 978-0-6450017-1-6; Doi: https://doi.org/10.36255/exonpublications.bioinformatics.2021

we apply for transcriptomics data may be easily extended to model the particularities of other omics data and to integrate the different regulatory layers.

**Keywords:** Bayesian inference; gene expression; omics; probabilistic models; stan

## INTRODUCTION

Omics techniques have allowed us to jump from the quantification of single or few molecules to the study of thousands of features at the same time, first through cDNA microarrays (1), which were rapidly substituted by Next Generation Sequencing (2) and extended to other molecular features, for example, proteins, metabolites, DNA methylation markers, and DNA variants, to name a few. However, it has become clear that omics techniques are prone to systematic and random error (3), and that appropriately modeling these errors is critical for success.

A probabilistic model allows the calculation of the probability of our data given a set of parameter values. For example, the probability of obtaining head or tails when tossing a coin; if the coin is fair, the parameter that characterizes this model, which is the probability of obtaining heads $p$, is 0.5. However, when studying a certain phenomenon, we do not know the actual value of $p$ that generated the data. Instead, we can use the data to guess the parameter values that best explain the obtained data in a process called inference. Thus, if we want to know whether the coin is fair, we can make a number of tosses and count the number of tails to estimate the actual value of $p$.

As experimental data is limited, we can only toss the coin a finite number of times, and as it is subject to random variation, we cannot know the exact underlying parameter value, but have an idea of the range of parameter values in which the real value may be. Characterization of this variability and how it is related to the data is essential to draw conclusions from experiments. Bayesian inference approaches this problem by calculating the probability of the parameter given by the data (posterior probability). Thus, rather than providing a fixed estimate of the true value of the parameters that generated the data, we obtain a probability distribution of those parameters characterizing the uncertainty in our estimations. Using this posterior distribution, we can calculate the probability of our hypothesis; for example, what is the probability that the difference in the expression of a single gene is greater than a certain threshold?

Despite the conceptual suitability of Bayesian inference, even relatively simple models often result in complex posterior distributions, lacking an analytical solution for calculating the probabilities of our hypotheses. Monte Carlo methods allow computing these probabilities through sampling from the distribution of interest: one just needs to count the number of samples matching the condition of interest; for example, we may sample from the posterior distribution of gene expression values and count how many of those samples are beyond a certain threshold. Markov Chain Monte Carlo (MCMC) algorithms have enabled sampling of complex posterior distributions, allowing the development of probabilistic programming frameworks such as OpenBUGS (4), JAGS (5) or *Stan* (6). The latter

implements a version of the Hamiltonian Monte Carlo (HMC) method (7), the No-U Turn Sampler (NUTS), which allows efficient sampling from high dimensional posterior distributions for approximate Bayesian inference on models with a large number of parameters. The NUTS algorithm is also available in other software packages than Stan (6), like *pymc3* (8) or *tf-probability* (9). Stan has a large community behind it, along with detailed documentation and examples, which has been key to its success and development.

In this chapter, we review the building blocks of probabilistic models and Bayesian inference. We explain how to build a simple probabilistic model for the inference of the expression level of a single gene in a single sample from scratch, and examples of how to use well-known and characterized probability distributions to model our data of interest. We build models of increasing complexity by simultaneously inferring the expression of every gene in the transcriptome and how to deal with some of the particularities of RNA-seq data, like ambiguous assignment of sequenced fragments, to more than a single gene. Finally, we present in detail the model proposed to estimate differences in gene expression across different samples and conditions, considering variability within groups. For this application, we provide *Stan* code for its implementation.

## BUILDING BLOCKS OF BAYESIAN INFERENCE

The aim of Bayesian inference is to update our knowledge about the parameters of a random process using the experimental information available. Let us think about the parameter θ, that is the log-2 normalized expression of gene *Pparγ*. Let us say, based on some previous experiments or on theoretical considerations, that we know that its expression should be around 1 in the log2 scale in steady-state conditions for a given sample and library depth. We perform an RNA-Seq experiment and measure the expression of *Pparγ* in three biological specimens at steady-state. Using this data, we want to update our knowledge about the expression of *Pparγ*. For that, we need to define the following:

1. ***Prior:*** $P(\theta = t)$ is the probability *a priori* that the unobserved parameter of interest $\theta$ takes a value $t$, based only on our knowledge before performing the experiment. For the example above, the prior probability distribution should be centered around 1. Bayesian statistics allows us to tune this expectation: If we are confident that the expression of *Pparγ* is 1, we will suggest as prior a distribution with almost no dispersion around the expected value (Figure 1A). That would be a *strong prior*. On the other hand, if our experiments were unclear, we would propose a non-informative or *weak prior* with a lot of dispersion around the expected value (Figure 1B).
2. **Likelihood:** $P(y \mid \theta = t)$, how plausible it is to have observed the data $y$ if we are in certain scenario, that is, if $\theta$ was to take a value of $t$. In our example, we know that the measurements of the gene expression using RNA-Seq after normalization and in the log2 scale follow approximately a normal distribution, centered around its expected actual expression 1:

$$P(y \mid \theta) \sim N(\theta, \sigma)$$

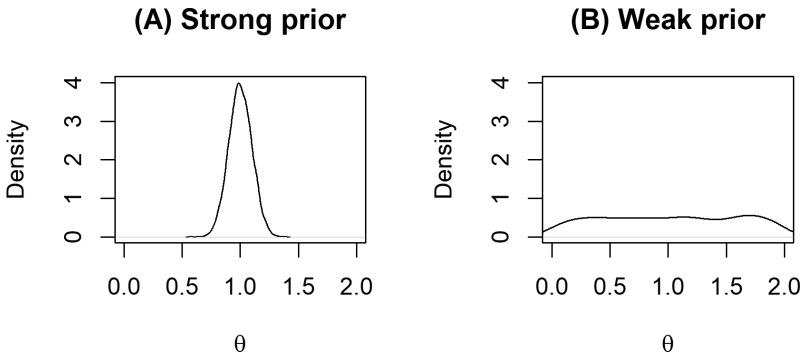



**Figure 1. Prior distributions for the gene expression level of *Pparg*. A.** In a strong prior, most of the probability density is concentrated around its center of mass. **B.** In a weak prior, the probability density is more homogeneously distributed across the whole parameter space.

3. **Posterior:** $P(\theta = t|y)$ is the probability that the parameter $\theta$ takes a value $t$, knowing now that we have observed the data $y$. Given the data and the *a priori* distribution of the parameter of interest, we want to infer the probability of the different possible values of the parameters of interest. Using the Bayes Theorem:

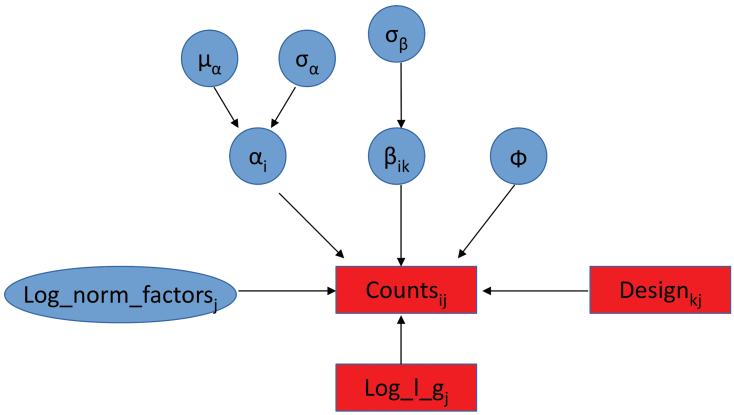
$$P(\theta \mid y) = \frac{p(y \mid \theta)\, p(\theta)}{p(y)} \propto P(y \mid \theta) \cdot P(\theta), \text{since } P(y) \text{ is constant}$$

In general, to report our findings we will use some summarization of the posterior distribution such as the posterior mean, median or mode. Another important concept that emerges from the posterior distribution that we will use through the chapter is the credibility interval: the interval within which the unobserved parameter lies with a given probability. There are many credibility intervals for a parameter; in general, we will try to choose one that contains the most likely values for the parameter of interest.

4. In complex models with many parameters, the relationships between the data and the parameters, and among parameters, can be represented in Directed Acyclic Graphs (DAGs), which are useful representations of complex probabilistic models and show their modular structure (10). Figure 2 shows the DAG for the Bayesian inference problem of trying to infer differences in gene expression across two conditions. In particular, DAGs help us to decompose the joint priors and posterior distributions through the chain rule, paying attention only to the *parents* of each parameter:

$$P(\theta_1, \ldots, \theta_n) = \prod_{i=1}^{n} P\left(\theta_i \mid parents(\theta_i)\right)$$

For *parentless* parameters an *a priori* distribution needs to be set.




**MODEL(DAG)**

$$Counts_{ij} \sim NB(\exp(\alpha_{ij} + \beta_{ik}\, Design_{kj} + Log\_norm\_factors_j + log\_eff\_length), \Phi)$$

$$\alpha_{ij} \sim Normal(\alpha_\mu, \alpha_\sigma)$$

$$\beta_{ik} \sim N(0, \sigma_\beta)$$

**PRIOR DISTRIBUTIONS**

$$\mu_\alpha \sim Normal(n/T, 1.17)$$

$$\sigma_\alpha \sim Normal(0, 2)$$

$$\sigma_\beta \sim Normal(0, 1)$$

$$\Phi \sim Uniform(0, 2000000000)$$

$$Log\_norm\_factors_j \sim Normal(0, 0.05)$$

**Figure 2. Directed Acyclic Graph (DAG) and probabilistic model for the identification of differentially expressed genes.** The DAG represents all the parameters that describe the model and the dependence relationships among them. Parameters and hyperparameters are depicted in blue whereas the data are shown in red boxes. We are modeling expression counts with an NB model. The logarithm of the expected counts for each gene depends on an average gene expression value α, the expression change β among the different conditions defined in the design matrix, the normalization factors of each sample and on each gene effective length. The variability of the NB model described by the parameter Φ.

## PROBABILISTIC MODELS FOR SINGLE GENE EXPRESSION

From all techniques used to quantify gene expression, RNA-Seq is the most widely used nowadays. To propose a proper probabilistic model, it is crucial to understand in depth the process that generates the data. In RNA-Seq, mRNA from the cell culture, tissue or sample of interest is reverse-transcribed into cDNA, which is then cut into smaller fragments. Some of these fragments are then sequenced using smaller overlapping reads that cover the transcriptome. Depending on the sequencing conditions, that is, number of cycles and paired-end or single-end

sequencing, we obtain a certain number of bases covered from one or both ends of each fragment. Assuming that there are no unknown positions in the transcriptome sequence and that the reads are long enough to unambiguously map them into their correct position in the transcriptome, the outcome of the sequencing experiment will be the number of reads covering each position of the transcriptome. In this section, we propose different probability models and state the explicit assumptions made about parameters and data to infer the expression ($\theta_g$) of a single gene in a single sample from an RNA-seq experiment.

## Binomial distribution

Each cell simultaneously expresses thousands of genes. The expression of a gene may be understood as the proportion of the fragments that may arise from this gene out of the total amount of sequenced fragments. According to this idea, each fragment that we obtain is equivalent to throwing a coin, and the probability of obtaining heads is equivalent to the probability of that read mapping into a given gene, for example, the expression of that gene ($\theta$), which naturally follows a Bernoulli distribution with parameter $\theta$.

$$p(F = 1|\theta) = Bernoulli(F = 1|\theta) = \theta$$

If the initial amount of RNA is so high that sampling with replacement can be assumed so that fragments are sampled independently from each other, we can calculate the probability of obtaining $k$ fragments from our gene of interest from a total of $n$ sequenced fragments in the experiment. This can be done by simply dividing the total number of individual events that may lead to obtain $k$ fragments by the total number of possible outcomes. The resulting distribution is known as the binomial distribution:

$$p(k \mid n,\theta) = Binomical(k \mid n,\theta) = \binom{k}{n}\theta^k(1-\theta)^{n-k}$$

To perform Bayesian inference under this simple model, we need to specify a prior distribution for the only parameter $\theta$. We know that, in a complex sample, several thousand genes may be simultaneously expressed, such that the probability of sampling a fragment from a particular gene is expected to be very low. It is virtually impossible that 50% of the sequenced fragments arise from a single gene. A more reasonable assumption may be that every gene is expressed at similar levels, such that the expected $\theta$ will depend on the number of expressed genes in the sample $N$. Thus, 1/N may be the expected value of our desired prior distribution. Next, we need to think about the uncertainty that we have around this value: how sure are we that every gene is expressed exactly the same? We can imagine that some genes may be expressed at higher levels, by 1 or 2 orders of magnitude, but hardly more. Thus, we can search for a distribution that can represent this prior belief: the beta distribution.

$$p(\theta) = Beta(\alpha, \beta)$$

Assuming about $N = 10^4$ expressed genes in a given biological condition, for $\alpha = 0.01$ and $\beta = 99.99$

$$E(\theta \mid \alpha = 0.01,\ \beta = 99.99) = \frac{\alpha}{\alpha+\beta} = \frac{1}{N} = 10^{-4}$$

$$P(\theta > 0.01 | \alpha = 0.01, \beta = 99.99) = 0.002$$

Here, as the beta and the binomial are conjugate distributions, the posterior distribution of $p$ is also a beta distribution. From this known posterior distribution, we may easily calculate the expected value, uncertainty, and probabilities for any hypothesis.

$$p(\theta \mid k,n) = \frac{p(\theta \mid n,p)p(p)}{p(k)} = Beta(\alpha + k,\ \beta + n - k) = Beta(0.01 + k,\ 99.99 + n - k)$$

For example, if we sequence a total of 20 million fragments for a given sample and find a total of 500 fragments from a particular gene, we can compute the posterior expectation $E(\theta|k = 500, n = 2 \times 10^7) = 2.5 * 10^{-5}$ and 95% credibility intervals ($[2.29,2.72] \times 10^{-5}$).

## Poisson distribution

Under large $n$ and low $\theta$ scenarios, as we generally have by sequencing millions of cDNA fragments, the binomial distribution can be approximated by the Poisson distribution with parameter $\lambda = n\theta$ according to the Poisson limit theorem, also known as law of rare events:

$$\lim_{n\to\infty} Binomial(k \mid n,\theta) \simeq Poisson(k \mid \lambda = n\theta)$$

Moreover, the number of fragments that can be generated from a gene does not only depend on its expression, but also on its length and the fragment size. A 1 kb long gene may generate up to 800 different 200 bp fragments, while a 500 bp gene will only produce 300 different fragments. We may define effective transcript length $l_g$ as the number of different fragments that can be generated from gene $g$. Thus, the expectation of the resulting distribution is not the gene expression $\lambda_g$ but its value scaled by $l_g$: $(\lambda_g^* = l_g\lambda_g)$:

$$p(k_g \mid \lambda_g^*) = Poisson(k_g \mid \lambda_g^*) = Poisson(k_g \mid l_g\lambda_g)$$

For doing Bayesian inference of $\lambda_g$, as previously done for $\theta$, we need to specify a prior distribution describing our prior beliefs on the potential values that $\lambda_g$ may take. Again, if we vaguely expect every gene to be expressed at similar levels, and hence, every position in the transcriptome to be able to give raise to a fragment with the same probability, we expect an average of $n/T$ expression of each gene, being $T$ the total transcriptome effective length.

$$E(\lambda_g) = \frac{n}{T} = \frac{n}{\sum_i^N l_i}$$

$$E\left(k_g\right) = \frac{l_g n}{\sum_i^N l_i}$$

Assuming a total of $n = 2 \times 10^7$ sequenced fragments from $N = 10^4$ genes, an effective transcript length of 200 bp $l_g = 700$ and a transcriptome length of $T = 10^7$, the expected number of fragments from gene $g$ is $E(k_g) = 1{,}400$. If we assume again that there is a 0.2% probability of obtaining over 1% of total expression from gene $g$, and we assume that the total expression is approximately Nn/T, then we need to specify a prior distribution with only 0.2% of being higher than $l_g 0.01\frac{Nn}{T} = 400$. We can choose the conjugate distribution of the Poisson (Gamma distribution) trying to match the 2 conditions.

$$p\left(\lambda_g^* \mid \alpha, \beta\right) = Gamma\left(\lambda_g^* \mid \alpha, \beta\right) = Gamma\left(\lambda_g^* \alpha = 0.011,\ \beta = 2.8 \times 10^{-5}\right)$$

As the Gamma distribution is the conjugate of the Poisson, the resulting posterior distribution is also a Gamma:

$$p\left(\lambda_g^* \mid k, \alpha, \beta\right) = \frac{p\left(k \mid \lambda_g^*\right) p\left(\lambda_g^* \mid \alpha, \beta\right)}{p(k)} = Gamma\left(\alpha + k,\ \beta + 1\right)$$

Thus, under the previous conditions with $k_g = 500$, the posterior expectation of the expected number of fragments is $E(\lambda^* | k, \alpha, \beta) \simeq 500$, with a 95% credible interval of [457.13,544.77]. If we divide this interval by $n = 2 \times 10^7$, we end up with an interval of $[2.28,2.72] \times 10^{-5}$, which is almost identical to the estimated under the Binomial model as anticipated. We derive the actual gene expression by dividing by the effective transcript length $l_g$:

$$\lambda_g = \frac{\lambda^*}{l_g}$$

$$E(\lambda_g | l_g = 200,\ k_g = 500) = 2.5$$

## BAYESIAN INFERENCE OF WHOLE TRANSCRIPTOME EXPRESSION

Until now, we have focused on the study of the expression of a single gene. However, with transcriptomics we aim to study the expression of every gene in the genome at the same time. Hence, our probabilistic model should include the expression of every gene as a parameter of interest. In this section, we focus on modeling the expression of multiple genes. We will generalize our previous models for inference of multiple genes and explain how to handle sequence fragments

that are ambiguously assigned to different genes. We will also explain how we can use hierarchical models for shrinking extreme and unlikely values towards average gene behavior. Finally, we will contextualize how *k-mer* mapping methods can be easily accommodated into our Bayesian Framework.

## From the Binomial to the Multinomial and Poisson distributions

In whole transcriptome analysis, the outcome of interest is no longer whether a sequenced fragment belongs to a given gene or not, which is accurately described by the Bernoulli distribution, but from which gene the sequenced fragment originated. This is equivalent to throwing a die instead of a coin: for each toss there are $N$ possible outcomes, being $N$ the number of sides of the die, or, in general, the number of possible and mutually exclusive outcomes. If we consider a sequenced fragment, the outcome may be from which of the N genes it was generated. This phenomenon naturally follows a *categorical distribution*. The *categorical distribution* is a generalization of the *Bernoulli* for more than 2 possible and mutually exclusive categorical outcomes and has $N$ parameters $\vec{\theta} = (\theta_1,...,\theta_N)$. In this case, $\theta_g$ corresponds to the probability of gene $g$ to generate a fragment, that is, its expression level. As $\sum\theta_i = 1$, there are effectively $N-1$ free parameters.

$$p\left(F = i \mid \vec{\theta}\right) = Categorical\left(F = i \mid \vec{\theta}\right) = \theta_i$$

Where $i = 1,\ldots,N$. Hence, the number of reads mapping into every gene $\vec{k}$ will follow a *multinomial distribution*, with parameters $\vec{\theta}$ as the actual expression of each gene:

$$p\left(\vec{k} \mid \vec{\theta}\right) = Multinomial\left(\vec{k} \mid \vec{\theta}\right)$$

The *multinomial* also has a conjugate distribution: the *Dirichlet*, which is itself a generalization of the Beta for $N$ dimensions. We can conveniently use it to formalize our prior knowledge on gene expression values and to obtain a closed form for the posterior distribution. Let's assume for simplicity that we sequenced a total of n = 500 fragments from $N = 5$ genes in our experiment. We do not know the expression of each gene or whether one is expected to be expressed at higher levels. However, we may vaguely expect relatively similar gene expression levels, that is, the probability of generating a fragment from each gene is the same. Since we expect every gene to generate about 20% of the fragments, with a 5% probability that over 40% of fragments are originated from one of them, we set a Dirichlet prior with $\alpha_i = \alpha = 2.5$:

$$p\left(\vec{\theta}\right) = Dirichlet\left(\vec{\alpha}\right) = Dirichlet(2.5)$$

Using this prior distribution, the resulting posterior is also a Dirichlet distribution with the following parameters:

$$p\left(\vec{\theta} \mid \vec{k}\right) = \frac{p\left(\vec{k} \mid \vec{\theta}\right)p\left(\theta\right)}{p\left(\vec{k}\right)} = Dirichlet\left(\vec{\alpha} + \vec{k}\right) = Dirichlet\left(2.5 + \vec{k}\right)$$

If out of the 500 sequenced fragments, we observed $\vec{k}$ = (150,25,40,60,225) originated from each gene, the posterior expectation of the expression of gene 1 would be $E[\theta_1]$ = 0.30, with a 95% credible interval of [0.26, 0.34]; while for gene 5 it would be $E[\theta_5]$ = 0.44, with a 95% credible interval of [0.40, 0.49].

BitSeq models the gene expression as a Dirichlet distribution and takes each sequenced fragment as an independent outcome that depends on the expression of the gene from which it was originated and taking into account a certain probability of mapping error and a known fragment size distribution. (11)

As in the single gene case, the Multinomial distribution may also be approximated by a multidimensional Poisson distribution when $\theta_i$ is very small and the number of fragments generated by each gene becomes nearly independent, as in $N$ independent Poisson distributions. In the Poisson approximation, for every gene there is a parameter that describes the mean and variance of the distribution of the number of counts observed for that gene. This parameter $\lambda^*_g$ is related to the effective length and the expression levels of that gene $\lambda^*_g = l_g\lambda_g$, as in the single gene case.

## Handling multi-mapping reads

Until now, we have assumed that genes are completely unrelated, and that cDNA fragments can be unambiguously assigned to each of them. However, in a real genome, we find transcript isoforms generated by alternative splicing of the same pre-mRNA, gene duplicates retaining a large degree of sequence similarity or highly conserved domain sequences. Thus, these transcripts may produce identical fragment sequences. Additionally, if we sequence few bases of the fragment ends, we may even find sequences that match by chance somewhere else in the genome and hinder the identification of the origin of that fragment. While these multimapping fragments may be discarded from the analysis, they carry additional information that may contribute to a better estimation of gene expression levels.

To tackle this issue, we can model the number of fragments at each transcriptome position instead of the counts per transcript. To do so, we define a $T$ sized vector $\vec{k}$ as the number of times a fragment from a given position in the transcriptome was sequenced, together with a matrix $M$ of size $T \times N$ that relates each position in the transcriptome with the gene from which it may be generated.

$$M_{ij} = \begin{cases} 1 & \text{if position } i \text{ belongs to gene } j \\ 0 & \text{otherwise} \end{cases} \qquad \text{(eq. 1)}$$

We can also define an $N$-sized vector $\vec{\mu}$ representing the gene expression across every gene in the transcriptome and a $T$-sized vector $\vec{\lambda}$ representing the expected number of fragments sequenced from a position in the transcriptome, such that $\vec{\lambda} = M\vec{\mu}$. Under this assumption, the expected number of fragments at each position is a sum of the expression levels of every gene possibly generating that fragment.

$$p\left(\vec{k} \mid \vec{\mu}, \mathbf{M}\right) = Poisson\left(\vec{k} \mid \mathbf{M}\vec{\mu}\right)$$

So far, the Boolean matrix M has only a 1 per row (per fragment) as fragments are unambiguously assigned to a certain transcript. If, for example, a fragment can map to two different genes, the corresponding row for that fragment must have two ones, one for each column corresponding to the two genes where it maps. In this situation, the expected number of reads per gene would result from adding the expression of every transcript from which it may be generated. As long as there are no duplicated rows in *M*, that is, as long as there are no transcripts that produce the exact same fragments thus rendering them completely indistinguishable, we should be able to infer the expression of each transcript from the data even if they share a large number of potential fragments. This problem has been often tackled using optimization algorithms, for example, Expectation-Maximization (EM) (12–14). In particular, Bray *et al.* (13) makes use of bootstrapping by subsampling a number of reads, providing an idea about the sensitivity of the gene expression estimates to a specific form of variation in the input data. However, it still does not provide a main estimate of the uncertainty over the parameters given the observed number of counts for each fragment in our samples. Full Bayesian inference of the model parameters, in contrast, would allow direct estimation of such uncertainty and show the degree of knowledge that we have about the gene expression levels after observing the data.

Full Bayesian inference of the model parameters would require modeling the number of fragments observed from each potential position in the transcriptome, which may be too large to handle efficiently from a computational point of view. We may simplify the model by defining an intermediate entity between individual fragments and genes: a gene group. A gene group includes groups of genes that share some fragments. Thus, instead of counting the number of sequenced fragments individually, we may count the number of sequence fragments in each gene group as read out. Each row of *M* now links each gene group with the transcripts that are included in it. The size of $\vec{k}$ is now the number of gene groups, which is at least equal to the number of transcripts, as a gene group should at least include a single transcript.

As we sum fragments over gene groups, we may encounter different numbers of fragments from each group depending on the number of transcriptomic positions that they include. Thus, we need to define an effective gene group length to scale the expected number of fragments from the actual transcript expression values. Thus, we additionally need a vector $\vec{l}$ with the effective length, such that, for each gene group *i*:

$$p\left(k_i \mid \vec{\mu}, M_i, l_i\right) = Poisson\left(k_i \mid l_i M_i \vec{\mu}\right)$$

## *k-mer* counting methods

This model may be further simplified if, instead of counting full length fragments, one counts short *k-mer* sequences in the sequencing data. Counting *k-mers* can be done very efficiently thanks to recently developed algorithms, and computationally expensive mapping steps may be avoided. Now, we know which *k-mers* may be generated by each transcript in the transcriptome, so we can again build a matrix *M* relating *k-mers* and transcripts and perform the same inferential process.

Methods like *kallisto* (13) and *salmon* (14) pioneered the use of *k-mers* for estimating gene expression with little loss in accuracy of the quantification, but a huge leap in computational cost.

## Hierarchical model for shrinkage of gene expression estimates

As previously discussed, although not every gene is necessarily expressed at the same level, we expect them not to actually be extremely different but drawn from a common distribution. Thus, we may model unobserved gene expression values $\vec{\mu}$ as following a particular distribution which is characterized by other parameters that are usually referred to as hyperparameters. For instance, we can assume that $\vec{\mu}$ are drawn from a normal distribution with a certain mean and standard deviation, which would be the hyperparameters in this expanded model. Thus, information from individual genes may help identify the global gene expression distribution, and, at the same time, the global information is used as prior for better estimation of the expression of the individual genes. Under this model, unusual gene expression values will be shrunk to better fit the global distribution and correct potential errors.

As the expression of a gene is bounded to be positive $\mu_g > 0$, we may take the log transformation and assume that they are drawn from a common normal distribution, characterized by a mean $\mu_0$ and standard deviation $\sigma_\mu$:

$$p(\mu_g|\mu_0, \sigma_\mu) = logNormal(\mu_g|\mu_0, \sigma_\mu) \qquad \text{(eq. 2)}$$

Now $\mu_0$ represents the average expression level across the whole transcriptome in a given sample. $\sigma_\mu$, on the other hand, informs us about the variability in the expression of different genes, for example, how likely it is to find a gene that is expressed twice the average gene expression.

Under this whole-transcriptome model, we have a much larger number of parameters, including the expression of every gene in the dataset, as well as the mean and standard deviation giving rise to such distribution and over-dispersion parameter ($\Theta = \vec{\mu}, \mu_0, \sigma_\mu, \tau$). As always, to completely specify our Bayesian model, we need to specify prior distributions for the remaining parameters $\mu_0$, $\sigma_\mu$, $\tau$.

To specify a prior on the average gene expression $\mu_0$, we may follow the same reasoning as before. If we expect genes to be expressed at similar levels, then the average of those genes may be close to the average number of expected fragments per position in the transcriptome $\frac{n}{T}$, with 95% probability of being within 1 orders of magnitude. As $\mu_0$ represents the average in the log scale, the following prior sets 95% probability within 0.1 and 10 times $\frac{n}{T}$.

$$p(\mu_0) = Normal\left(\mu_0 | \log\left(\frac{n}{T}\right), 1.17\right)$$

Regarding variation in expression across genes, we may again expect that a single gene would very rarely represent a large proportion of the total gene expression. As before, we can formalize this belief by setting a 1% probability of reaching a standard deviation allowing more than 100 times the average expression for 1% of the genes ($p(\sigma_\mu > 2.35) = 0.01$). We can reach this condition with the following prior distribution:

$$p(\sigma_\mu) = Exponential(1.96)$$

If we use the same exponential prior for $\tau$ as before, we can derive the full posterior distribution and use MCMC methods to sample from it and estimate the marginal distribution of the parameters, including average expression and standard deviation, but more importantly, the expression of each gene in the transcriptome $\vec{\mu}$.

$$p\left(\bar{\mu}, \mu_0, \sigma_\mu \mid \vec{k}\right) = \frac{p\left(\mu_0\right) p\left(\sigma_\mu\right) p\left(\vec{k} \mid \vec{\mu}, M, \tau\right) p\left(\vec{\mu} \mid \mu_0, \sigma_\mu\right)}{p\left(\vec{k}\right)}$$

## MODELING GENE EXPRESSION ACROSS SAMPLES

Up to now, we have dealt with cases where we just have one measurement for each gene, with no biological replicates. Let's now consider the case when we have replicates and, therefore, variance in gene expression values for the same gene across samples must also be modeled.

### Negative binomial distribution: modeling biological variability between replicates

So far, we have focused on the inference of gene expression values across the whole transcriptome in a single sample. However, we often want to infer the average expression in a specific group of samples considering the genetic, environmental, or technical variations among them. Thus, rather than observing $k_g$ fragments from a given gene $g$, we now define as outcome a variable $\vec{k_g}$, representing the total number of fragments observed across the $S$ samples.

Formally, assuming a different expected number of counts per sample for a gene would mean that there is not a common $\lambda$ for each fragment but that $\lambda$ can vary across samples, generating a different average number of fragments for each of them. Practically, this would result in an increased variability among the counts obtained for each gene, which according to our previously assumed Poisson-distributed model, is expected to be equal to the mean. When modeling this higher variability, we can assume that whatever the source of variability in $\lambda$ for fragments generated by the same gene, it may follow a Gamma distribution. Why a Gamma distributed $\lambda$? Mainly because it is mathematically convenient. We can integrate over all possible values of $\lambda$ and calculate the probability of obtaining certain number of fragments directly depending on the parameters of the Gamma distribution, which results on a Negative Binomial (NB) distribution. In this way, we transition from the Poisson model to the Poisson-Gamma model, more commonly referred to as Negative Binomial.

$$p(k \mid \Theta) = \int_0^\infty p(k \mid \lambda)\, p\left(\lambda \mid \phi, \frac{\phi}{\mu}\right) d\lambda = NB(k \mid \mu, \phi)$$

We generally parametrize the NB as a function of its mean $\mu$ and over-dispersion parameter $\phi$ or its inverse $\alpha = \frac{1}{\phi}$, which represents the extra dispersion over the

Poisson distribution introduced by the variability on the underlying $\lambda$ across fragments. Thus, when $\phi \to \infty$, it assumes no over-dispersion over the Poisson distribution with $\lambda$ as mean and variance.

$$Var(k \mid \mu, \phi) = \mu + \frac{\mu^2}{\phi} = \mu + \alpha\mu^2 \qquad \text{(eq. 3)}$$

If we want to use this model now to infer the average expression of a given gene in a group of samples from the observed $\vec{k}$, we first need to specify a prior distribution for the underlying parameters $\Theta = (\mu, \alpha)$. We may use the same hierarchical distribution for $\mu_g$ as before. As for $\phi$, we may assume that the increased variation is unlikely to increase the variance as much as twice, when compared with the Poisson distribution. Using equation (eq. 3) we can see that this happens when $\phi = \mu$. Thus, we need a prior distribution that allocates little mass probability beyond this value, $P(\phi < \mu) = 0.01$. To do so, we may define a new parameter $\tau = \frac{\phi}{\mu}$ and set a prior on $\tau$ such that the probability of it being higher than 1 (when $\phi < \mu$) is 0.01. We may use an exponential distribution, for which we can easily derive the parameter value that meets this condition.

$$p(\tau) = Exponmential(\tau|4.61)$$

$$p(\tau > 1) = 0.01$$

As there are usually few samples to estimate the variability for each gene independently, we may assume that $\tau$ is common for every gene and pool information across genes to infer this parameter. $k$ is expanded to be a matrix, with a column for each sample in the dataset. The probability of observing the whole matrix of counts is then given by:

$$p(k \mid \vec{\mu}, \tau, M) = NB(k \mid M\vec{\mu}, \tau M\vec{\mu})$$

The NB distribution for the data has been widely applied to model gene expression variability across different samples in popular tools such as edgeR (15) or DESeq2 (16).

## Modeling uneven coverage across samples: normalization factors

In addition to the variability within biological replicates, we have an additional issue: samples are sequenced at relatively different depths. In other words, the total number of sequenced fragments may change across samples and thus the expected number of fragments from each position or gene across the whole transcriptome changes. To tackle this issue, we define a new set of S – 1 parameters, corresponding to the log transformation of the normalizing factors $\log(\vec{f})$.

By adding the corresponding factor $\log(f_s)$ to the log-expression of each sample $s$, we are effectively scaling the expression of the remaining samples to have the same average across the whole transcriptome.

$$p(\vec{k_s}, \mid \vec{\mu}, \tau, M, f_s) = NB(\vec{k_s} \mid f_s M\vec{\mu}, \tau f_s M\vec{\mu})$$

As we expect samples to be relatively equilibrated in terms of total sequencing depths, we set a strongly informative prior on $\log(f_s)$ around 0:

$$p(\log(f_s)) = Normal(\tau|0,0.05)$$

Considering these newly introduced factors arisen by modeling several samples simultaneously, we can expand our previous posterior distribution for full inference of model parameters:

$$p\left(\vec{\mu}, \mu_0, \sigma_\mu, \tau, \vec{f} \mid k\right) = \frac{p(\mu_0)\,p(\sigma_\mu)\,p\left(\vec{f}\right)p\left(k \mid \vec{\mu}, M, \tau, \vec{f}\right)p\left(\vec{\mu} \mid \mu_0, \sigma_\mu\right)}{p\left(\vec{k}\right)}$$

## BAYESIAN DIFFERENTIAL GENE EXPRESSION: A *STAN* CASE STUDY

Up to now we have introduced relatively simple statistical models of gene expression for inferring the mean expression of each gene, first independently of the rest of the genes and afterwards from a transcriptome-wide perspective by pooling information from the rest of the genes. However, our final aim is rarely obtaining gene expression estimates for a single sample, but to compare different samples among them and draw conclusions from the comparison of gene expression between different experimental conditions. This type of question is often referred to in the literature as differential expression (15, 17).

We know by now that for any problem of Bayesian inference we need to define four main parts: (i) What type of data are we dealing with for our inferences? (ii) What are the unknown parameters of interest that I want to infer? (iii) What is my current knowledge about the parameters of the model (prior) and how is this data distributed (likelihood)? How do we define the relationships between the data and the parameters (DAG)? (iv) Do our inferences make sense? We will now walk you through this workflow from a theoretical perspective but also with Stan code for its implementation.

### What type of data are we dealing with for our inferences?

The first thing to consider when modeling differential gene expression is the type of data used for inference. The first thing we need to define in Stan is the data. In this case, we need to define:

- The integer G will be the number of genes in the experiment.
- The integer S represents the number of samples from which differential expression will be inferred.
- An integer matrix of size G × S containing the expression data, particularly the number of counts per gene and per sample obtained in our experimental setup.
- The design matrices relating each sample to a given condition. Design vectors/matrices are binary vectors/matrices which associate each sample with the experimental condition they belong to, such that traditionally:

$$D_{ij} = \begin{cases} 1 & \text{if sample } i \text{ belongs to condition } j \\ 0 & \text{otherwise} \end{cases}$$

This encoding for the design matrix assumes choosing a reference sample (condition different from j) to which different priors to the rest of the samples will be defined. Although sometimes this may not impact final inference, it is definitely more appropriate to define equal priors for all samples, unless there are specific reasons not to do so. To solve this conundrum, the design matrix can be transformed from a D Boolean matrix of 0 and 1 into a D* matrix of –1 and 1 such that in a simple scenario with 2 different experimental groups, we can simplify the matrix $\boldsymbol{D}^*$ to a vector for which:

$$D^*_{ij} = \begin{cases} 1 & \text{if sample } i \text{ belongs to condition } j \\ -1 & \textit{otherwise} \end{cases} \quad \text{(eq. 4)}$$

This matrix D* will be a vector, as only two conditions are studied, named in our model as design2, and will allow us to define the same prior distribution for every condition.

The last element from the data block in our model is the effective gene length, $l_g$, of every gene included in the analysis. This element is named 'eff_length' in our model.

In the transformed data block, we first calculate the specific design matrix (design2) from the traditional one (design). After that, we compute a variable (expected_gene_mean) that will be used when defining the hyperpriors from a hyperparameter in the model, the total mean expression of the sample which will be defined as reference for the normalization process (in our model will be the first one, as will be explained in the next section). Finally, the effective gene lengths introduced in the model as data, are logarithmically transformed so that they are easily introduced into the model in the following steps.

```
data {
  int<lower=1> G;
  int<lower=1> S;
  int<lower=0> expression[G, S];
  vector<lower=0, upper=1>[S] design;
  vector<lower=0>[G] eff_length;
}

transformed data {
  vector<lower=-1, upper=1>[S] design2 = 2 * design - rep_vector(1, S);

  real<lower=0> expected_gene_mean = sum(expression[, 1])/sum(eff_length);
  vector<lower=0>[G] log_eff_length = log(eff_length);
}
```

## What are the unknown parameters of interest that I want to infer?

In a differential expression model, we are mainly interested in the effect of an experimental condition over the expected counts of the different genes' expression. We can define a single parameter per gene, $\beta_g$, representing the differences in the expression level between condition 1 and 2, for each gene. As the gene

expression cannot be negative, $\mu$, the expected expression per gene and per sample, is bounded at zero. For that reason, we can use $log(\mu)$ in our regression model. The gene expression in sample $s$ can be then obtained as:

$$log\left(\mu_{g,s}\right) = \alpha_g + \vec{\beta}_g D_s$$

where $\alpha_g$ is the expected expression levels of gene g in the fictional average sample, named "alpha" in our model. Using hierarchical modeling, the expected expression of every gene, $\alpha_g$, will come from a common distribution that is described by two other parameters, $\mu_\alpha$, $\sigma_\alpha$ (mu_alpha and sigma_alpha in our Stan model), related to the average expression of all genes and its standard deviation. The expected change in expression induced by each experimental condition, $\beta_g$, will depend on a hyperparameter $\sigma_\beta$, which will describe expected variability of the observed changes in expression, as explained below.

In addition to the biological changes experienced between conditions, we have an additional issue: samples are sequenced at relatively different depths, because the total number of sequenced fragments may change and thus the expected number of fragments from each gene may change. To tackle this issue, we define a new set of $S - 1$ parameters, being $S$ the number of samples, corresponding to the log transformation of the normalizing factors ($\log(f)$). By adding the corresponding factor $log(f_g)$ to the log-expression of each sample s ($log(\mu_{g,s})$), we are effectively scaling the expression of the remaining samples to have the same average across genes:

$$log\left(\mu_{g,s}\right) = \alpha_g + \vec{\beta}_g D_s + log\left(f_s\right)$$

The last parameter to define in our model is the dispersion parameter. Using Stan parametrization, we will use $\phi$ to model our data as a negative binomial distribution. This parameter gets infinite values in models where data is equidispersed (Poisson-based models). We assume our data is over-dispersed and therefore, use a negative binomial model. However, we impose a limit to the possible values $\phi$ can take to avoid computational problems. This limit is so high that an inferential result close to it would mean that our date is equidispersed and our model should be substituted by a Poisson model.

The model parameters are defined in Stan as follows:

```
parameters {
  real mu_alpha;
  real<lower=0>  sigma_alpha;
  vector[G] alpha;
  vector[G] beta;
  real<lower=0>  sigma_beta;
  vector[S-1] log_norm_factors_ref;

  real<lower=0, upper=2000000000> phi;
}

transformed parameters {

  vector[S] log_norm_factors = append_row(0, log_norm_factors_ref);

}
```

## What is my current knowledge about the parameters (prior)? How is the data distributed under this model (likelihood)?

In this model, $\alpha_g$ is equivalent to $\mu_g$ in the single sample model for a fictional average condition sample (eq. 2), and may assumed to be normally distributed around a mean of $\alpha_0$ and $\sigma_\alpha$:

$p(\alpha|\mu_\alpha, \sigma_\alpha) = Normal(\alpha|\mu_\alpha, \sigma_\alpha)$ The hyperprior of the hyperparameters $\mu_\alpha$, $\sigma_\alpha$ can be assumed also normal for simplicity, as described in the previous sections about hierarchical modeling. A possible option would be:

$$p\left(\sigma_\beta\right) = Normal\left(\mu_o | \log\left(\frac{n}{T}\right), 1.17\right)$$

$$p(\sigma_\alpha) = Normal^+ (0,2)$$

where $\frac{n}{T}$ has been calculated in the data block from the first sample (used as reference in the normalization) and has been called 'expected_gene_mean' in our model.

We expect that the total gene expression will remain constant after any treatment, as differences in total expression are more likely driven by differences in sequencing depths among samples. Thus, although the expression of some genes may increase *after* a certain treatment, there may be other genes with decreased expression levels such that the total gene expression remains constant. In other words, we expect $\beta_g$ to be normally distributed around zero, with a particular standard deviation $\sigma_\beta$. Similar assumptions have been implemented in DESeq2 (16) for shrinkage of estimated changes under an empirical Bayes framework (16). We still need to specify a prior for $\sigma_\beta$. As we vaguely expect little to no change in overall gene expression, we may set the mode at zero, with certain probability of being larger. This can be achieved with a Half-Normal distribution centered at 0 and with a standard deviation of 1, placing only 5% of the probability mass beyond 1.96, which is an already large standard deviation for $\beta_g$

$$p(\sigma_\beta) = Normal^+ (0,1)$$

About the normalization factors, as we expect samples to be relatively equilibrated in terms of total sequencing depths, we set a strongly informative prior on $log(\vec{f})$ around 0:

$$p(log(\vec{f})) = Normal^+ (0,0.05)$$

Because of the definition we have chosen for our design matrix, only half of the changes in expression $\beta_g/2$ will be summed to samples in condition j and substracted from samples in the other condition, being $\beta_g$ the total change induced when comparing both conditions.

Finally, to model our count data we will use a negative binomial that accounts for the over-dispersion generated by the technical and biological variability. In the previous sections, we have reparametrized the model to use an over-dispersion

parameter $\tau$, however, in our model, for simplicity, we will restrict ourselves to using Stan parametrization $\phi$, defining a non-informative uniform prior over it, but imposing a limit over which an increase would have no impacts in our inferences (high values of $\phi$ would mean no over-dispersion and those results would point to a Poisson model).

$$p(\phi) = Uniform(0,2000000000) \quad \text{(eq. 5)}$$

Finally, using the information given by the DAG (Figure 2) we can derive the full posterior distribution for the complete model for differential expression:

$$p\left(\vec{\alpha}, \vec{\beta}, \alpha_0, \sigma_\alpha, \sigma_\beta, \log\left(\vec{f}\right), \phi \mid \mathbf{k}, \mathbf{D}, \vec{l}\right)$$

$$= \frac{p\left(\mathbf{k} \mid \vec{\alpha}, \vec{\beta}, \tau, \vec{l}, log\left(\vec{f}\right), \mathbf{D}\right) p\left(\vec{\alpha} \mid \alpha_0, \sigma_\alpha\right) p\left(\vec{\beta} \mid 0, \sigma_\alpha\right) p(\alpha_0) p(\sigma_\alpha) p\left(\sigma_\beta\right) p\left(log\left(\vec{f}\right)\right) p(\phi)}{p(\mathbf{k})}$$

The Stan code to implement such model is:

```
model {
  mu_alpha ~ normal(log(expected_gene_mean), 1.17) ;
  sigma_alpha ~ normal(0, 2);
  alpha ~ normal(mu_alpha, sigma_alpha);
  sigma_beta ~ std_normal();
  beta ~ normal(0, sigma_beta);

  log_norm_factors_ref ~ normal(0, 0.05);

  phi ~ uniform(0, 2000000000);

  for (i in 1:G) {
    for (j in 1:S) {
      expression[i, j] ~ neg_binomial_2_log (alpha[i] + beta[i] *
design2[j] + log_eff_length[i] + log_norm_factors[j], phi);
    }
  }
}
```

As we already pointed out, Stan makes use of a programmatically more efficient computational MCMC algorithm, the Hamiltonian Monte Carlo. A very interesting introduction to HMC has been written by McElreath (18) and others (7, 19).

## Do our inferences make sense? Testing our model inference

After inferences are performed using Stan, it is important to check that chains have properly mixed and a high enough effective sample size is obtained for each parameter (warning messages appear if these conditions are not met), otherwise the model must be reparametrized. Once we can computationally trust our

inferences, we must consider if our results are sensible. This part of Bayesian modeling is more an art than a science and there is no defined strategy for it. We can draw inspiration from previously defined Bayesian workflows such as (20).

One option is to use leave-one-out cross-validation algorithms such as the loo R package to look for influential observations that may be distorting our inferences. To do so, the log likelihood (log_lik in our model) of each iteration in the model has to be computed. For that, the last block of Stan models can be used, the generated quantities block, where the variable log_lik is defined to be computed every iteration.

Another possibility is using posterior predictive checks: generating new expression samples for each value of the parameters inferred. This can be done in the generated quantities block of our Stan model, where a new variable 'expression_rep' has been defined. If the model works properly, our data (expression) must be in the range of our simulated data 'expression_rep'.

Our final generated quantities block would look like this:

```
generated quantities {
  matrix[G, S] log_lik;
  int expression_rep[G, S];
  for (i in 1:G) {
    for (j in 1:S) {
      log_lik[i, j] = neg_binomial_2_log_lpmf(expression[i, j] | alpha[i]
+ beta[i] * design[j] + log_norm_factors[j] + log_eff_length[i], phi);
      expression_rep[i, j] = neg_binomial_2_log_rng(alpha[i] + beta[i]
* design[j] + log_norm_factors[j] + log_eff_length[i],phi);
    }
  }
}
```

With this, our model is complete. Before relying on its inferences, an important aspect must be tested: its computational faithfulness; for example, the ability of the model to infer the values of parameters from data that has been generated from known parameters. This can be done by simulating new samples by sampling from the prior distribution of our model and performing inference upon those values. If our model is well calibrated, approximately 95% of the simulated parameters will be contained within a 95% credibility interval. Figure 3 shows the results obtained with this model for just one simulation from the prior. The code for the model and for sampling from a Stan model and running Stan from R using rstan is included in the supplementary materials and in a github account: https://github.com/vjimenezj/BayesBook

If this same process is performed multiple times, we are performing a simulated-based calibration (21): if we compute the rank of each simulation from the prior within all our posterior samples, these quantities must be uniformly distributed. These final checks ensure that if the data is configured as defined in the prior, the inference will be faithful. Unfortunately, nothing is certain about data that is behaving differently as described in our prior.

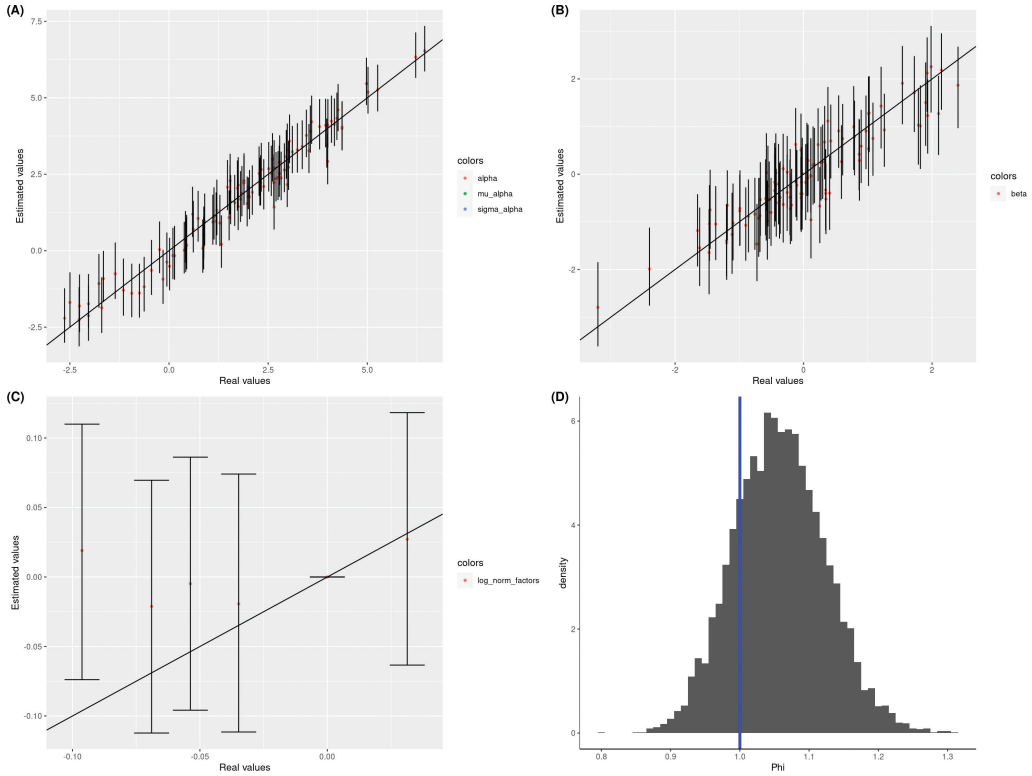



**Figure 3. Results from the simulation and inference of differential gene expression using the Negative Binomial model with Stan.** For the generation of these figures, a simulation of a dataset of 100 genes has been inferred with our model. For each parameter of interest, a posterior distribution of 2000 samples were obtained: **A.** $\mu_\alpha$, $\sigma_\alpha$ and $\alpha$[1:100]. **B.** Log normalization factors[1:6]. **C.** $\beta$[1:100] coefficients. The simulated value is represented in the x axis whereas the inferred value in the y axis. The 90% posterior credible interval is represented as a vertical segment that covers the values of this interval along the y axis. The colored points represent the mean of the posterior distribution of each of the parameters. Correctly inferred parameters are those whose corresponding segments cross the diagonal line, indicating that the credible interval contains the simulated value. If the model is correctly calibrated, we expect that 90% of the credibility intervals for our parameters contain the real value simulated and therefore approximately 90% of the vertical segments should touch the diagonal. In **D,** a histogram of the posterior distribution for $\Phi$ phi is shown, with the simulated value that is trying to be estimated depicted as a vertical line in blue.

## CONCLUSION

The definition of new methods for differential gene expression using Bayesian (22, 23) and non-Bayesian (15, 16, 17) methods has been an active research question in recent years, However, this chapter is not aimed at providing yet another method but, on the contrary, to show the reader how Bayesian inference is a flexible framework that can be used to make inference on the parameters of a model of increasing complexity and to provide with basic notions of *Stan* for its implementation. We believe that the rigorous, yet simple and systematic nature of Bayesian inference coupled with the latest advances in technology might strongly contribute to pushing the frontiers of knowledge.

**Acknowledgement:** We would like to acknowledge the members of the CNIC Bioinformatics Unit for continuous support, particularly to Fernando Martinez de Benito for support with the computing infrastructure and to Alvaro Serrano for long discussions about Bayesian inference and probabilistic models. FS-C received support from the Spanish Ministerio de Economía y Competitividad [grant no. RTI2018-102084-B-I00]; EL-P received support from the Spanish Ministerio de Economía y Competitividad (RTI2018-096961-B-I00), from the European Union (CardioNeT-ITN-289600 and CardioNext-ITN-608027) and the Spanish Carlos III Institute of Health (RD12/0042/066); M.A.d.P received support from the Spanish Ministerio de Economía y Competitividad (SAF2017-83130-R) and from the European Union Horizon 2020 research and innovation program under Marie Sklodowska-Curie grant agreement nº 641639 BIOPOL-ITN-641639; V.J-J. received an ESR contract from BIOPOL-ITN-641639). M.A.d.P is member of the Tec4Bio consortium (ref. S2018/NMT4443). The CNIC is supported by MCIU and the Pro-CNIC Foundation and is a Severo Ochoa Center of Excellence [MCIU award SEV-2015-0505].

**Conflict of interest:** The authors declare no potential conflict of interest with respect to research, authorship and/or publication of this chapter.



## REFERENCES

1. DeRisi J, Penland L, Brown PO, Bittner ML, Meltzer PS, Ray M, et al. Use of a cDNA microarray to analyse gene expression patterns in human cancer. Nat Genet. 1996;14(4):457-60. https://doi.org/10.1038/ng1296-457
2. Marioni JC, Mason CE, Mane SM, Stephens M, Gilad Y. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. Genome Res. 2008.;18(9):1509-17. https://doi.org/10.1101/gr.079558.108
3. Brazma A. Minimum information about a microarray experiment (MIAME) - Successes, failures, challenges. ScientificWorldJournal. 2009;9:420-3. https://doi.org/10.1100/tsw.2009.57

4. Thomas A, O'Hara RB, Ligges U, Sturtz S. Making BUGS Open. Making BUGS Open. R News 2006; 6(1): 12-17R News 2010.
5. Kruschke JK. Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan, second edition. Cambridge, USA. Academic Press; 2014. 776 p. https://doi.org/10.1016/B978-0-12-405888-0.00008-8
6. Stan Development Team. Stan Modeling Language User's Guide and Reference Manual, Version 2.19.2. Interact. Flow Model. Lang. 2020.
7. Betancourt M, Girolami M. Hamiltonian Monte Carlo for Hierarchical Models. In: Current Trends in Bayesian Methodology with Applications. London, UK, Chapman and Hall/CRC; 2015. 680 p.2015.
8. Salvatier J, Wiecki T V, Fonnesbeck C. Probabilistic programming in Python using PyMC3. PeerJ Computer Science 2:e55. https://doi.org/10.7717/peerj-cs.55
9. Piponi D, Moore D, Dillon J V. Joint Distributions for TensorFlow Probability. arXiv 2020. https://arxiv.org/pdf/2001.11819.pdf
10. Vogl C, Sanchez-Cabo F, Stocker G, Hubbard S, Wolkenhauer O, Trajanoski Z. A fully bayesian model to cluster gene-expression profiles. Bioinformatics. 2005;21 Suppl 2:ii130-6. https://doi.org/10.1093/bioinformatics/bti1122
11. Glaus P, Honkela A, Rattray M. Identifying differentially expressed transcripts from RNA-seq data with biological variation. Bioinformatics. 2012;28(13):1721-8. https://doi.org/10.1093/bioinformatics/bts260
12. Li B, Dewey CN. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. BMC Bioinformatics. 2011; 12:323. https://doi.org/10.1186/1471-2105-12-323
13. Bray NL, Pimentel H, Melsted P, Pachter L. Near-optimal probabilistic RNA-seq quantification. Nat Biotechnol. 2016;34(5):525-7. https://doi.org/10.1038/nbt.3519
14. Patro R, Duggal G, Love MI, Irizarry RA, Kingsford C. Salmon provides fast and bias-aware quantification of transcript expression. Nat Methods. 2017;14(4):417-419. https://doi.org/10.1038/nmeth.4197
15. Robinson MD, McCarthy DJ, Smyth GK. edgeR: A Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 2009;26:139-140. https://doi.org/10.1093/bioinformatics/btp616
16. Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. 2014;15:550. https://doi.org/10.1186/s13059-014-0550-8
17. McCarthy DJ, Chen Y, Smyth GK. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res. 2012;40:4288-4297. https://doi.org/10.1093/nar/gks042
18. McElreath R. Statistical rethinking: A bayesian course with examples in R and stan. 2018. London, UK; Chapman & Hall/CRC Texts in Statistical Science; p. 612. https://doi.org/10.1201/9781315372495
19. Betancourt M. A Conceptual Introduction to Hamiltonian Monte Carlo. arXiv 2017. https://arxiv.org/abs/1701.02434; https://doi.org/10.3150/16-BEJ810
20. Schad DJ, Betancourt M, Vasishth S. Toward a principled Bayesian workflow in cognitive science. arXiv 2019. https://doi.org/10.1037/met0000275
21. Talts S, Betancourt M, Simpson D, Vehtari A, Gelman A. Validating Bayesian inference algorithms with simulation-based calibration. arXiv 2018. https://arxiv.org/pdf/1804.06788.pdf
22. Lee J, Ji Y, Liang S, Cai G, Müller P. Bayesian Hierarchical Model for Differential Gene Expression Using RNA-Seq Data. Stat Biosci. 2015;7(1):48-67. https://doi.org/10.1007/s12561-013-9096-7
23. Zheng S, Chen L. A hierarchical Bayesian model for comparing transcriptomes at the individual transcript isoform level. Nucleic Acids Res. 2009;37(10):e75. https://doi.org/10.1093/nar/gkp282