Journal of Computational Science 47 (2020) 101245

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

OutPyR: Bayesian inference for RNA-Seq outlier detection

Edin Salkovic [a,*], Mostafa M. Abbas [b], Samir Brahim Belhaouari [a], Khaoula Errafii [c,d], Halima Bensmail [b,a]

[a] College of Science Engineering, Hamad Bin Khalifa University, PO Box 34110, Doha, Qatar
[b] Qatar Computing Research Institute, Hamad Bin Khalifa University, PO Box 34110, Doha, Qatar
[c] Qatar Biomedical Research Institute, Hamad Bin Khalifa University, PO Box 34110, Doha, Qatar
[d] College of Health and Life Sciences, Hamad Bin Khalifa University, PO Box 34110 Doha, Qatar

ARTICLE INFO

Keywords:
RNA-Seq
Outlier detection
Bayesian modeling

ABSTRACT

High-throughput RNA sequencing technologies (RNA-Seq) have recently started being used as a tool for helping diagnose rare genetic disorders, as they can indicate abnormal gene expression counts — a telltale sign of genetic pathology. Existing solutions either require a large number of samples or do not provide proper statistical significance testing.

We present a Bayesian model (OutPyR) for identifying abnormal RNA-Seq gene expression counts in datasets, particularly those with a small number of samples. The model incorporates recently introduced data-augmentation techniques to efficiently and accurately infer parameters of the underlying negative binomial process, while also assessing the uncertainty of the inference, and giving the possibility to generate simulated data. The model's software implementation is object oriented and thus easily extensible, provides parameter-trace exploration, fault-tolerance and recovery during the parameter estimation process. We also develop a *p*-value based outlier score that naturally stems from our model. We apply the model to real and simulated datasets, for different organisms and tissues, and present comparisons with existing models.

Our model is implemented purely in Python and its standalone source code is available at https://github.com/esalkovic/outpyr.

1. Introduction

One area in which RNA-Seq data analysis has been gaining traction recently is identifying the causes of Mendelian disorders, which have traditionally been studied using whole or exome genome sequencing (WGS or EGS). Mendelian disorders are genetic disorders that stem from mutations in a single gene, and can hence be inherited according to Mendel's law. In many patients that might be suffering from these types of disorders pathogenic variants cannot be identified from genome sequencing data alone [1,2]. One suspected reason is that these variants are regulatory in nature, and regulatory sequences are significantly less well understood than coding sequences. Another difficulty is that the number of rare non-coding variants is large: around 60,000 non-coding single nucleotide variants (SNVs) vs. 475 rare variants that affect proteins [3,4].

Several recent studies have shown that RNA-Seq data can add value to the discovery of pathological regulatory gene expression such as: discovery of expression outliers, defects in splicing and loss-of-function heterozygous variants that are expressed in one allele [5,6].

When it comes to the identification of outliers, these studies approached it differently. Cummings et al. [6] used z-scores of logarithms of gene counts that were normalized (mean subtracted and divided by standard deviation). In this setting, the read counts that had a greater than 3 z-score were identified as outliers, without any statistical significance test. However, no outliers were identified using their setup. On the other hand, Kremer et al. [5] used DESeq2's [7] z-score for the same task, by identifying as statistically significant outliers those counts that had a *z*-score greater than 3. They applied the test to every sample by comparing it to the rest of the group and identified 5 potential outliers within their count data.

Brechtmann et al. [4] proposed a model for identifying aberrant gene expression in RNA-Seq samples that also controls for co-variation in gene read counts. They implemented the model as a Bioconductor package called OUTRIDER. They use an autoencoder (AE) for modeling

* Corresponding author.
E-mail address: esalkovic@hbku.edu.qa (E. Salkovic).

https://doi.org/10.1016/j.jocs.2020.101245
Received 14 April 2020; Received in revised form 29 September 2020; Accepted 18 October 2020
Available online 31 October 2020

expectations of read counts based on co-variation among genes. The co-variation is caused by genetic, environmental and technical factors, none of which are known *a priori*. The read counts are modeled as to follow a negative binomial distribution (NB), whose dispersion parameter is gene specific. They fit the model in such a way as to, reportedly, get the best correction of data that is corrupted artificially. An important aspect of their work is that it clearly shows that statistical outliers can directly help in identifying actual aberrantly expressed genes. Concretely, 146 statistical outliers identified automatically by their tool contained 6 pathogenic outliers validated by Kremer et al. [5], whereas the ad hoc approach of Kremer et al. [5] identified 557 statistical outliers, out of which only 4 were pathogenic.

One limitation of their model is that it relies on a considerable number of samples (at least 60) for recalling the majority of pathogenic events. However, performance on smaller dataset sizes is relevant, due to the associated cost of producing the data, and due to the available number of patients for some particular rare disease in actual clinical practice. Another downside is the inherent ad hoc nature of the way they train their model, utilizing noise injection and an arbitrary number of training iterations. Due to that, their model, as well as their statistical p-value based outlier score, cannot precisely express the uncertainty of the estimated parameters of the NB, and similar can be said for the other models we mentioned.

To help mitigate some of the shortcomings of previous approaches we introduce OutPyR,[1] a Bayesian method to detect abnormal values in datasets of RNA-Seq gene expression counts with a small number of samples. It uses a systematic statistical approach to model the data based on recent advances in Bayesian inference that allow it to correctly estimate its model's parameters even for a small number of datapoints, while still quantifying uncertainty of those parameters and therefore allowing its propagation to downstream analysis.

1.1. OutPyR's approach

A high-level overview of OutPyR's workflow can be seen in Fig. 1: after obtaining raw RNA-Seq data from a sequencing machine and using some of the widely available tools to map such raw data to an appropriate genome, a gene expression count matrix is produced; OutPyR takes such a matrix as input and, based on it, estimates the parameters of its Bayesian model using a Markov chain Monte Carlo (MCMC) method for sampling the posterior distribution of the parameters; from the MCMC sampling traces, a matrix of p-values (i.e. means of the p-value trace) and their standard deviations (capturing uncertainty) will be produced for every count, allowing direct outlier identification, or further downstream analysis.

OutPyR is written in Python, using an object-oriented and modular approach. We assume a NB distribution of the underlying process that generates the gene expression counts, and rely on recent advances in data-augmentation techniques by Zhou et al. [8] and Zhou and Carin [9] in order to obtain a closed-form Bayesian inference of the posterior distribution through efficient Gibbs sampling. More concretely, these advances allow a mathematically correct estimation of the dispersion parameter of NB, even when the number of data points is small. The dispersion parameter is critical in NB modeling, and its inference using the maximum likelihood estimator (ML), as performed by Brechtmann et al. [4], could be significantly biased and could actually fail when the number of samples is small [10].

1.2. Our contribution

Here we briefly outline four important contributions of this work towards improved aberrant gene expression count detection in RNA-Seq data:

1. p-value reporting along with uncertainty.
2. Compared to recent work done in Bayesian inference for DE by Dadaneh et al. [11] who model the dispersion parameter r to be sample-dependent, in our model r is gene-dependent in order to allow for the calculation of p-values that are gene-specific. Moreover, unlike DE approaches which assume a case/control setup, our approach assumes that all samples can be treated as belonging to the same homogeneous group, and gene-specific r inference across all samples in this case is more appropriate for outlier identification.
3. Additionally, our model is flexible, and can be matched upstream with other models, e.g. any batch-effect or confounder controlling tool that supports integration with external tools.
4. Our model is easily parallelizable, can be easily implemented on a GPU, and even allows for a combined CPU + GPU approach.

1.3. Paper structure

In this article we adopt a Bayesian framework to assess the discovery of plausible aberrant counts. We show how our simple model accounts for uncertainty while predicting aberrant genes counts. In Section 2 we introduce the notation, the count model, the p-value based outlier score, and some software implementation details of OutPyR. In Section 3 we introduce real and synthetic benchmark datasets we use for measuring performance of OutPyR against competing methods. In Section 4 we briefly introduce those methods and provide a detailed analysis of theirs, as well as OutPyR's results after running them on our benchmark datasets. Finally, we conclude the paper with Section 5.

2. Methods

2.1. Count model

The NB is a distribution used for modeling overdispersed count data. The NB model consists of a negative binomial component representing the count process. For RNA-Seq data, the model represents the process that gives rise to the count data. Since the counts of a gene across samples are often over-dispersed, it is natural to model them using the NB, where its variance σ^2 is related to its mean μ as:

$$\sigma^2 = \mu + \frac{1}{r}\mu^2 \tag{1}$$

r is called the dispersion parameter.[2] Also, the NB can be parameterized in two common ways: either via its mean μ and dispersion parameter r, or via its dispersion parameter r and probability parameter p which are related by $\mu = \mathrm{rp}/(1-p)$ and $\sigma^2 = \mu(1-p)$. The NB has been widely applied to model biological data [12], socio-economic data [13] and ecological data [14], to name a few.

It is well known that when a distribution belongs to the exponential family of distributions, such as the NB, there exists an associated class of priors called the class of conjugate priors which are appealing for the simple computational reason that the posterior distributions are exactly of the same form as the prior distributions. And it is also well known that the NB can be seen as equivalent to a Gamma–Poisson mixture distribution in which we can obtain a count $c|p,r \sim \mathrm{NB}(p,r)$ by first drawing $\lambda \sim \Gamma(r,(1-p)/p)$ and then generating $c|\lambda \sim \mathrm{Pois}(\lambda)$. Furthermore, it is straightforward to prove that the beta distribution is the conjugate prior of the NB probability parameter p $(0 < p_j < 1)$, which makes its inference trivial. However, the conjugate prior of NB's dispersion parameter r is unknown and its inference has been a challenge until recent work by Zhou et al. [8], Zhou and Carin [9]. They show that a NB can also be augmented through an intermediate Poisson representation, so that by

[1] Rhymes with "outlier".

[2] Some authors use the term "dispersion" for $\frac{1}{r}$, and the term "shape" for r, or vice versa.

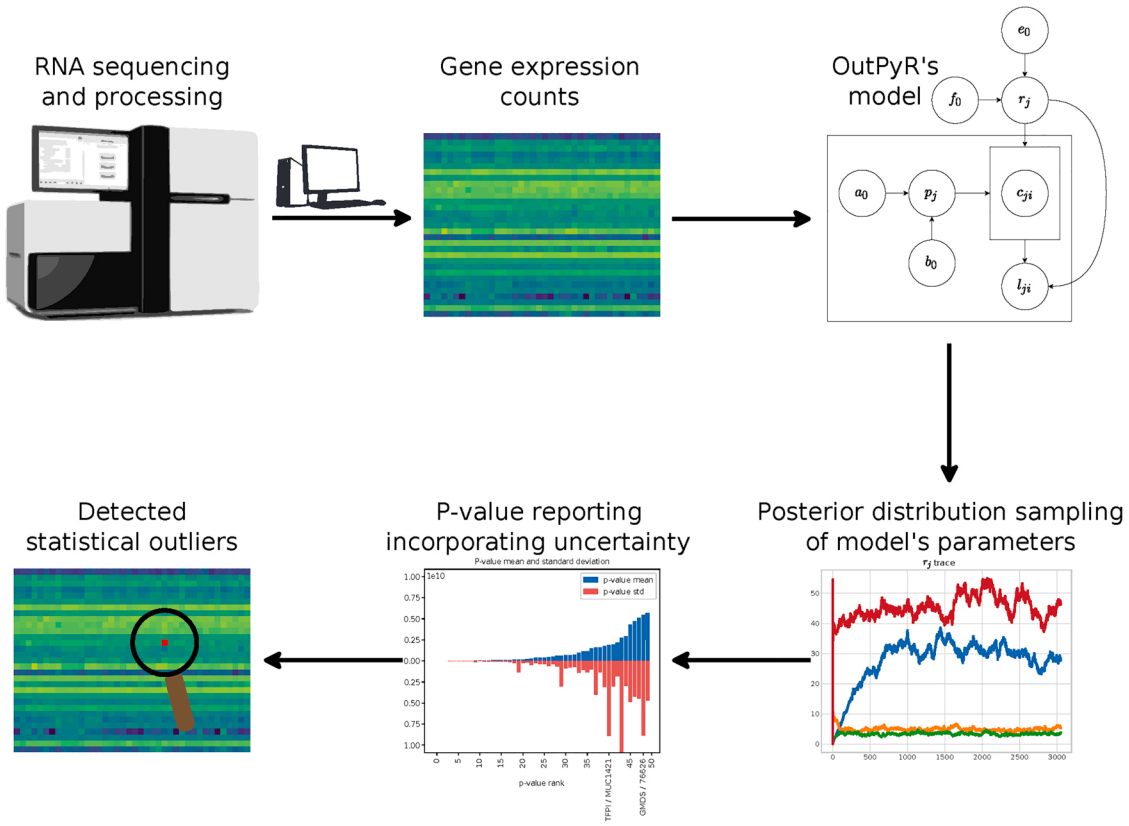


Fig. 1. A typical OutPyR workflow. A gene expression count dataset obtained through an upstream RNA-Seq sequencing and processing protocol is fed into OutPyR's model. Based on this dataset, OutPyR infers the posterior trace of the model's parameters using tractable MCMC Gibbs sampling. Posterior predictive *p*-values are calculated for the whole parameters' trace, thus capturing uncertainty. Based on these *p*-values, statistically aberrant counts are detected.

including both parameters r and p with well-chosen conjugate priors, it will be possible to tractably draw samples from their posterior distribution. For this reason, we parameterize the NB with its dispersion r and probability p parameters throughout the paper.

As a summary, Eqs. (2) and (3) describe the NB equation we use as our data model. The count data are modeled as:

$$c_{ji}|p_j, r_j \sim \mathrm{NB}(p_j, r_j) \tag{2}$$

where $j \in 1, \ldots, J$ denotes a gene, while J is the total number of genes; and where $i \in 1, \ldots, N$ denotes a sample, while N is the total number of samples. Therefore, the counts c_{ji} for a particular gene j result from an underlying NB process that is gene-specific. This process is fully parameterized with p_j $(0 < p_j < 1)$, which is the probability parameter, and $r_j > 0$, which is the dispersion parameter, accounting for over-dispersion. The probability mass function (PMF) is then:

$$\mathrm{PMF}_{\mathrm{NB}}(c_{ji}|p_j, r_j) = \binom{r_j + c_{ji} - 1}{c_{ji}} p_j^{c_{ji}} (1 - p_j)^{r_j} \tag{3}$$

If we allow c_{ji} and r_j to be continuous, then we can define an analogous probability density function as:

$$\mathrm{PDF}_{\mathrm{NB}}(c_{ji}|p_j, r_j) = \frac{\Gamma(c_{ji} + r_j)}{\Gamma(c_{ji} + 1)\Gamma(r_j)} p_j^{c_{ji}} (1 - p_j)^{r_j} \tag{4}$$

This parametrization, provided appropriate priors are chosen, allows Gibbs sampling of the posterior distributions of both p_j and r_j parameters in closed form using recently introduced data augmentation methods by Zhou and Carin [15,9]. The sampling procedure is expounded in Section 2.3, Eqs. (6)–(11).

2.2. Pre-processing

OutPyR supports DESeq's normalization [16], namely the sample-wise size factors based on geometric means that control for library size differences and sequencing depth:

$$s_i = \underset{i}{\mathrm{median}} \frac{c_{ji}}{\left(\prod_{t=1}^{N} c_{jt}\right)^{\frac{1}{N}}} \tag{5}$$

However, by default, no normalization is applied, which allows for other normalization methods to be applied if desired. If DESeq-like normalization is enabled, the input to the Bayesian parameter inference model, c_{ji}, will be the normalized value.

2.3. Bayesian inference

To model the gene counts in each sample with the NB, we consider the null hypothesis that with sample-specific NB probability parameters, the posterior distributions of the gene-specific NB dispersion parameters, regularized by the gamma process in the prior, are the same across different samples. Eq. (6) shows the weak conjugate priors for parameters p_j, r_j:

$$\begin{aligned} p_j &\sim \mathrm{Beta}(a_0, b_0) \\ r_j &\sim \mathrm{Gamma}(e_0, 1/f_0) \end{aligned} \tag{6}$$

where $a_0 = b_0 = 1$, and $e_0 = f_0 = 0.1$ are the hyperparameters of the priors, that can be changed by the user. The choice of the hyperparameter values is not that important, as long the initial values of the priors are valid: the posterior samples will over time converge to the correct parameter values, due to the aforementioned tractability of the sampling procedure.

During posterior sampling, the parameters are updated as follows:

$$p_j|c_{ji}, r_j \ldots \sim \text{Beta}\left(a_0 + \sum_{i=1} c_{ji}, b_0 + n r_j\right) \tag{7}$$

$$r_j|l_{ji}, p_j \ldots \sim \text{Gamma}\left(e_0 + \sum_{i=1} l_{ji}, \frac{1}{f_0 - nlog(1-p_j)}\right) \tag{8}$$

$$l_{ji}|c_{ji}, r_j \ldots \sim \text{CRT}(c_{ji}, r_j) \tag{9}$$

where CRT stands for Chinese Restaurant Table distribution. The CRT is computed as follows:

$$l_{ji} = \sum_{m=1}^{c_{ji}} b_m \tag{10}$$

$$b_m \sim \text{Bernoulli}\left(\frac{r_j}{m - 1 + r_j}\right) \tag{11}$$

The time complexity of computing the value of the latent count l_{ji} is $O(c_{ji})$, i.e. linear with respect to a particular count c_{ji}, so higher counts will take more time to compute.

The whole graphical model can be seen in Fig. 2.

2.4. Posterior predictive p-values

In Bayesian models generally it is possible to calculate the posterior predictive *p*-value using an arbitrary discrepancy function, provided it is a function of the parameters of the model and the data [17]. Here we use the empirical CDF function as the discrepancy function.

For every pair of gene *j* and sample *i*, we test the null hypothesis that counts follow the distribution described by Eq. (2). The algorithm

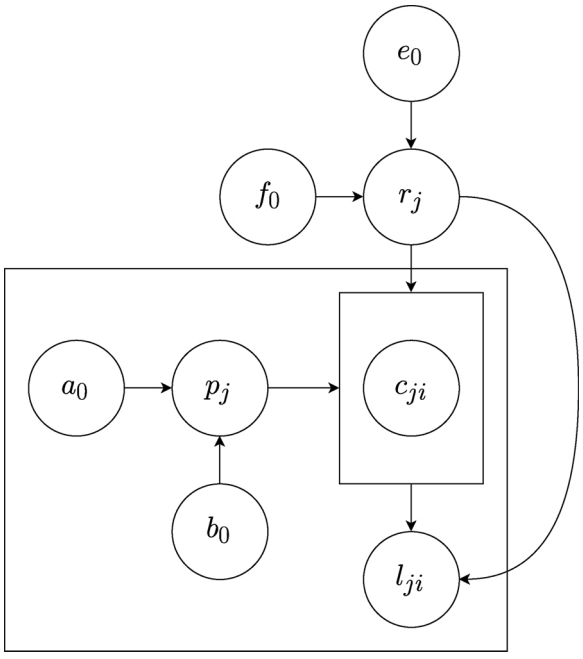


Fig. 2. A graphical representation of OutPyR's parameter model ((6)–(11)). The circle containing c_{ji} represent the data—RNA-Seq GE counts (2). Circles containing zero-indexed letters represent prespecified hyperparameters. The remaining circles represent the latent counts l_{ji} (9) and the gene-specific NB parameters p_j (7) and r_j (8) which are used for calculating the final *p*-values. In the Gibbs sampling inference procedure parameters p_j and r_j are initialized using their respective hyperparameters (6); subsequently, their posterior estimates are sampled using (7) and (8), with latent count l_{ji} being the recently introduced [8] key component for correctly estimating r_j.

computes two-sided posterior predictive *p*-values for which we assume that under H_0, c_{ji} has a $\text{NB}(p_j, r_j)$ distribution. Then we calculate the predictive posterior *p*-values as follows:

$$p\text{-value}_{ji} = 2 \cdot min\left\{\text{CDF}_{\text{NB}}(c_{ji}|p_j, r_j), 1 - \text{CDF}_{\text{NB}}(c_{ji}|p_j, r_j)\right\} \tag{12}$$

where $\text{CDF}_{\text{NB}}(c_{ji}|p_j, r_j)$ is the cumulative distribution function of the NB, which can be calculated with:

$$\text{CDF}_{\text{NB}}(c_{ji}|p_j, r_j) = \int_{x=0}^{c_{ji}} \text{PDF}_{\text{NB}}(x|p_j, r_j)\text{dx} \tag{13}$$

$\text{PDF}_{\text{NB}}(x|p_j, r_j)$ was previously defined in Eq. (4).

We can treat *p*-values obtained in this way as an outlier score: the lesser the *p*-value, the more of an outlier a particular count will be according to OutPyR's model.

2.5. Computational and software implementation details

OutPyR is written only in Python, using widely available packages. It has been tested on Linux and Windows. Several options can be modified by supplying a settings file. OutPyR can take advantage of multiple cores, following a multi-process approach. It supports progress reporting, trace saving to disk, with recovery if the program is stopped during execution. It uses pre-defined, but easily modifiable, Pseudo Random Number Generator (PRNG) seeds, allowing for experiment reproducibility. In the case of using multiple cores, every (sub)process has its own PRNG, which allows computationally correct sampling. In addition, the full PRNG states for every process are saved along with the trace, providing resilient recovery even in the multi-process setting.

If multi-processing is enabled it splits the data according to the desired number of virtual cores/processes, albeit smartly, taking into consideration not only the number of genes, but also the sum of counts in each gene, as the inference is computationally bottlenecked by CRT, which in turn is bottlenecked by count values, as outlined previously in comments regarding Eq. (10).

Still, the whole running time is somewhat affected by saving the trace to disk, although it is possible to set the times at which the script saves to disk, or disable it completely. Consideration has to be taken with regards to how much RAM is utilized, as the script will keep part of the trace in RAM between saves.

In order for the Python code execution to be as fast as possible, NumPy's [18] array processing methods are used wherever possible. The CRT distribution is implemented with Numba [19], which produces binary code whose speed is comparable to a C/C++ based implementation, but still allows for writing code in pure Python without the need for a C/C++ compiler toolchain. Due care is taken to synchronize NumPy's PRNG states with Numba's, in order to ensure proper PRN generation.

As we rely on a closed-form and tractable Gibbs sampling procedure to infer all parameters, there is no need for running multiple sampling chains in order to check the convergence of the sampling procedure: the samples are guaranteed to converge to the actual posterior distribution after a certain number of iterations. For datasets with a small number of samples, we used 3000 iterations for the warm-up[3] stage, and saved the consequent 1000 posterior samples of p_j and r_j. For datasets with a large number of samples ($N \geq 100$) we used 2000 iterations for the warm-up, and 1000 as posterior samples. The smaller warm-up number for larger datasets was used as the sampling chains' convergence took less iterations than for the smaller datasets, and also helped in decreasing the overall running time. The running time of the whole sampling procedure on a 24-core Threadripper 2970WX CPU was 1-3 h for smaller datasets, while it took 4-6 h for larger datasets. The longer times are due to larger outliers which detrimentally affect the CRT calculation step.

[3] Also called "burn-in".

Then, we calculated the predictive posterior p-value from Eq. (12), for all posterior samples and for every count. We then calculated their (a) mean for every count, which is our reported p-value estimate that can be used for comparison with other models; and (b) standard deviation, which contains the information about the uncertainty of the estimate.

3. Datasets used for experiments

3.1. Base datasets

To measure the performance of OutPyR and competing methods, we relied on three base datasets: two real datasets and one synthetic. We used only real datasets with a relatively small number of samples, as processing datasets with a large number of samples requires confounder control, which OutPyR does not implement currently.

The base datasets are the following:

1. **Kremer-10** The first real dataset is a 10-sample subset of the dataset introduced in [5], containing 119 samples of skin fibroblasts RNA-Seq. The patients from that study were suspected of having some rare mitochondrial disorder. Non-expressed genes were filtered out utilizing the pipeline of Brechtmann et al. [4], resulting in retaining 10,556 out of 27,682 genes. We selected 10 samples that contained most prominent outliers according to OUTRIDER. This was done to make it more difficult for the models to recall artificial outliers injected according to the procedure described in Section 3.2.
2. **GSE62368-16** The second 16-sample real dataset was chosen in order to show the applicability of OutPyR and other models for other types of organisms. It is accession GSE62368 from Gene Expression Omnibus (GEO) [20] which contains RNA-Seq data of whole circulating blood from a control group of 16 Rattus norvegicus organisms. Three hours prior to their blood being utilized for sequencing, these organisms were subjected to four doses of 1 ml/kg saline injections every two hours, while not having access to food nor water. We downloaded the gene expression count matrix from the GEO website and filtered out non-expressed genes utilizing the pipeline of Brechtmann et al. [4], resulting in retaining 17,781 out of 18,031 genes.
3. **Simulated-100** We generated a 100-sample synthetic dataset using inferred p_j and r_j values from the filtered gene subset of the Kremer et al. [5] full dataset (all 119 samples) together with a NB random generator in order to get artificial data that are similar to Kremer et al. [5] data. As there are no confounder effects in this dataset, even though the number of samples is high (100) OutPyR should be able to compete successfully with the other methods.

3.2. Datasets derived using artificial outlier injection

In order to accurately assess and compare the performance of all methods, we tested them on datasets that were based on aforementioned real and synthetic datasets, but contained artificially injected outliers.

The outliers were injected as follows:

- an outlier could be injected in any cell of the count matrix, if possible;
- frequency of injection was 10^{-4};
- there were three schemes corresponding to
  1. only overexpressed outliers (the derived dataset would contain only injected outliers whose value was greater than the mean/mode of a particular gene),
  2. only underexpressed (outlier value was less than the mean/mode of a particular gene), and
  3. both overexpressed and underexpressed outliers with a ratio of 50%/50%;

  under each of these three schemes there were two sub-schemes where outliers were injected with z-scores of 3 and 4, although there was no mixing of different z-scores in the same dataset. Consequently, for each of the three base datasets, there were $3 \times 2 = 6$ derived datasets, corresponding to combinations of different types of injected outliers (3) and different z-scores (2). So, in total, there were $3 \times 3 \times 2 = 18$ derived datasets.

Using a similar set up as in [4], the original counts were replaced by outliers whose rounded values were calculated as follows:

$$o_{ji} = \left[s_i 2^{\mu_{q_j} \pm e^z \sigma_{q_j}} \right] \tag{14}$$

where z is the z-score (3 or 4), and μ_{q_j} and σ_{q_j} are the gene-wise mean and standard deviation of

$$q_{ji} = log_2 \left(\frac{c_{ji}}{s_i} + 1 \right) \tag{15}$$

s_i is the sample-specific size factor (equation (5)).

In contrast to the outlier injection procedure of Brechtmann et al. [4], an upper bound cutoff of $2^{24} - 1 = 16{,}777{,}215$ was enforced for overexpressed injected counts, because higher values would lead to slow CRT computation and therefore slow down the overall inference. Also, counts higher than this cutoff might be unrealistic for actual datasets: e.g. the maximum count value in the Kremer et al. [5] dataset is 4,718,136, while in GSE62368 it is 2,712,788.

In the outlier injection algorithm for underexpressed genes, for the special case when the value of the outlier was 0 (i.e. smallest outlier possible), we made sure that only one z-score value will produce an outlier of 0 in a particular cell: e.g. if a 0 outlier is injected at some particular location in the matrix with a z-score of 3, then it will be impossible to inject a 0 outlier with a z-score of 4 in that same cell. This issue of outliers whose value is 0 was overlooked by Brechtmann et al. [4]. Brechtmann et al. [4] also used a z-score of 2 in their outlier-injecting experiments, but we did not, as both Cummings et al. [6] and Kremer et al. [5] used only z-scores greater or equal to 3 for determining outliers.

4. Results and discussion

We compared OutPyR's performance on the derived datasets against (1) the novel autoencoder-based OUTRIDER model introduced by Brechtmann et al. [4], denoted by OUTRIDER-AE; (2) the alternative model that OUTRIDER supports using PEER [21] for confounder control, denoted by OUTRIDER-PEER; and (3) another alternative model that OUTRIDER supports which uses PCA for confounder control, denoted by OUTRIDER-PCA.

For all methods, data were normalized using DESeq's normalization. For the encoding dimension option requirement of OUTRIDER's autoencoder, the dimension was set to the value returned by its provided function `findEncodingDim`. Other parameters of the autoencoder were left at their default values.

In order to compare the performance of OutPyR against other methods we ran both OutPyR and other methods on the aforementioned derived datasets of all base datasets, estimated their model parameters and finally calculated the p-values for every count based on those parameters, using functions provided by the methods. Similar to OutPyR's p-values, these p-values can be treated as an outlier score: the smaller the p-value, the greater an outlier a particular count is for a particular method.

Ideally, the smallest p-values would correspond directly to actual injected outliers, however, in reality, none of the methods were detecting outliers perfectly. Therefore, to accurately compare the methods against each other, we did not use absolute p-values, but we instead ranked p-values of each method separately from the smallest to the largest. In that way, the p-value ranks were identical between different methods, and thus directly comparable. The method that

would have actual injected outliers among the lowest *p*-value ranks would be the best performing method.

However, in most of the cases there was no method that would clearly outperform other methods for every single outlier. So, in order to visually compare the performance of all methods, we plotted precision-recall (PR) plots of all methods against each other using different colors. We then grouped such (sub)plots of all 6 derived datasets of a particular base dataset in a single plot. Fig. 3 shows such a group of PR plots for datasets derived from the Kremer-10 base dataset.

As sometimes PR curves might not give a clear visual clue for which method is performing best, in order to get a precise comparison among different methods we also calculated area under curve (AUC) numerical values for the PR curves of each method and each derived dataset, and grouped such values of all derived datasets of a particular base dataset in a single table. One such table is Table 1, showing AUC-PR values of all methods for all derived datasets from the Kremer-10 base dataset. The AUC-PR values in the table correspond to PR curves in Fig. 3.

If we look at Table 1, we can see that OutPyR is outperforming other models for all 6 datasets derived from the Kremer-10 base datasets, except for the derived dataset that contains underexpressed artificial outliers with a z-score of 4, where it is outperformed with a slight margin by OUTRIDER-PEER (0.4344 vs. 0.4686). In relative terms, i.e. when we look only at the ratios between AUC-PR values of different methods, and not at the absolute value, the AUC-PR value of OutPyR is almost 4 times greater than the next method's for the case of overexpressed outliers with a z-score of 4. However, for that particular derived dataset all methods perform poorly in absolute terms, with the best performing OutPyR achieving an AUC-PR value of only 0.0237. For the remaining derived datasets, OutPyR outperforms other methods by a margin of at least 10%, and up to 70%. It is interesting to note that the second method

Table 1
AUC values for PR curves for different methods that were ran on the datasets derived from the Kremer-10 base dataset, corresponding fully to Fig. 3 (refer to its caption for more detail).

	Over- &Underexpressed	Overexpressed	Underexpressed
z-score 3			
OutPyR	**0.2220**	**0.1148**	**0.3792**
OUTRIDER-AE	0.1133	0.0017	0.1782
OUTRIDER-PEER	0.2067	0.0966	0.2205
OUTRIDER-PCA	0.0846	0.0305	0.0209
z-score 4			
OutPyR	**0.2222**	**0.0237**	**0.4344**
OUTRIDER-AE	0.0546	0.0011	0.2781
OUTRIDER-PEER	0.1457	0.0036	0.4686
OUTRIDER-PCA	0.0102	0.0062	0.0263

performance-wise is consistently OUTRIDER-PEER, and not the novel OUTRIDER-AE.

Similarly, for the GSE62368-16 base dataset, Fig. 4 shows the PR plots of OutPyR and other methods for various outlier injection schemes, while their AUC-PR values can be seen in Table 2.

Again, OutPyR is, on average, the best performing method. However, this time his AUC-PR score is outmatched by the almost 4 times better OUTRIDER-PCA for the derived dataset containing only overexpressed genes with a z-score of 3. Still, on this derived dataset all methods are performing poorly, with the best OUTRIDER-PCA having a 0.1619 AUC-PR value. Additionally, OUTRIDER-AE is outperforming OutPyR by a slight margin for the case of only underexpressed outliers with a z-score of 3 (0.6835 vs. 0.6333). For the remaining derived datasets OutPyR is outperforming other models by a margin of at least 28%. Interestingly,

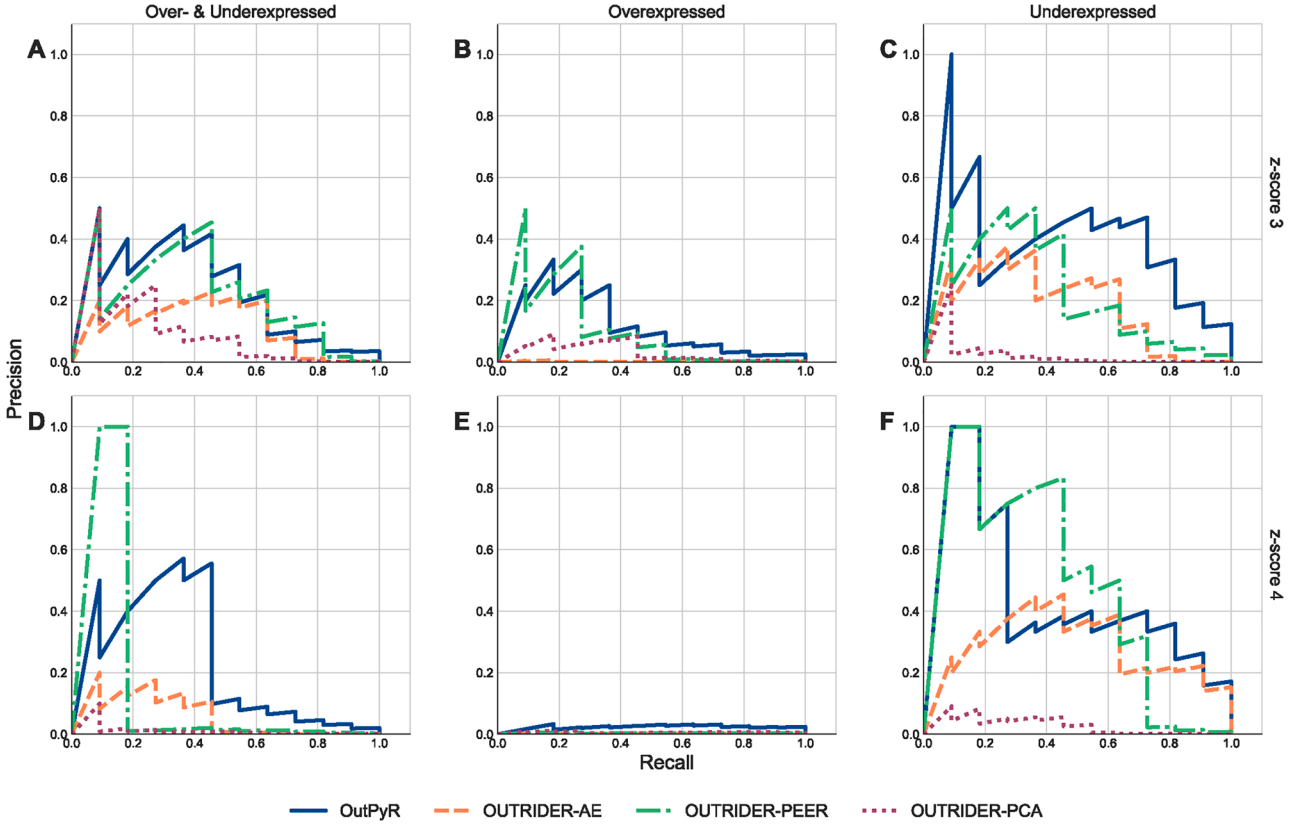


Fig. 3. The ratio of injected outliers and reported outliers (precision) plotted vs. the ratio of all reported injected outliers and all injected outliers (recall) for the datasets derived from the Kremer-10 base dataset. 11 outliers in total were injected in every derived dataset separately. Graphs for all methods we tested are plotted using different lines. Subplots in different rows correspond to the different z-scores of injected outliers. Subplots in different columns correspond to the different types of injected outliers: "Over- & Underexpressed" refers to outliers of which half were overexpressed and half underexpressed, "Overexpressed" refers to outliers that were only overexpressed and "Underexpressed" refers to outliers that were only underexpressed.

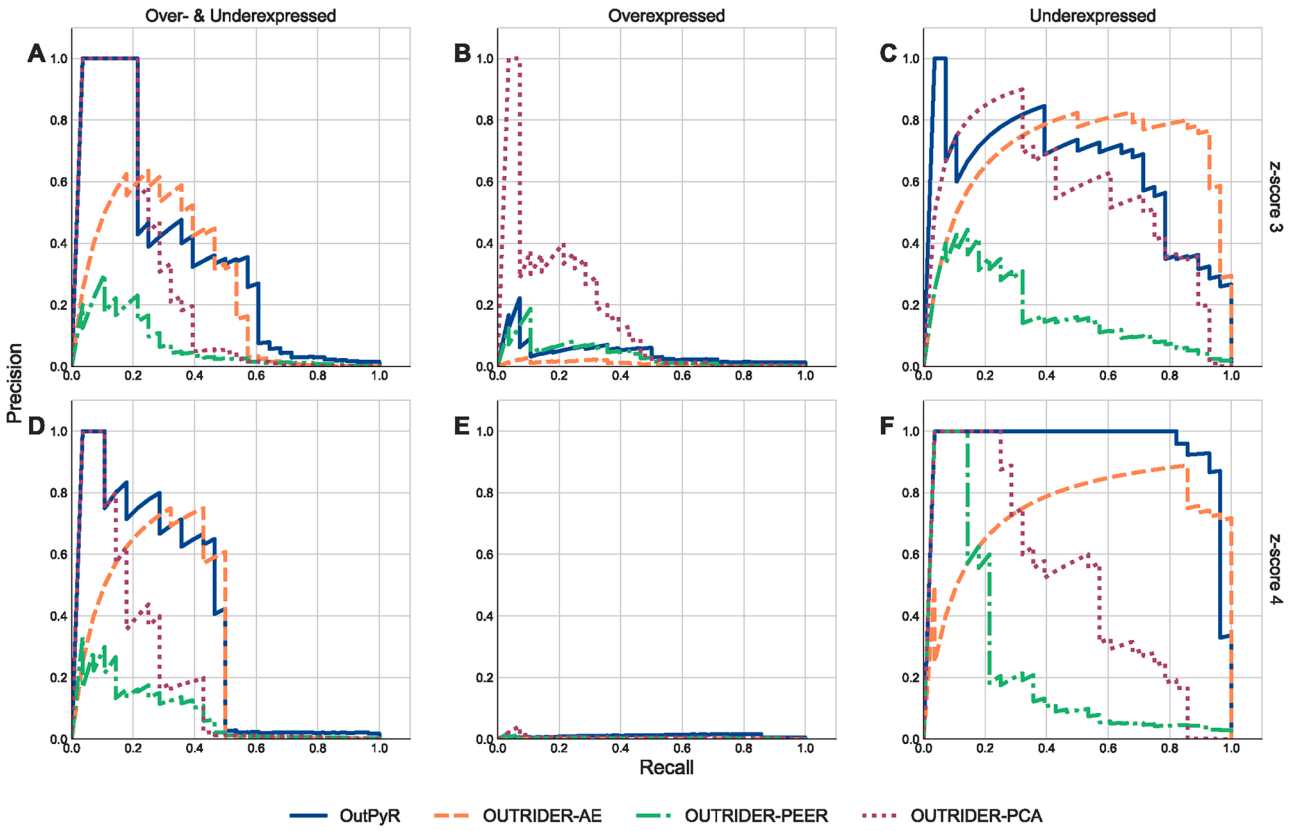


Fig. 4. The ratio of injected outliers and reported outliers (precision) plotted vs. the ratio of all reported injected outliers and all injected outliers (recall) for the datasets derived from the GSE62368-16 base dataset. 28 outliers in total were injected in every derived dataset separately. Graphs for all methods we tested are plotted using different lines. Subplots in different rows correspond to the different z-scores of injected outliers. Subplots in different columns correspond to the different types of injected outliers: "Over- & Underexpressed" refers to outliers of which half were overexpressed and half underexpressed, "Overexpressed" refers to outliers that were only overexpressed and "Underexpressed" refers to outliers that were only underexpressed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
AUC values for PR curves for different methods that were ran on the datasets derived from the GSE62368-16 base dataset, corresponding fully to Fig. 4 (refer to its caption for more detail).

	Over- &Underexpressed	Overexpressed	Underexpressed
z-score 3			
OutPyR	**0.3573**	**0.0428**	**0.6333**
OUTRIDER-AE	0.2590	0.0091	0.6835
OUTRIDER-PEER	0.0643	0.0371	0.1718
OUTRIDER-PCA	0.2695	0.1619	0.5519
z-score 4			
OutPyR	**0.3718**	**0.0107**	**0.9469**
OUTRIDER-AE	0.2937	0.0017	0.7300
OUTRIDER-PEER	0.0773	0.0044	0.2352
OUTRIDER-PCA	0.2110	0.0064	0.5077

with this base dataset OUTRIDER-PEER is the worst performant method, in contrast to the Kremer-10 base dataset, where it was the second best, even slightly outperforming OutPyR on one derived dataset. This might be to the slightly poorer quality of the GSE62368-16 dataset, when compared to the Kremer-10, which can be inspected by plotting the histogram of log-transformed counts for both datasets.

Finally, for all datasets derived from the Simulated-100 base dataset, all methods were performing with an AUC-PR value close to 1, i.e. almost perfectly, so we are omitting PR plots and the AUC-PR table for brevity. This confirms our assumption that, provided there are no confounders, OutPyR can match the performance of other methods.

To summarize: from the results, it is clear that for the majority of derived datasets from the two base datasets with a small number of samples OutPyR outperforms the other methods. Even for the simulated dataset with a large number of samples OutPyR is very competitive, due to the absence of confounders.

In addition to that, due to the data-driven Bayesian nature of the underlying statistical model, OutPyR's results incorporate uncertainty, in an arguably better manner than what the other (non-Bayesian) methods can provide. This can be particularly important in case there is a need for additional downstream analysis.

5. Conclusion and future work

Our model is an example of how recent advances in Bayesian modeling of the NB can be used to outperform existing models for outlier detection on real RNA-Seq datasets with a small number of samples, independent of the underlying organism or tissue type. Although our model currently does not control for confounders which are inherent in RNA-Seq datasets with a large number of samples, the performance of our model on a simulated dataset with a large number of samples shows that, provided confounders are controlled for, our model could potentially match, or even outperform other models. A recently created Bayesian method similar to ours by Dadaneh et al. [11], but tailored to the differential expression setting, showed that NB models introduced by Zhou and Carin [9] can be successfully extended to incorporate confounding factors, albeit known ones. That further suggests that our model could incorporate similar control of confounding factors, perhaps even unknown ones. Moreover, as Bayesian inference is general and extensible, more information could be introduced to the model, such as splicing events and upstream transcript-level analysis results. Furthermore, although the straightforward parallelizability of our model allows

faster execution, perhaps some further speed optimizations are possible, as running times of other models are significantly better (several hours vs. several minutes). Finally, while our *p*-value score is sufficient for the task at hand, it would be useful to see whether other Bayesian approaches to outlier detection could further improve the performance of the model.

Authors' contribution

All of the authors were involved in the production of the manuscript and have approved its final version.

Declaration of competing interest

The authors report no declarations of interest.

References

[1] S.B. Wortmann, D.A. Koolen, J.A. Smeitink, L. van den Heuvel, R.J. Rodenburg, Whole exome sequencing of suspected mitochondrial patients in clinical practice, J. Inherited Metab. Dis. 38 (2015) 437–443.

[2] J.C. Taylor, H.C. Martin, S. Lise, J. Broxholme, J.-B. Cazier, A. Rimmer, A. Kanapin, G. Lunter, S. Fiddy, C. Allan, et al., Factors influencing success of clinical genome sequencing across a broad spectrum of disorders, Nat. Genet. 47 (2015) 717.

[3] G.P. Consortium, et al., A global reference for human genetic variation, Nature 526 (2015) 68.

[4] F. Brechtmann, C. Mertes, A. Matuseviciute, V.A. Yépez, Z. Avsec, M. Herzog, D. M. Bader, H. Prokisch, J. Gagneur, OUTRIDER: a statistical method for detecting aberrantly expressed genes in RNA sequencing data, Am. J. Human Genet. 103 (2018) 907–917.

[5] L.S. Kremer, D.M. Bader, C. Mertes, R. Kopajtich, G. Pichler, A. Iuso, T.B. Haack, E. Graf, T. Schwarzmayr, C. Terrile, et al., Genetic diagnosis of Mendelian disorders via RNA sequencing, Nat. Commun. 8 (2017) 15824.

[6] B.B. Cummings, J.L. Marshall, T. Tukiainen, M. Lek, S. Donkervoort, A.R. Foley, V. Bolduc, L.B. Waddell, S.A. Sandaradura, G.L. O'Grady, E. Estrella, H.M. Reddy, F. Zhao, B. Weisburd, K.J. Karczewski, A.H. O'Donnell-Luria, D. Birnbaum, A. Sarkozy, Y. Hu, H. Gonorazky, K. Claeys, H. Joshi, A. Bournazos, E.C. Oates, R. Ghaoui, M.R. Davis, N.G. Laing, A. Topf, Genotype-Tissue Expression Consortium, P.B. Kang, A.H. Beggs, K.N. North, V. Straub, J.J. Dowling, F. Muntoni, N.F. Clarke, S.T. Cooper, C.G. Bönnemann, D.G. MacArthur, Improving genetic diagnosis in Mendelian disease with transcriptome sequencing, Sci. Transl. Med. 9 (2017).

[7] M.I. Love, W. Huber, S. Anders, Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2, Genome Biol. 15 (2014) 550.

[8] M. Zhou, L. Li, D. Dunson, L. Carin, Lognormal and gamma mixed negative binomial regression, ICML 2012 (2012).

[9] M. Zhou, L. Carin, Negative binomial process count and mixture modeling, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015) 307–320.

[10] E. Pieters, C. Gates, J. Matis, W. Sterling, Small sample comparison of different estimators of negative binomial parameters, Biometrics (1977) 718–723.

[11] S.Z. Dadaneh, X. Qian, M. Zhou, BNP-Seq: Bayesian nonparametric differential expression analysis of sequencing count data, J. Am. Stat. Assoc. 113 (2018) 81–94.

[12] C.I. Bliss, R.A. Fisher, Fitting the negative binomial distribution to biological data, Biometrics 9 (1953) 176–200.

[13] R. Winkelmann, Econometric Analysis of Count Data, Springer Science & Business Media, 2008.

[14] A. Lindén, S. Mäntyniemi, Using the negative binomial distribution to model overdispersion in ecological count data, Ecology 92 (2011) 1414–1421.

[15] M. Zhou, L. Carin, Augment-and-conquer negative binomial processes. Advances in Neural Information Processing Systems, 2012, pp. 2546–2554.

[16] S. Anders, W. Huber, Differential expression analysis for sequence count data, Genome Biol. 11 (2010) R106.

[17] T. Asparouhov, B. Muthén, Prior-posterior predictive p-values, Mplus Web Notes, 2017.

[18] T. Oliphant, NumPy. A Guide to NumPy, Trelgol Publishing, USA, 2006. https://www.numpy.org/.

[19] S.K. Lam, A. Pitrou, S. Seibert, Numba: a LLVM-based Python JIT compiler, Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (2015) 7.

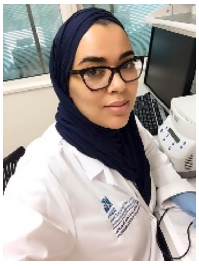
[20] R. Edgar, M. Domrachev, A.E. Lash, Gene expression omnibus: NCBI gene expression and hybridization array data repository, Nucleic Acids Res. 30 (2002) 207–210.

[21] O. Stegle, L. Parts, M. Piipari, J. Winn, R. Durbin, Using probabilistic estimation of expression residuals (PEER) to obtain increased power and interpretability of gene expression analyses, Nat. Protoc. 7 (2012) 500.

Mr. Edin Salkovic is currently pursuing a Ph.D. degree in Computer Science and Engineering at the College of Science and Engineering, Hamad Bin Khalifa University, in the Information and Communication Technologies division. His research interests include algorithms, machine learning, statistics, bioinformatics and more broadly, anything related to computing. He obtained his BSc and MSc degrees from the Faculty of Electrical Engineering, University of Montenegro.

Dr. Mostafa Mansour Abbas is a postdoctoral researcher at the department of imaging science and innovation, Geisinger Medical Center, USA. During his work in producing this manuscript, Dr. Abbas worked at Qatar Computing Research Institute, as a postdoctoral researcher. Dr. Abbas received his Ph.D. degree in computer science from Al-Azhar University, Egypt in October 2015. In 2005, he received his BSc in Mathematics and Computer Science from Al-Azhar University and got his MSc in Computer Science from Al-Azhar University, in 2009. His research interests include algorithms, bioinformatics, metagenomics, data visualization, functional data analysis, and applied machine learning.

Dr. Samir Brahim Belhaouari received a master's degree in telecommunications and Network from the Institut Nationale Polytechnique of Toulouse, France, in 2000, and a Ph.D. degree in mathematics from the Federal Polytechnic School of Lausanne-Switzerland, in 2006. He is currently an associate professor at Hamad Bin Khalifa University, Qatar Foundation, in the Division of Information and Communication Technologies, College of Science and Engineering. In recent years, he also held several Research and Teaching positions at Innopolis University-Russia, Alfaisal university-KSA, University of Sharjah-UAE, university technology PETRONAS-Malaysia, and EPFL Federal swiss school-Switzerland. His research interests include areas from Applied Mathematics, Statistics, and Data analysis as well as Artificial Intelligence, Image & Signal Processing (Biomedical, Bioinformatics, Forecasting, etc.), due to both Mathematics and Computer Science backgrounds.



Mrs. Khaoula Errafii received her Bachelor's degree from Qatar University in 2008, with a major in Biological Sciences and minor in Biomedical Sciences. She immediately pursued her Master's in Biomedical Biotechnology Sciences at Akhawayn University in Ifrane, Morocco. She joined Shafallah Medical Genetics Center in 2011 and worked closely with the autism team in deciphering novel genes associated with Autism Disorder Spectrum and trained on Next Generation Sequencing Platforms (SOLiD4/5, Ion torrent and Illumina). In 2014, Mrs. Khaoula joined Qatar Biomedical Research Institute as a Senior Research Associate in the Genomics Core and was involved in multiple applications including RNA sequencing, Exome Sequencing, Small RNA Sequencing, ChIP Sequencing, Bisulfite Sequencing, and Aptamer Sequencing, in addition to data analysis using CLC workbench. She will be investigating the Role of lncRNA in insulin resistance and non-alcoholic fatty liver disease. She is currently registered as a Ph.D. student in Hamad Bin Khalifah University.

Dr. Halima Bensmail received her Ph.D. from the University of Pierre & Marie Currie (Paris 6) in France, spent one year as a postdoc at the University of Washington (UW), in Seattle and one year at the Fred Hutchinson Cancer Research Center (FHCRC), and then spent two years as a research scientist at the Data Theory Group, University of Leiden, the Netherlands.

She was also an assistant and associate professor of statistical machine learning at the University of Tennessee in statistics in 2001 and an associate professor of biostatistics at the University of Virginia in 2006. Currently, she is a principal scientist at Qatar Computing Research Institute, working in the field of Bioinformatics and Computational Biology, and a joint associate professor at the college of Computer and Science Engineering at Hamad Bin Khalifa University. She published more than 80 peer reviewed papers in high impact journal such as nucleic acids research, genome research, bioinformatics, briefings in bioinformatics, pattern recognition and journal of American statistical association. She received several distinguished awards such as the International Federation of Classification in Society award in 1997 and Qatar Foundation Computing Research award in 2011.