Implementing and evaluating a Gaussian mixture framework for identifying gene function from TnSeq data

Kevin Li
Department of Mathematics, Columbia University, New York, NY 10027, USA
Email: kl2918@columbia.edu

Rachel Chen
Department of Statistics, North Carolina State University, Raleigh, NC 27695, USA
Email: rschen@ncsu.edu

William Lindsey
Department of Mathematics and Statistics, Dordt College, Sioux Center, IA 51250, USA
Email: William.Lindsey@dordt.edu

Aaron Best
Department of Biology, Hope College, Holland, MI 49423, USA
Email: best@hope.edu

Matthew DeJongh
Department of Computer Science, Hope College, Holland, MI 49423, USA
Email: dejongh@hope.edu

Christopher Henry
Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA
Email: chrisshenry@gmail.com

Nathan Tintle
Department of Mathematics and Statistics, Dordt College, Sioux Center, IA 51250, USA
Email: Nathan.Tintle@dordt.edu

The rapid acceleration of microbial genome sequencing increases opportunities to understand bacterial gene function. Unfortunately, only a small proportion of genes have been studied. Recently, TnSeq has been proposed as a cost-effective, highly reliable approach to predict gene functions as a response to changes in a cell's fitness before-after genomic changes. However, major questions remain about how to best determine whether an observed quantitative change in fitness represents a meaningful change. To address the limitation, we develop a Gaussian mixture model framework for classifying gene function from TnSeq experiments. In order to implement the mixture model, we present the Expectation-Maximization algorithm and a hierarchical Bayesian model sampled using Stan's Hamiltonian Monte-Carlo sampler. We compare these implementations against the frequentist method used in current TnSeq literature. From simulations and real data produced by E.coli TnSeq experiments, we show that the Bayesian implementation of the Gaussian mixture framework provides the most consistent classification results.

Keywords: Bayesian; bacteria; genetics.

1. Introduction

1.1. *TnSeq Motivation and Background*

Understanding of bacterial gene function has not kept pace with the rapid acceleration of microbial genome sequencing. Only a small proportion of genes have had their functions experimentally examined and function estimates for unexamined genes have proven inaccurate.[1] Transposon mutagenesis with next generation Sequencing (TnSeq) is a recent method that alleviates this shortcoming in the study of gene function by allowing the simultaneous examination of a wide array of microbial genes.

In TnSeq, a transposon inserts itself into bacterial genes, creating mutants and potentially disrupting bacterial functions. In a library of mutants, DNA is isolated from a section of the bacterial pool as a control group. The remaining section can then be subjected to a test condition. Bacteria whose disrupted genes are essential for growth should decrease in frequency after exposure to the condition. PCR amplifies the DNA sequences bordering the insertions, which are then sequenced and map back to the genome. The change in a gene's fitness can be quantified by comparing the abundance of mutants before and after the test condition. Based on this change, we can then examine the effect of the disrupted genes in specific test conditions.[2] The test conditions under which the mutants suffer fitness penalties are then used to infer gene function.

1.2. *Motivation and New Methods*

The data produced by TnSeq poses classic statistical challenges. First, TnSeq allows researchers to produce fitness measurements for thousands of poorly understood genes across hundreds of experimental conditions.[3] This increase in scale from traditional experimental methods complicates attempts to create a universal decision rule for identifying a gene insertion's fitness condition. The inflated number of experiments also increases the frequency of outliers and edge cases. Furthermore, the magnitude of fitness change varies between gene insertions and experimental noise can be unpredictable. Current practice implements a frequentist statistical significance framework that does not incorporate assumptions inherent in TnSeq and ignores inter-gene information for classification. These shortcomings lead to overly conservative predictions due to overestimates of variance given the unique nature of TnSeq data. The frequentist framework also requires tuning to control the false-positive rate.[1] Finally, the current frequentist framework does not produce an easily interpretable uncertainty estimate for its classifications.

In this paper, we propose modeling the fitness measurements for gene insertions as two-component Gaussian mixture models. We use simulations to show that this framework increases sensitivity to fitness changes while controlling the false discovery rate at acceptable levels. We also provide two distinct methods for fitting these mixture models. The Expectation-Maximization algorithm is a widely accepted method for fitting such models. We also propose a hierarchical Bayesian approach in which we model the parameters of our Gaussian mixture as random variables with prior distributions. This strategy allows us to incorporate inter-gene information and prior

knowledge of the TnSeq method as soft constraints on our estimates. We will ultimately compare the performance of these methods against the current frequentist framework.

2. Methods

2.1. *TnSeq Experimental Data*

We present a model of transposon sequencing in which only one strain of each gene insertion is counted. A control count is first obtained for each gene insertion by examining its growth under a condition known to have no effect on bacterial survival. Given n insertions, and m experimental conditions, TnSeq then produces an n x m matrix where each row represents an insertion, and each column contains fitness counts for an experimental condition. Thus if we denote this matrix $\boldsymbol{C}$, the matrix element $C_{i,j}$ represents the fitness counts for gene insertion i under experimental condition j. The final fitness measurement for each insertion under each experimental condition is calculated via the equation:

$$f = \log(n_1 + 1) - \log(n_0 + 1)^1 \quad (1)$$

where n_1 is the cell is count under the experimental condition and n_0 is the cell count under the control condition. The total variance of the gene's fitness value is calculated via:

$$V = \frac{\frac{1}{1+n_1} + \frac{1}{1+n_0}}{\ln(2)^2} \quad (2)$$

This variance assumes Poisson noise and is later used for calculating a t-like statistic for the frequentist method.[3]

2.2. *Mixture framework*

We apply our novel Gaussian mixture framework to the n x m matrix representing the fitness measurements of each insertion. We denote this matrix E. The matrix element $E_{i,j}$ represents the fitness measurement of the ith insertion under the jth experimental condition. We wish to identify each $E_{i,j}$ as the result of a neutral or deleterious experimental condition. Fitness measurements under deleterious experiments indicate that the mutant's disrupted gene is relevant to some function. Note that whether an experiment is neutral or deleterious depends on the mutant. To evaluate the likelihood of our label estimate, we propose modeling each row of E as a two-component Gaussian mixture. We would like the first mixture component to capture experiments in which fitness is unaffected such that $E_{i,j} | \, unaffected \sim \mathrm{N}(\mu_{i,0}, \sigma_i)$. The second component captures experiments in which fitness is affected such that $E_{i,j} | \, affected \sim \mathrm{N}(\mu_{i,1}, \sigma_i)$. Due to the nature of TnSeq data, we expect $\mu_{i,0}$ to be close to 0 and $\mu_{i,1}$ to be negative. This second component mixture exists because groups of experiments deliberately test similar bacterial functions and therefore produce similar fitness changes. This aspect of TnSeq also allows us to assume variances for the mixtures. We therefore define the likelihood of row i of the matrix as:

$$E_j \sim \theta\phi(\mu_{i,0}, \sigma_i) + (1 - \theta)\phi(\mu_{i,1}, \sigma_i) \quad (3)$$

where ϕ is the pdf of a normal distribution, and θ is the proportion of experiments in which the mutant is unaffected.

This framework can generate a probability that any fitness measurement is the product of a deleterious experiment. This probability that fitness measurement $E_{i,j}$ is produced by a deleterious experiment is defined as:

$$a_{i,j} = \frac{\phi(E_{i,j}|\mu_{i,1},\sigma_i)}{\phi(E_{i,j}|\mu_{i,1},\sigma_i) + \phi(E_{i,j}|\mu_{i,0},\sigma_i)} \quad (4)$$

This value is simply the density of the fitness-affected mixture divided by the total density. We classify the $E_{i,j}$ as the result of a deleterious experiment if $a_{i,j}$ is greater than .5.

2.3. *Classification methods*

2.3.1. *Novel method – EM*

An accepted statistical method for estimating unobserved labels under a Gaussian mixture likelihood is the Expectation-Maximization (EM) algorithm.[4] The EM algorithm iteratively fits a Gaussian mixture model by constructing a monotonically increasing sequence of lower bounds for the log likelihood function. We allow the mixture that is closest to zero represent the experiments that do not affect mutant fitness. The selection of a two-component mixture model as opposed to classifying all experiments as neutral is based upon the commonly used Bayesian Information Criterion (BIC).[4] We fit a two-component mixture model if it has the lower BIC compared to a simple Gaussian model. Otherwise we assume the insertion's fitness values are all produced from neutral experiments. We make this assumption as it is biologically improbable that all or even most experiments will harm fitness. We implement the algorithm through the R package Mclust.[5]

2.3.2. *Current method – t-statistic*

The current method in TnSeq literature leverages the estimated variance of fitness measurements to calculate the statistical significance of fitness changes.[1,3] It calculates a t-like statistic:

$$t = \frac{f}{\sqrt{.1+V}} \quad (5)$$

where .1 is a small regularizing constant, and V is the variance estimate for the insertion's fitness measurements as described in section 2.1. An experiment is considered deleterious if $|t| > 4$ and $|f| > .5$. This statistic is assumed to have a standard normal distribution[3].

The frequentist approach does not provide an easily interpretable probability for label estimates. For the sake of comparison, we define $a_{i,j}$ for the t-statistic classifier as:

$$a_{i,j} = 1 - \phi(\mathrm{t}) \quad (6)$$

where $\phi(\mathrm{t})$ represents a standard normal cdf. This expression is simply one minus the probability that we obtain a statistic as extreme as t under the assumption of no fitness change. This $a_{i,j}$ can be interpreted as the confidence of the classification.

2.3.3. *Bayesian hierarchical model*

We finally adopt a Bayesian hierarchical modeling framework for fitting a Gaussian mixture model. The hierarchical approach assumes that model estimates for individual insertions are conditional on some unobserved parameters shared across all insertions. We denote these parameters as hyper-parameters. The hyper-parameters have their own hyper-prior distribution which are estimated from all insertions in the data set. This strategy of conditioning estimates for individual genes on these sample-wide hyper-priors achieves a pseudo pooling effect. The hyper-prior distributions leverage across-gene information to weaken the influence of outliers and increase sensitivity to small mixture probabilities.[6]

We fit our hierarchical Bayesian model in the R interface to the probabilistic programming language, Stan.[7] Stan allows fast, out-of-the-box fitting of Bayesian models without the computation of the conditional parameter distributions or tuning variables.[8] We later provide strategies for partitioning our data set in order to speed computations and allow parallelization.

We use the following priors in our Bayesian model. We give $\mu_{i,0}$ prior distribution $N(0,\delta)$. The location of the prior is fixed at 0 to reflect the experiments' null effect on fitness. The scale of the prior is modeled by hyper-parameter δ with a $InverseGamma(20,1)$ prior. The parameters of the prior and hyper-prior reflect our strong belief that neutral experimental conditions should consistently produce fitness measurement close to zero plus or minus some error common to the mutants in the sample. The hierarchical structure on δ estimates this error from the mutants in sample. We default to the Inverse Gamma distribution for its conjugacy properties.

We constrain $\mu_{i,1}$ to be negative by the assumptions of transposon sequencing[3]. We give $\mu_{i,1}$ prior distribution $N(-3,\lambda)$. The mean of the prior is fixed at a negative real to prevent degenerate label switching with the first mixture. We choose –3 because it represents a moderate change in fitness.[3] The choice of –3 specifically as compared to any other reasonably small negative real is unimportant due to the choice of the un-informative scale prior λ, which has a prior distribution that is uniform across all positive real numbers. The uninformative prior allows λ to become arbitrarily large as the data demands.[6] The data dominates the value of λ in this the model and reflects our lack of prior information of the true distribution of the fitness measurements. We model λ as a hierarchical parameter to prevent outliers from overly affecting $\mu_{i,1}$ estimates and to increase sensitivity to departures from zero. Although $\lambda's$ prior is not a proper distribution, the joint distribution of $\mu_{i,1}$ and λ is proportional to an inverse gamma distribution, which ensures that the integral of the posterior distribution is finite.[6]

We give θ_i a beta prior with symmetric uniform hyper-priors for its flexibility over the [0,1] interval as well as by the methods of Disselkoen 2016.[10] The hierarchical structure on theta resists outliers and prevents overfitting on single mutants.

We give σ_i a $Cauchy(0,5)$ prior. The prior is weakly informative by allowing for large values in the heavy tails of the distribution. This reflects our weak confidence that most variances should be reasonably small with a few exceptions. We select the Cauchy distribution by recommendation of Gelman 2006.[6]

2.3.4. *Data partitioning for the Bayesian model*

Markov Chain Monte Carlo sampling methods are computationally intensive for large data sets and sensitive to the true parameter diversity of the data. Therefore, we propose fitting the Bayesian model separately on partitions of the data that maximize within-partition similarity. Partitioning the data speeds sampling and makes the computations easily parallelizable. To maximize the similarity of genes within the partitions, we use the k-means clustering algorithm on the normalized log-fitness vectors of the genes. This clustering is equivalent to clustering the gene insertions by angular distance or correlation of their fitness measurement vectors.[11] For computational considerations in our simulation scenarios, we currently set the number of clusters such that there are on average 20 genes per partition.

2.4. *Simulation*

To evaluate the performance of our classifier, we simulate sets of insertions and fitness measurements under a fixed number of experiments. We simulate different scenarios where we vary the proportion of insertions that affect fitness under any experimental conditions. In this study we simulate cases where 0%, 25%, 50%, 75%, and 100% of insertions affect fitness. Simulating these distinct scenarios is important because the hierarchical Bayesian model estimates parameters of individual insertions from a parameter distribution estimated over the entire data set. For each scenario, we simulate 100 separate sets of 100 gene insertions to test the performance of the three methods. We note that the Bayesian model is fit separately on each of these sets of 100.

We adopt the following algorithm for simulating bacterial counts and fitness measurements. First, across all gene insertions in a set we define a probability δ that a gene insertion affects fitness under any experimental conditions. We then proceed through the following steps to draw the mutant counts.

For each gene insertion i:

- Draw parameter τ from gamma distribution $gamma(\hat{\alpha}, \hat{\beta})$, in which $\hat{\alpha}$ and $\hat{\beta}$ are the gamma parameter maximum likelihood estimates from the experimental control counts of E.coli mutants provided by Price 2018.[1] This distribution is not significantly different from the empirical control count distribution by the Kolmogorov-Smirnov test ($p > .3$).
- Draw the simulated control count from $poisson(\tau)$. Denote $poisson(\tau)$ as the neutral distribution.
- Choose a fitness factor, F from $uniform(.15, .95)$. We denote $poisson(\tau * F)$ as the affected distribution.
- With probability δ, draw θ from $uniform(.3, .95)$. Else set θ to be 1. θ is the probability that an experiment does not affect mutant fitness.
- For every experiment, draw a count from the control distribution with probability θ. Otherwise draw a count from the deleterious distribution.

Pre-fixed simulation distribution parameters were chosen to account for all reasonable biological possibilities. Uniform distributions were chosen by the maximum entropy principle to reflect our uncertainty surrounding the true distribution of real data sets.[12] The fitness measurements and t-

statistics for each experiment can be calculated for each gene insertion using the control count and Eq. (5) and (6).

2.5. *Real data*

We apply our methods to Escherichia coli BW25113 TnSeq data provided by Price 2018.[1] They examine the fitness of E.coli mutants produced by 3789 distinct gene insertions. They subjected mutants to 162 experimental conditions. We apply the EM and Bayesian classifiers to the provided 3789 x 162 matrix of fitness measurements. We use the t-statistic classification results provided by Price 2018.

3. Results

We evaluate the following performance metrics for each of the classification methods. We use the mean of the posterior distribution draws of the Gaussian mixture parameters to define the Bayesian model.[6] We use the following metrics to evaluate the performance of the classifiers.

3.1. *Metrics*

Define the true label for fitness measurement $E_{i,j}$ as $l_{i,j}$, taking value 0 if $E_{i,j}$ is the result of a neutral experiment and value 1 if $E_{i,j}$ is the result of a deleterious experiment. Let the predicted label for $E_{i,j}$ be $\widehat{l_{i,J}}$. Similarly $\widehat{l_{i,J}}$ is 0 if the classifier labels the fitness measurement as a neutral result and 1 if the classifier labels the measurements as a deleterious result.

3.1.1. *Classification rate*

The classification rate is the raw percentage of experiments that the model classifies correctly. Therefore the Classification Rate for the ith insertion would be:

$$CR_i = \frac{\sum_{j=1}^{m} I_{\{l_{i,j} = \widehat{l_{i,J}}\}}}{m} \tag{7}$$

where $I_{\{l_{i,j} = \widehat{l_{i,J}}\}}$ is an indicator function that takes value one if $l_{i,j} = \widehat{l_{i,J}}$ and zero otherwise.

3.1.2. *False positive rate*

The false positive rate is the Type I error. It is the percentage of neutral experiments that the model incorrectly classifies as deleterious. In ideal scenarios, this value should be low. The False Positive Rate for the ith mutant is therefore:

$$FP_i = \frac{\sum_{j=1}^{m} I_{\{\widehat{l_{i,J}} = 1 \ \wedge \ l_{i,j} = 0\}}}{\sum_{j=1}^{m} I_{\{l_{i,j} = 0\}}} \tag{8}$$

where $I_{\{\widehat{l_{i,J}} = 1 \wedge l_{i,j} = 0\}}$ is an indicator function that takes value one if $\widehat{l_{i,J}} = 1$ and $l_{i,j} = 0$.

3.1.3. *Positive classification rate*

The positive classification rate is the percentage of deleterious experiments that the model correctly classifies as deleterious. In ideal scenarios, this value should be low. The positive classification rate for the ith insertion is therefore:

$$PC_i = \frac{\sum_{j=1}^{m} I_{\{\widehat{l_{i,j}}=1 \ \wedge \ l_{i,j}=1\}}}{\sum_{j=1}^{m} I_{\{l_{i,j}=1\}}} \tag{9}$$

where $I_{\{\widehat{l_{i,j}}=1 \ \wedge \ l_{i,j}=1\}}$ is an indicator function that takes value one if $\widehat{l_{i,j}} = 1$ and $l_{i,j} = 1$. Otherwise the function takes value 0.

3.1.4. *Cross entropy*

We measure the accuracy of our probabilistic estimates using cross-entropy. The cross entropy for the classification of the ith insertion is defined as:

$$CE_i = -\sum_{i=1}^{m} \widehat{l_{i,j}} \log(a_{i,j}) + (1 - \widehat{l_{i,j}}) \log(1 - a_{i,j}) \tag{10}$$

Cross entropy is a common loss function for evaluating classifiers that produce probability estimates ranging from 0 to 1.[4] The greater the difference between the true and model classifications, the higher the cross entropy will be. For example, if the true label is 1 and $a_{i,j}$ is 0, then the classifier performs badly and the cross entropy will be high. However, a better probability estimate of .49 will correspond to a lower cross entropy value.

3.2. *Simulation Results*

We simulate the scenarios in which 0%, 25%, 50%, 75%, and 100% of gene insertions are affected by experimental conditions. For each scenario, we simulate one hundred sets of one hundred insertions. On each set, we separately fit the Bayesian model on a single Markov chain with 1000 warm-up iterations and 1000 sampling iterations. We take the posterior means of the Gaussian mixture parameters to define our Bayesian classification model.

The simulation results demonstrate that the three methods provide identical classifications for 64% of the 50,000 simulated genes. These classifications produced models with over a 98% classification rate. This is expected as the simulated fitness values for many gene insertions are either obviously unimodal or clearly clustered into two groups. In an additional 10% of cases, all the classifiers achieved at least a 90% classification rate. Thus, the entire simulation population does not tell us much about the relative performance of the classifiers on difficult classification problems.

We proceed to examine only the 26% of the cases where the t-statistic, EM algorithm, and Bayesian classifier do not provide identical classifications and at least one of the classifiers fails to achieve an 90% classification rate. We call this the difficult subset.

We see in Figure 1 and Table 1, Column 3 that the t-statistic performs relatively well when the proportion of affected mutants is small (0%, 25%). For higher proportions, we see that the t-

statistic's performance deteriorates in the second and third classification quantiles relative to the other methods. On the other hand, the EM algorithm performs well when the proportion of affected mutants is large (75%, 100%). The EM algorithm suffers in performance for the first and second and third quantiles, especially for lower proportion (0%, 25%, 50%). Only the Bayesian model demonstrates consistent behavior across proportions and quantiles, outperforming both the other methods except when the proportion of affected mutants is 0%.

We see from the positive classification rate in Figure 1 and Column 4 in Table 1 that the t-statistic is by far the least sensitive to changes in fitness and therefore has the lowest positive classification rate. The Bayesian algorithm provides a vast improvement on the positive classification rate. But the EM algorithm overall provides the most sensitive classification results, especially true at lower proportions. The EM algorithm achieves this sensitivity by incurring higher false positive rates. The Bayesian algorithm does not suffer from as high false positive rates. The t-statistic expectedly maintains the lowest false positive rate. Therefore, we see that the Bayesian algorithm achieves higher and consistent classification by compromising between sensitivity of the EM algorithm and the conservatism of the t-statistic.

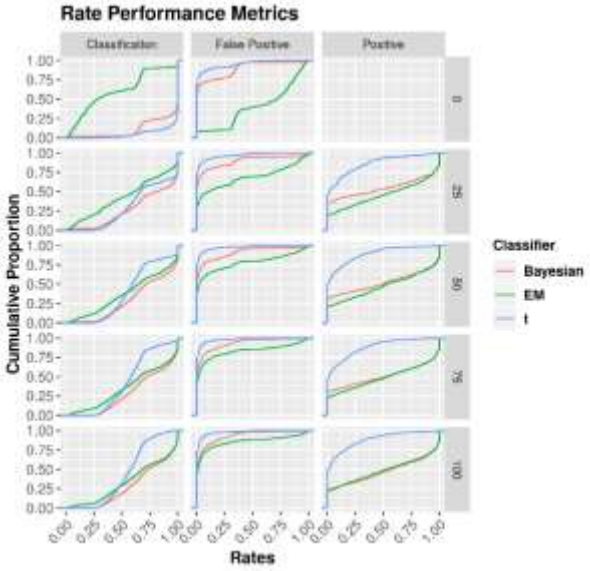

Fig. 1. Cumulative distributions of classification, false positive and positive classification rates on the difficult subset of simulated gene insertions. Columns indicate the metric displayed, and rows indicate the proportion of mutants affected in each mutant set.

Table 1. Mean Classification Rate, Positive Classification Rate, False Positive Rate and Cross Entropy for Classifiers

	2. % Affected	3. Mean CR	4. Mean PCR	5. Mean FPR	6. Mean CE
Bayesian	0	.90	NA	.08	65.13
	25	.75	.57	.07	242.27
	50	.72	.60	.06	279.49
	75	.73	.61	.05	301.30
	100	.73	.64	.05	288.97
EM	0	.40	NA	.33	305.95
	25	.58	.65	.20	313.95
	50	.63	.64	.14	320.05

	75	.66	.63	.10	334.89
	100	.68	.63	.09	334.79
T	0	.95	NA	.05	171.68
	25	.72	.22	.03	267.99
	50	.62	.21	.02	308.60
	75	.59	.22	.02	339.06
	100	.57	.21	.02	343.20

From Figure 2 and Column 6 in Table 1, we see that the Bayesian and EM method produce smaller cross entropy losses for most classifications compared to the t-statistic. However, we also see that the Bayesian and EM methods have fatter tails, indicating a significant subset of cases where the two methods provide poor probability estimates. From Table 1 Column 6, we see that from an entropy standpoint, the Bayesian algorithm outperforms the EM algorithm and t-statistic on average in every scenario. Therefore, we can see that the Bayesian algorithm provides accurate probabilistic estimates more consistently.

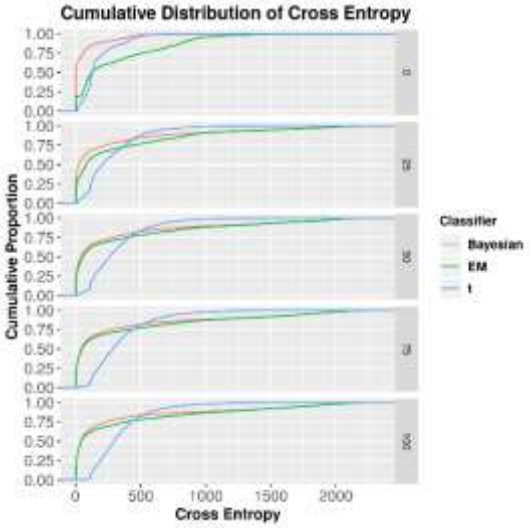

Fig. 2. Cumulative distribution of Cross Entropy Distributions. Cross entropy values near zero indicate accurate probability estimates of classification confidence.

3.3. *Comparisons on real data*

We apply the EM and Bayesian methods to the fitness measurements from the real E.coli data (see section 2.5 for details). For the t-statistic, we use the classifications produced by the work of Price 2018[1]. The t-statistic is by far the most conservative, identifying 496 genes as important to some examined bacterial function. The EM algorithm identifies 1322 genes and the Bayesian method identifies 1786 genes. Of the 496 genes identifies by the t-statistic, the EM algorithm shares 137 identifications. The Bayesian algorithm shares 455 gene identifications with the t-statistic. In Figure 3 we present three examples where each of the three classifiers fails to identify a gene's function where the other two are successful.

The mutant from the insertion into gene b0002 is an instance where the t-statistic does not identify a gene where the Bayesian model and EM algorithm do. The EM algorithm and Bayesian

model provide the same classifications for b0002, while we see that the t-statistic fails to identify any changes in fitness. This failure of the t-statistic behavior can be attributed to the clear existence of two separate mixture components with separate variances. The t-statistic calculates the variance from both mixtures and therefore underestimate significance.

We next give an example where the EM algorithm does not identify a gene (b0008) that the t-statistic and Bayesian model identify. In this case in Figure 3, we see that the BIC does not detect the presence of two mixtures and our implementation of the EM algorithm and therefore assumes no changes in fitness. We have considered changing the BIC threshold for two-mixture selection, but any changes resulted in much worse simulation results.

Now we examine the insertion on b1198. This insertion belongs to the 16 cases where the Bayesian algorithm does not identify a gene that the EM algorithm and t-statistic both identify as important to some function. In each of these cases the EM algorithm and t-statistic identify a positive fitness change from a gene insertion. This is improbable, as a gene deletion should not increase fitness. The Bayesian model's priors explicitly prevent this classification result.

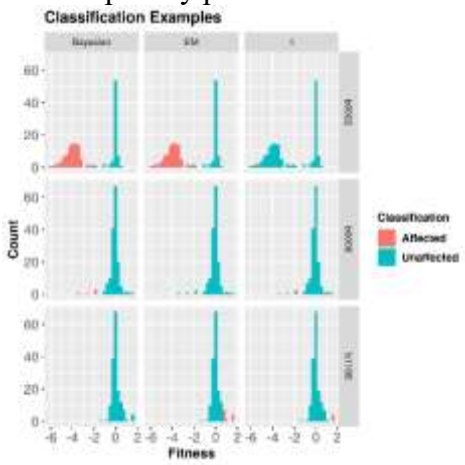

Fig. 3. Classifications for mutants produced by insertions into genes b0002, b0008, and b1198. Bars represent counts of fitness measures under various experimental conditions.

3.4. *Software*

R scripts for the implementation of the classification methods can be found at: http://www.nathantintle.com/supplemental/TnSeqRFunctions.R

4. Discussion

We have presented a two-component Gaussian mixture framework for classifying experimental effects on mutant fitness. This framework provides an alternative to the current frequentist framework. We have shown how the frequentist approach produces conservative estimates due to its estimation of a large variance encompassing all of mutant's fitness values despite the existence of two smaller distributions. The mixture framework addresses this problem by estimating the smaller variances of two smaller components.

Furthermore, simulations demonstrate that the Bayesian classifier generally outperforms the EM algorithm. By incorporating reasonable priors and exploiting a hierarchical structure, the Bayesian

model leverages inter-gene information to provide a compromise between the sensitivity of the EM algorithm and the conservatism of the t-statistic. The Bayesian model's performance is also nearly invariant under the proportion of mutants affected. Given high uncertainty about the genes studied, the Bayesian model should be the model of choice for classification.

On the real E.coli data, we see that the Bayesian classifier is able to identify all the genes with negative fitness changes that the t-statistic identifies. The Bayesian classifier demonstrates significantly more sensitivity to fitness changes while maintaining consistency with the t-statistic. This behavior is distinct from the EM algorithm, which has significantly different identifications and seems to be insensitive to lower mixing probabilities. Still, both mixture classifiers are able to identify multi-functional genes at a much higher rate than the t-statistic.

Despite the promise of the methods proposed, further work is necessary to validate our approach on additional datasets for which true fitness changes are known. We note that while the performance of the Bayesian classifier is generally better than the EM algorithm, the computational time of the Bayesian classifier may be prohibitive in some cases (e.g., it takes 30.8 hours with 5 cores to fit the E.coli 3789 x 162 fitness measurement matrix). Further work will seek to enhance the computational time of the Bayesian classifier, though we acknowledge that it may never by as 'instantaneous' as the EM algorithm or t-statistic approaches.

The success of the Bayesian classifier encourages further expansion of the hierarchical model structure. Hyper-prior distributions can be defined to account for multiple strains per mutant or even genes across bacteria. Covariance priors can be added to leverage co-fitness information[1] to make more robust classifications. Further development of the hierarchical structure will allow rich probabilistic models of gene function and fitness. In the meantime, we suggest use of the proposed Bayesian classifier to improve classification accuracy of changes in mutant fitness.

Acknowledgments The authors of this project were partially supported by NSF-MCB-1715211.

References

1. M. N. Price *et al.*, *Nat.* **557**, 503—509 (2018).
2. T. van Opijnen, K. L. Bodi and A. Camilli, *Nat. Methods* **6**, 767—772 (2009).
3. K. M. Wetmore *et al.*, *mBio* **6**, e00306-15 (2015).
4. T. Hastie, R. Tibshirani and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction* (2nd ed.). New York, NY: Springer (2009).
5. L. Scrucca, M. Fop, T. B. Murphy and A. E. Raftery, *The R Journal* **8**, 205—223 (2016).
6. A. Gelman, J. B. Carlin, H. S. Stern and D. B. Rubin, *Bayesian data analysis* (3rd ed.). Taylor & Francis (2013).
7. Stan Development Team, *RStan: the R interface to Stan* **2.16.2** (2017).
8. B. Carpenter *et al.*, *Journal of Statistical Software* **76**, (2017).
9. A. Gelman, *Bayesian Anal.* **1**, 515—534 (2006).
10. C. Disselkoen *et al.*, *Front. Microbiol.* **7**, 1191 (2016).
11. S. Zhong, *International Joint Conference on Neural Networks* **5**, 3180—3185 (2005).
12. W. Boomsma, J. Ferkinghoff-Borg and K. Lindorff-Larsen, *PLoS Comput. Biol.* **10**, e1003406 (2014).