2014-04-29

RVD2: An ultra-sensitive variant detection model for low-depth heterogeneous next-generation sequencing data

Yuting He
Worcester Polytechnic Institute

RVD2: An ultra-sensitive variant detection model for low-depth heterogeneous next-generation sequencing data

by

Yuting He

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Biomedical Engineering

by

May 2014

APPROVED:

Professor Patrick J. Flaherty, Major Thesis Advisor

Professor Dirk R. Albrecht, Committee member

Professor Andrew C. Trapp, Committee member

Abstract

Motivation: Next-generation sequencing technology is increasingly being used for clinical diagnostic tests. Unlike research cell lines, clinical samples are often genomically heterogeneous due to low sample purity or the presence of genetic subpopulations. Therefore, a variant calling algorithm for calling low-frequency polymorphisms in heterogeneous samples is needed.

Results: We present a novel variant calling algorithm that uses a hierarchical Bayesian model to estimate allele frequency and call variants in heterogeneous samples. We show that our algorithm improves upon current classifiers and has higher sensitivity and specificity over a wide range of median read depth and minor allele frequency. We apply our model and identify twelve mutations in the PAXP1 gene in a matched clinical breast ductal carcinoma tumor sample; two of which are loss-of-heterozygosity events.

Acknowledgements

I would like to begin by expressing my deepest gratitude to my major advisor, Professor Patrick Flaherty, for all the support and guidance from the onset of my experience at Worcester Polytechnic Institute. He has rich fund of knowledge in machine learning, statistics, bioinformatics and cancer genomics, fields I have been so much fascinated by and would like to dedicate my lifelong research to. I owe unending thanks to him for taking me into his lab as his first graduate student, helping me get off to an amazing project and supporting me with his sustained patience, optimism, confidence, expertise and guidance throughout this entire project. I have greatly benefited from his help in many other ways as well such as building up confidence, confronting personal shortcomings, fighting pressure, improving technical communication skills in order to achieve overall success in my life. He is the best advisor and most thoughtful friend I can possibly have the fortune to know in my life and I will never forget him.

I would like to thank the rest of my thesis committee: Professor Andrew Trapp and Professor Dirk Albrecht, for their encouragement and insightful comments. Their input and assistance was vital to the completion of this project and my thesis.

Special thanks to Professor Yitzhak Mendelson, who took me on-board and taught me the first research skills during my first lab rotation at WPI. His rigorousness in research, teaching and life keeps spurring me to pursue high standard outcomes. I have Professor Ki Chon to thank for all his group meetings I attended, which have been a valuable source of knowledge, inspiration and friendship. My appreciation also goes to Professor Marshal Rolle, who was always there offering me constructive comments and warm encouragement when I was struggling with making decisions.

My life at WPI was enriched by a circle of good friends in and out of the cam-

pus. Many thanks to Jiaojiao Li, Linglong Zhu, Fan Zhang, Yanni Wang, Hachem Saddiki, Hugo Fernando Posada Quintero, Bersain Reyes, Duy Dao, Dr. Natasa Reljin, Jeffrey Bolkhovsky, Kyra Burnett, Joshua Harvey and other friends whom I've had the honor of meeting and knowing. I am tremendously thankful for their company and friendship.

I have my special thanks to my boyfriend, Heng Zhao, who has been extraordinarily considerate and supportive in the last four years. Twelve hours of jet lag and 11K kilometers apart for two years is a enormous challenge to any relationship, and I'm so grateful that he has been together with me through all the ups and downs. Finally and most importantly, my families on the other side of earth will constantly be the original force which keeps me moving forward.

To my parents Guoying Yi and Fude He.

Contents

List of Figures

Chapter 1

Introduction

1.1 Next generation sequencing

The fundamental goal of genetics is to associate genotypes with observed phenotypes. In order to observe the genotype, DNA must be sequenced. DNA sequencing technology has numerous application fields such as diagnostic, molecular biology, evolutionary biology, ecology, epidemiology and virology research.

The conventional capillary-based sequencing technology, or Sanger sequencing technology is the mainstream technology for from late 1970s till late 1990s. Next-generation sequencing(NGS) method made its debut in the mid to late 1990s [60]. The real sequencing revolution came along with the sequencing-by-synthesis technology from 454 Life Sciences5 and the multiplex polony sequencing protocol developed in George Church's lab [44, 63, 64]. The major revolution in next-generation sequencing lies in the ability to process millions of sequence reads in parallel rather than 96 at a time in first-generation sequencing[43]. The cost of sequencing in parallel is that each individual read is short. The high-throughput attribute dramatically enriches the data we are able to obtain at low cost with minimum time. Quail et al. [54] shows that protocol and platform engineering improvements have enabled the

generation of 1×10^9 bases of sequence data in 27 hours for approximately $1000. Given highly functional statistical tools, next-generation technology will grant us the ability to deeply interpret polymorphisms and ultimately better understand the live beings.

Three major types of NGS platforms have been commercially available today, including Roche/454, Illumina/Solexa and SoLiD. There are also several 3rd generation, or next-next-generation sequencing systems in the market, such as HeliScope, Ion Torrent, PacBio and Starlight. These platforms differ in many aspects such as library preparation, amplification and sequencing method, accuracy, instrument purchase and running cost, and thereby primary applications. Up to now, no single platform is able to completely replace any other platforms, as they are often complementary in shortcomings. Currently, Illumina is most broadly utilized today due to the lowest unit running cost [27, 63].

Next-generation sequencing (NGS) technology has enabled the systematic interrogation of the genome for a fraction of the cost of traditional assays [36]. NGS is increasingly being used as a general platform for research assays for methylation state [38], DNA mutations [10], copy number variation [1], promoter occupancy [52] and others [57]. NGS diagnostics are being translated to clinical applications including noninvasive fetal diagnostics [35], infectious disease diagnostics [7], cancer diagnostics [49], and human microbial analysis [11].

Increasingly, NGS is being used to interrogate mutations in heterogeneous clinical samples. For example, NGS-based non-invasive fetal DNA testing uses maternal blood sample to sequence the minority fraction of cell-free fetal DNA [17]. Infectious diseases such as HIV and influenza may contain many genetically heterogeneous sub-populations [20, 24]. DNA sequencing of individual regions of a solid tumor has revealed genetic heterogeneous within an individual sample [49].

1.2 Mutation detection

Currently, the primary statistical tools for calling variants from NGS data are optimized for homogeneous samples. Samtools/bcftools and GATK uses naive Bayesian decision rule to call variants [14, 39]. GATK involves more sophisticate pre- and post-processing steps wherein the genotype prior is fixed and constant across all loci and the likelihood of an allele at a locus is a function of the phred score [46].

Recently, researchers have developed algorithms to call low-frequency or rare variants in heterogeneous samples. Yau et al. [73] developed a Bayesian framework which can model the normal DNA contamination and intra-tumor heterogeneity by parameterizing the normal genotype cell proportion at each SNP. VarScan2 combines algorithmic heuristics to call genotypes in the tumor and normal sample pileup data and then applies a Fisher's exact test on the read count data to detect a significant difference in the genotype calls [37]. Strelka uses a hierarchical Bayesian approach to model the joint distribution of the allele frequency in the tumor and normal samples at each locus [62]. With the joint distribution available, one is able to identify locations with dissimilar allele frequencies. muTect uses a Bayesian posterior probability in its decision rule to evaluate the likelihood of a mutation [9]. RVD uses a hierarchical Bayesian model to capture the error structure of the data and call variants [13, 20]. However, that algorithm requires a very high read depth to estimate the sequencing error rate and call variants.

Several studies have compared the relative performance of these algorithms. Spencer et al. [66] demonstrated that VarScan-somatic performed the best comparing with SAMtools, GATK and SPLINTER in detecting minor allele frequencies (MAFs) of 1% to 8%, with >500 coverage required for optimal performance. However, Spencer et al. [66] also highlighted the fact that VarScan2 yielded more false

positives at high read depth. Stead et al. [68] showed that VarScan-somatic outperformed Strelka and had performance on-par with muTect in detecting a 5% MAF for read depths between 100 and 1000.

1.3 Graphical Model

A graphical model is a graph representation of conditional dependence between random variables with different probability distributions. In a graphical model framework, a shaded node represents an observed random variable, an unshaded node represents an unobserved or latent random variable and a directed edge represents a functional dependency between the two connected nodes. A rounded box or "plate" represents replication of the nodes within the plate. Graphical model connects graph theory and probability theory in a visual, intuitive and natural way, which greatly facilitates statistical inference, such as computing marginal and conditional probabilities of interest [33].

Graphical model can be divided into two groups in general, directed graphical models and undirected graphical models. In an undirected model, also known as Markov random field or Markov network, there is no direction arrow in the edges between two nodes. Undirected graphical models are more applicable to areas such as image processing [41, 74], computer vision [40], where internal relationships are better described by non-causal relationships. On the other hand, directed graphical models are intensively used as representation for Bayesian hierarchical models due to its ability to represent hierarchical latent structures intuitively [33].

A good example of graphical model application in Bayesian statistics is Latent Dirichlet Allocation(LDA) model proposed by Blei et al. [6]. The model is shown in Figure 1.1. LDA is a generative hierarchical model, in which documents are

modeled as a mixture of a set of topic distributions, and a topic distribution is consist of unevenly weighted words from the corpus vocabulary. More specifically, for each document w in a corpus $\mathcal{D}$, a word is represented by a multinomial variable w_n, conditioned on the word probabilities parameter β_{z_n} and in topic z_n. z_n is also a multinomial variable depending on parameter θ, a Dirichlet vector determined by parameter vector α. Therefore, LDA is able to explicitly explain the similarity in the data by introducing two layers of latent variables. Upon on the original LDA structure, people have been trying to improve the model by assigning different priors or modifying some structure. Also, people have been adapting the model to various application from natural scene categorization in computer vision [18] to heterogeneous tumor subtype classification [61] in bioinformatics.

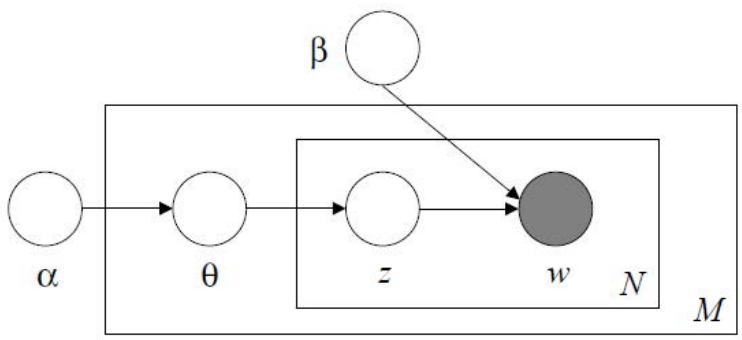


Figure 1.1: Graphical model representation of LDA for document topic modeling. The outer plate represents M documents, while the inner plate represents a N_m-word document [6].

1.3.1 Exact Inference

In general, the problem of inference in graphical model is about obtaining conditional probability of hidden variables given observed variables. According to Bayes rule,

$$P(H|O) = \frac{P(H,O)}{P(O)}, \tag{1.1}$$

where H represents hidden variables, and O stands for observable variables, also known as "evidence". $P(H, O)$ is the joint probability of latent variables and evidence, while $P(O)$ is the marginal probability of the evidence. Depending on the computational complexity of marginal probability $P(O)$, there are exact inference approach for low computational complexity and approximate inference approach for high computational complexity, including sampling inference and variational infernece [70].

People have developed different algorithms to compute the exact conditional probability $P(H|O)$ in order to do exact inference [32, 65]. When applicable, exact inference can be satisfactory as it provides the exact posterior distribution for inference. However, exact inference is limited to cases when time and space complexity of the calculation is manageable.

Exact inference is unpractical when there are many joint latent variables in the hierarchical model or for high dimensional data. When exact conditional probability is intractable, people can sacrifice some accuracy/optimality for the sake of computability. Approximate inference methods including sampling algorithms and variational algorithms are major substitutes under such circumstance.

1.3.2 Sampling Inference

Sampling algorithms provide a methodology for probabilistic inference when exact inference is not applicable. Sampling algorithms have many merits such as guaranteed global optimality and relative easy implementation, which have gained sampling algorithms much popularity. However, as a stochastic method, sampling algorithms requires many samples to converge to the stationary distribution, which might be time-consuming comparing with other deterministic inference methods.

Markov chain Monte Carlo (MCMC) methods have become a popular class of

sampling algorithms with the development of computer computing power in recent years [21, 23, 56, 59]. In MCMC methods, large number of successive random samples are generated from a Markov chain, which can be approximated as target distribution. Rejection sampling [15], Metropolis-hasting sampling [28, 47, 48] are two classical MCMC sampling algorithms. In the Metropolis-Hasting sampling algorithm, random samples are drew from an arbitrary proposal distribution, and are discarded or retained based on a acceptance rule. Gibbs sampling is a special case of Metropolis-hasting sampling algorithm in which the proposal distributions are the conditional distribution of one variable given all other variables, and the conditional distributions are easy to sample from [8, 22, 25]. There are many hybrid algorithms, where Gibbs sampling is incorporated into other MCMC sampling algorithms such as Metropolis-Hasting sampling or Slice sampling [26].

1.3.3 Variational Inference

Variational methods are popular alternatives to sampling algorithms when exact inference is intractable [5, 31]. The basic idea of variational method is to approximate the exact posterior distribution using optimization approach. Variational methods can be generalized as the process of minimizing the Kullback-Leibler (KL) divergence between exact posterior distributions and the approximate distributions, which are generally obtained by decoupling distributions from a graphical model [69, 70]. We actually can't minimize the KL divergence directly; however, it is equivalent to maximizing the evidence lower bound (ELBO) of the data log-likelihood. Neal and Hinton [50] has proposed to maximize the ELBO via Expectation-Maximization(EM) algorithm, which turns out to be very popular.

An important class of methods in the variational inference is Mean Field methods, which originated from statistic physics field. The core idea of mean field ap-

proaches approximate posterior distribution with a class of simplified or tractable distributions which can facilitate the KL divergence minimization. Naive Mean Field method, the simplest Mean field method, assumes the conditional independence of all distribution of interest. This means the conditional probability is optimized as a product of simple distributions. Higher-order mean field methods requires more complex structures [34, 71].

Both variational methods and sampling methods have their advantages and drawbacks. Variational methods provide a locally-optimal,analytical approximation to exact posterior distribution, whereas Monte Carlo sampling algorithms provide a globally-optimal, numerical approximation using large number of samples [34]. As a deterministic method, variational methods often provides comparable accuracy to sampling algorithms with significantly less time. However, deriving and implementing the set of equations for variational algorithm might require a large amount of careful work compared with effort spent on sampling algorithm for comparable results.

1.4 Thesis Organization

Chapter 1 provides background information for the project. It summarizes information on next-generation sequencing technology, review of existing mutation detection methods using next-generation sequencing data, basic knowledge of graphical model and three inference approaches: exact inference, sampling inference and variational inference.

Chapter 2 presents the overall methodology of RVD2 algorithms. It first presents the graphical model of RVD2 with detailed structure interpretation. Then it develops the a Metropolis-within-Gibbs sampling approach to estimate the model and

obtain empirical posterior distribution of interest. Further, it provides a variational approximation approach to estimate model and perform inference. The final part of the Methodology chapter is hypothesis testing algorithms. RVD2 combines two hypothesis testing algorithms to call variants: a posterior distribution test to evaluate whether the case and control samples are significant different from the reference or from each other. A χ^2 test is to performed to test uniformity of non-reference base distribution and to remove false positives from posterior distribution test.

Chapter 3 provides two datasets to test the performance of RVD2. A synthetic DNA sequence data is used to call variants, and the performance can be evaluated by statistics such as sensitivity, specificity and false discovery rate, as the variant positions are know a-prior. A clinical sequence data, HCC1187 sample from subject with primary breast cancer is used to test the performance of RVD2 in real clinical applications.

Chapter 4 shows the variants calling result of RVD2 using sampling inference. Both synthetic data and clinical HCC1187 data are analyzed using RVD2. We compare the performance of RVD2(MCMC sampling) to several other variant calling algorithms for a range of read depths and minor allele fractions. We show that RVD2 is able to call variants on a heterogeneous clinical sample and identify two novel loss-of-heterozygosity events. The performance of variational RVD2 is not currently available as the variational RVD2 is still under implementation at present stage.

Chapter 5 provides some alternative settings for MCMC inference procedure. This includes choosing proposal distribution for Metropolis-Hasting sampling process, determining Gibbs sampling and Metropolis-Hasting sampling size and finding optimal threshold τ^* for posterior density test.

Chapter 6 summarizes the contribution of the work, addresses some concerns

about the preject and provides suggestion of the future work.

Chapter 2

Methodology

2.1 Model Structure

RVD2 uses a two-stage approach for detecting rare variants. First, it estimates the parameters of a hierarchical Bayesian model under two sequencing data sets: one from the sample of interest (case) and one from a known reference sample (control). Then, it tests for a significant difference between key model parameters in the case and control samples and returns called variant positions.

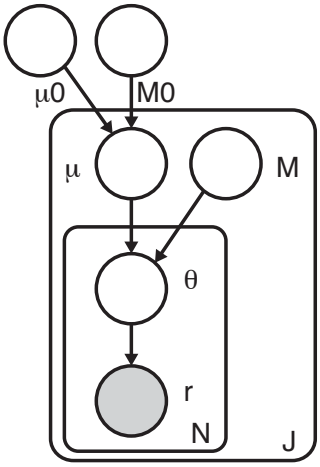


Figure 2.1: RVD2 Graphical Model.

For a given sample, the observed data consists of two matrices $r \in I\!R^{J \times N}$ and $n \in I\!R^{J \times N}$, where r_{ji} is the number of reads with a non-reference base at location j in experimental replicate i and n_{ji} is the total number of reads at location j in replicate i. J is the region of interest length and N is the number of technical replicates in the sample. Technical replicates are used to establish experimental variability in next-generation sequencing procedure [53, 58], though multiple replicates are not necessary for RVD2.

The model generative process given hyperparameters μ_0, M_0 and $M$is as follows:

noitemsep For each location j:

(a) Draw an error rate $\mu_j \sim \text{Beta}(\mu_0, M_0)$

(b) For each replicate i:

i. Draw $\theta_{ji} \sim \text{Beta}(\mu_j, M_j)$

ii. Draw $r_{ji}|n_{ji} \sim \text{Binomial}(\theta_{ji}, n_{ji})$

The generative process involves several hyperparameters: μ_0, a global error rate; M_0, a global precision; μ_j, a local error rate. M_j, a local precision. The global error rate, μ_0, estimates the expected error rate across all locations. The global precision, M_0, estimates the variation in the error rate across locations. The local error rate, μ_j, estimates the exepected error rate across replicates at location j. The local precision, M_j, estimates the variation in the error rate across replicates at location j.

RVD2 has three levels of sampling. First, a global error rate and global precision are chosen once for the entire data set. Then, at each location, a local precision is chosen and a local error rate is sampled from a Beta distribution. Finally, the error rate for replicate i at location j is drawn from a Beta distribution and the number of non-reference reads is drawn from a binomial.

RVD2 hierarchically partitions sources of variation in the data. The distribution $r_{ji}|n_{ji} \sim \text{Binomial}(\theta_{ji}, n_{ji})$ models the variation due to sampling the pool of DNA molecules on the sequencer. The distribution $\theta_{ji} \sim \text{Beta}(\mu_j, M_j)$ models the variation due to experimental reproducibility. The variation in error rate due to sequence context is modeled by $\mu_j \sim \text{Beta}(\mu_0, M_0)$. Importantly, increasing the read depth n_{ji} only reduces the sampling error, but does nothing to reduce experimental variation or variation due to sequence context.

Figure 2.1 shows a graphical representation of the RVD2 statistical model.

The joint distribution over the latent and observed variables for data at location j in replicate i given the parameters can be factorized as

$$p\left(r_{ji}, \theta_{ji}, \mu_j | n_{ji}; \mu_0, M_0, M_j\right) = p\left(r_{ji}|\theta_{ji}, n_{ji}\right) p\left(\theta_{ji}|\mu_j; M_j\right) p\left(\mu_j; \mu_0, M_0\right), \quad (2.1)$$

where

$$\begin{aligned} p\left(\mu_j; \mu_0, M_0\right) &= \frac{\Gamma(M_0)}{\Gamma(\mu_0 M_0)\Gamma(M_0(1-\mu_0))} \cdot \\ &\quad \mu_j^{M_0\mu_0 - 1}(1-\mu_j)^{M_0(1-\mu_0)-1}, \\ p\left(\theta_{ji}|\mu_j; M_j\right) &= \frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))} \cdot \\ &\quad \theta_{ji}^{M_j\mu_j - 1}(1-\theta_{ji})^{M_j(1-\mu_j)-1}, \\ p\left(r_{ji}|\theta_{ji}, n_{ji}\right) &= \frac{\Gamma(n_{ji}+1)}{\Gamma(r_{ji}+1)\Gamma(n_{ji}-r_{ji}+1)} \cdot \\ &\quad \theta_{ji}^{r_{ji}}(1-\theta_{ji})^{n_{ji}-r_{ji}}. \end{aligned} \quad (2.2)$$

Integrating over the latent variables θ_{ji} and μ_j yields the marginal distribution of the data,

$$p\left(r_{ji}|n_{ji};\mu_0,M_0,M_j\right)=\int_{\mu_j}\int_{\theta_{ji}}p\left(r_{ji}|\theta_{ji},n_{ji}\right)$$
$$p\left(\theta_{ji}|\mu_j;M_j\right)p\left(\mu_j;\mu_0,M_0\right)d\theta_{ji}d\mu_j. \quad (2.3)$$

Finally, the log-likelihood of the data set is

$$\log p\left(r|n;\mu_0,M_0,M\right)=\sum_{j=1}^{J}\sum_{i=1}^{N}\log\int_{\mu_j}\int_{\theta_{ji}}p\left(r_{ji}|\theta_{ji},n_{ji}\right)$$
$$p\left(\theta_{ji}|\mu_j;M_j\right)p\left(\mu_j;\mu_0,M_0\right)d\theta_{ji}d\mu_j. \quad (2.4)$$

RVD2 improves on RVD in three ways. First, RVD2 has a Beta(μ_0, M_0) prior on local error rate μ_j, which captures the global across-position error rate. The prior distribution allows μ_j to borrow information from adjacent positions and allows RVD2 to handle low read depths. second, this method smoothly handles multiple replicates in case samples. Third, RVD2 has a more accurate Bayesian hypothesis testing method compared to a frequentist normal z-test in RVD. We show a performance comparison between RVD and RVD2 in Section 4.1.3.

2.2 Sampling Inference

The primary object of inference in this model is the joint posterior distribution function over the latent variables,

$$p(\mu,\theta|r,n;\phi)=\frac{p(\mu,\theta,r|n;\phi)}{p(r|n;\phi)}, \quad (2.5)$$

where the parameters are $\phi \triangleq \{\mu_0, M_0, M\}$.

The Beta distribution over μ_j is conjugate to the Binomial distribution over θ_{ji}, so we can write the posterior distribution as a Beta distribution. However, there is not a closed form for the product of a Beta distribution with another Beta distribution, so exact inference is intractable.

Instead, we have developed a Metropolis-within-Gibbs approximate inference algorithm shown in Algorithm 1. First, the hyperparameters are initialized using method-of-moments (MoM). Given those hyperparameter estimates, we sample from the marginal posterior distribution for μ_j given its Markov blanket using a Metropolis-Hasting rejection sampling rule. Finally, we sample from the marginal posterior distribution for θ_{ji} given its Markov blanket. Samples from θ_{ji} can be drawn from the posterior distribution directly because the prior and likelihood form a conjugate pair. This sampling procedure is repeated until the chain converges to a stationary distribution then we draw samples from the posterior distribution over latent variables.

Algorithm 1 Metropolis-within-Gibbs Algorithm

```
Initialize θ, μ, M, μ_0, M_0
repeat
    for each location j do
        Draw T samples from p(μ_j | θ_ij, μ_0, M_0) using M–H
        Set μ_j to the sample median for the T samples
        for each replicate i do
            Sample from p(θ_ij | r_ij, n_ij, μ_j, M)
        end for
    end for
until sample size sufficient
```

2.2.1 Initialization

The initial values for the model parameters and latent variables is obtained by a method-of-moments (MoM) procedure. MoM works by setting the population moment equal to the sample moment. A system of equations is formed such that the number of moment equations is equal to the number of unknown parameters and the equations are solved simultaneously to give the parameter estimates. We simply start with the data matrices r and n and work up the hierarchy of the graphical model solving for the parameters of each conditional distribution in turn.

We present the initial parameter estimates here and provide the derivations in Supplementary Information. The MoM estimate for replicate-level parameters are $\hat{\theta}_{ji} = \frac{r_{ji}}{n_{ji}}$. The estimates for the position-level parameters are $\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_{ji}$ and $\hat{M}_j = \frac{\hat{\mu}_j(1-\hat{\mu}_j)}{\frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_{ji}^2} - 1$. The estimates for the genome-level parameters are $\hat{\mu}_0 = \frac{1}{J}\sum_{j=1}^{J}\hat{\mu}_j$ and $\hat{M}_0 = \frac{\hat{\mu}_0(1-\hat{\mu}_0)}{\frac{1}{J}\sum_{j=1}^{J}\hat{\mu}_j^2} - 1$.

2.2.2 Sampling from $p\left(\theta_{ij}|r_{ij}, n_{ij}, \mu_j, M\right)$

Samples from the posterior distribution $p(\theta_{ji}|r_{ji}, n_{ji}, \mu_j, M_j)$ are drawn analytically because of the Bayesian conjugacy between the prior $p(\theta_{ji}|\mu_j, M_j) \sim \text{Beta}(\mu_j, M_j)$ and the likelihood $p(r_{ji}|n_{ji}, \theta_{ji}) \sim \text{Binomial}(\theta_{ji}, n_{ji})$. The posterior distribution is

$$p(\theta_{ji}|r_{ji}, n_{ji}, \mu_j, M_j) \sim \text{Beta}\left(r_{ji} + M_j\mu_j, n_{ji} - r_{ji} + M_j(1-\mu_j)\right). \tag{2.6}$$

2.2.3 Sampling from $p\left(\mu_j|\theta_{ji}, M_j, \mu_0, M_0\right)$

The posterior distribution over μ_j given its Markov blanket is

$$p(\mu_j|\theta_{ji}, M_j, \mu_0, M_0) \propto p(\mu_j|\mu_0, M_0)p(\theta_{ji}|\mu_j, M_j). \tag{2.7}$$

Since the prior, $p(\mu_j|\mu_0, M_0)$, is not conjugate to the likelihood, $p(\theta_{ji}|\mu_j, M_j)$, we cannot write an analytical form for the posterior distribution. Instead, we sample from the posterior distribution using the Metropolis-Hastings algorithm.

A candidate sample is generated from the symmetric proposal distribution $Q(\mu_j^*|\mu_j^{(p)}) \sim \mathcal{N}(\mu_j^{(p)}, \sigma_j^2)$, where $\mu_j^{(p)}$ is the pth from the posterior distribution. The acceptance probability is then

$$a = \frac{p(\mu_j^*|\mu_0, M_0)p(\theta_{ji}^{(p+1)}|\mu_j^*, M_j)}{p(\mu_j^{(p)}|\mu_0, M_0)p(\theta_{ji}^{(p+1)}|\mu_j^{(p)}, M_j)} \tag{2.8}$$

We fixed the proposal distribution variance for all the Metropolis-Hastings steps within a Gibbs iteration to $\sigma_j = 0.1 \cdot \hat{\mu}_j \cdot (1 - \hat{\mu}_j)$ if $\hat{\mu}_j \in (10^{-3}, 1 - 10^{-3})$ and $\sigma_j = 10^{-4}$ otherwise, where $\hat{\mu}_j$ is the MoM estimator of μ_j. Though it is not theoretically necessary, we have found that the algorithm performance improves when we take the median of five or more M-H samples as a single Gibbs step for each position (More information shown in Section 5.3).

We resample from the proposal if the sample is outside of the support of the posterior distribution. We typically discard 20% of the sample for burn-in and thin the chain by a factor of 2 to reduce autocorrelation among samples (Detailed autocorrelation analysis shown in Appendix E). Since, each position j is exchangeable given the global hyperparameters μ_0 and M_0 this sampling step can be distributed across up to J processors.

2.3 Variational Inference

As shown in Equation 2.5, inference in RVD2 model is the joint posterior distribution over the latent variables,

$$p(\mu, \theta | r, n; \phi) = \frac{p(\mu, \theta, r | n; \phi)}{p(r | n; \phi)}.$$

Besides Metropolis-within-Gibbs sampling inference, we have developed two variational inference algorithms to approximate the posterior distribution $p(\mu, \theta | r, n; \phi)$ [72]. Here we summarize the algorithm briefly; Appendix J provides a detailed derivation for variational inference procedure.

2.3.1 Evidence Lower Bound (ELBO)

Using Jensen's inequality, the log-likelihood of the data is lower-bounded:

$$\begin{aligned}
\log p\left(r|\phi\right) &= \log \int_{\mu} \int_{\theta} p\left(r, \mu, \theta\right) d\theta d\mu \\
&= \log \int_{\mu} \int_{\theta} p\left(r, \mu, \theta\right) \frac{q\left(\mu, \theta\right)}{q\left(\mu, \theta\right)} d\theta d\mu \\
&= \int_{\mu} \int_{\theta} q\left(\mu, \theta\right) \log \frac{p\left(r, \mu, \theta\right)}{q\left(\mu, \theta\right)} d\theta d\mu + \int_{\mu} \int_{\theta} q\left(\mu, \theta\right) \log \frac{q\left(\mu, \theta\right)}{p\left(\mu, \theta | r\right)} d\theta d\mu \\
&= \mathcal{L}(q, \phi) + KL\left(q(\mu, \theta) || p(\mu, \theta | r)\right) \\
&\geq \mathcal{L}(q, \phi)
\end{aligned} \tag{2.9}$$

where $\phi = (\mu_0, M_0, M)$, $q(\mu, \theta)$ is the variational distribution.

As can be seen from Equation 2.9, the second term of log-likelihood is KL divergence between the variational distribution $q(\mu, \theta)$ and the true posterior distribution $p(\mu, \theta | r)$. The divergence $KL\left(q(\mu, \theta) || p(\mu, \theta | r)\right)$ is always non-negative, and equals

zero if and only if the variational distribution $q(\mu, \theta)$ is exactly the true posterior distribution $p(\mu, \theta|r)$. Therefore, The item $\mathcal{L}(q, \phi)$ is the lower bound of the log-likelihood of the data. The goal of variational inference is to minimizing the KL divergence between these two distribution, but can not be done directly. Equivalently, we maximize the global evidence lower bound $\mathcal{L}(q, \phi)$ in order to minimize the KL divergence.

2.3.2 Factorization

We fully factorize the exact posterior distribution using the naive mean-field method,

$$q(\mu, \theta) = q(\mu)q(\theta) = \prod_{j=1}^{J} q(\mu_j) \prod_{i=1}^{N} q(\theta_{ji}). \tag{2.10}$$

In Equation 2.10, distribution $q(\mu_j)$ approximate the posterior distribution of local error rate μ_j in position j across replicates, while distribution θ_{ji} approximate the posterior distribution of θ_{ji}, the error rate distribution in position j replicate i [4].

2.3.3 Variational Expectation-Maximization

With the conditional independence granted by proposed factorization, we are able to write out ELBO $\mathcal{L}(q, \phi)$ in a relative simple form,

$$\begin{aligned}
\mathcal{L}(q, \phi) &= E_q \left[\log p\left(r, \mu, \theta | n; \phi\right)\right] - E_q \left[\log q\left(\mu, \theta\right)\right] \\
&= E_q \left[\log p\left(r|\theta, n\right)\right] + E_q \left[\log p\left(\theta|\mu; M\right)\right] + E_q \left[\log p\left(\mu; \mu_0, M_0\right)\right] \\
&\quad - E_q \left[\log q\left(\mu\right)\right] - E_q \left[\log q\left(\theta\right)\right].
\end{aligned} \tag{2.11}$$

Writing out each component shows that in order to compute ELBO, we need to obtain the following expectations with respect to variational distribution: $E_q \left[\log \theta_{ji}\right]$, $E_q \left[\log\left(1 - \theta_{ji}\right)\right]$, $E_q \left[\log \mu_j\right]$, $E_q \left[\log(1 - \mu_j)\right]$, $E_q \left[\mu_j\right]$ and $E_q \left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right]$.

As shown in Appendix J.3.1, we propose to use Beta distribution $q(\theta_{ji}) \sim \text{Beta}\,(\delta_{ji})$, which proves to be the optimal variational distribution, to approximate posterior distribution of θ_{ji}. The optimal variational distribution of μ_j is in a complex form; instead, we propose two possible distributions as variational distribution so that the expectations are computable. The first distribution is Beta distribution $q(\mu_j) \sim \text{Beta}\,(\gamma_j)$, which will greatly facilitate the variational inference process. The other choice is Laplace approximation to find an optimal normal distribution $p(\mu_j) \sim \mathcal{N}(\hat{\mu}_j, -f''(\hat{\mu}_j)^{-1})$ [72].

Next, we use a variational EM procedure [34] to maximize ELBO and find the variational parameters that give the best approximate posterior distributions. This process produces approximate maximum-likelihood estimates of the hyperparameters $\phi = \{\mu_0, M_0, M\}$ as well. Variational EM algorithm maximizes the ELBO using coordinate ascent inference – iteratively optimizing each variational distribution while fixing the others.

Variational EM algorithm works by alternating between ELBO maximization over variational distribution $q(\theta, \mu)$(E-step) and ELBO maximization over hyperparameters $\phi = \{\mu_0, M_0, M\}$ (M-step). The inference procedure is shown in Algorithm 2 and Algorithm 3, with the major difference in variational distribution $q(\mu_j)$.

Algorithm 2 RVD2 Variational Inference:
. $q(\theta_{ji};\delta_{ji}) = \text{Beta}(\delta_{ji})$, $q(\mu_j;\gamma_j) = \text{Beta}(\gamma_j)$

```
Initialize q(θ, μ) and φ̂
repeat
    repeat
        for j = 1 to J do
            for i = 1 to N do
                Optimize L(q, φ̂) over q(θ_ji; δ_ji) = Beta(δ_ji)
            end for
        end for
        for j = 1 to J do
            Optimize L(q, φ̂) over q(μ_j; γ_j) = Beta(γ_j)
        end for
    until change in L(q, φ̂) is small
    Set φ̂ ← arg max_φ L(q, φ)
until change in L(q, φ̂) is small
```

Algorithm 3 RVD2 Variational Laplace Inference
. $q(\theta_{ji};\delta_{ji}) = \text{Beta}(\delta_{ji})$, $p(\mu_j) \sim \mathcal{N}(\hat{\mu}_j, -f''(\hat{\mu}_j)^{-1})$

```
Initialize q(θ, μ) and φ̂
repeat
    repeat
        for j = 1 to J do
            for i = 1 to N do
                Optimize L(q, φ̂) over q(θ_ji; δ_ji) = Beta(δ_ji)
            end for
        end for
        for j = 1 to J do
            Construct function f(μ_j) with items in ELBO depending on μ_j;
            Set μ̂_j ← arg max_{μ_j} f(μ_j);
            Approximate q(μ_j) ≈ N(μ̂_j, −f''(μ̂_j, φ̂)^{−1})
        end for
    until change in L(q, φ̂) is small
    Set φ̂ ← arg max_φ L(q, φ)
until change in L(q, φ̂) is small
```

2.4 Hypothesis Testing

2.4.1 Posterior Distribution Test

Posterior Difference Test

Metropolis-within-Gibbs provides samples from the posterior distribution of μ_j given the case or control data. For notational simplicity, we define the random variables associated with these two distributions μ_j^{case} and μ_j^{control} and the associated samples as $\tilde{\mu}_j^{\text{case}}$ and $\tilde{\mu}_j^{\text{control}}$.

A variant is called if $\mu_j^{\text{case}} > \mu_j^{\text{control}}$ with high confidence,

$$\Pr(\mu_j^{\text{case}} - \mu_j^{\text{control}} > \tau) \approx \frac{1}{N_{\text{MH}}} \sum_{k=1}^{N_{\text{MH}}} \mathbb{1}_{\tilde{\mu}_{jk}^{\text{case}} - \tilde{\mu}_{jk}^{\text{control}} > \tau} > 1 - \alpha, \tag{2.12}$$

where τ is a detection threshold and $1 - \alpha$ is a confidence level. We draw a sample from the posterior distribution $\tilde{\mu}_j^{\Delta} \triangleq \tilde{\mu}_j^{\text{case}} - \tilde{\mu}_j^{\text{control}}$ by simple random sampling with replacement from $\tilde{\mu}_j^{\text{case}}$ and $\tilde{\mu}_j^{\text{control}}$.

The threshold, τ, may be set to zero or optimized for a given median depth and desired MAF detection limit. The optimal τ maximizes the Matthews Correlation Coefficient (MCC),

$$\tau^* = \arg\max_{\tau} \{\text{MCC}(\tau)\}. \tag{2.13}$$

While we are able to compute the optimal τ threshold for a test data set, in general we would not have access to τ^*. With sufficient training data, one would be able to develop a lookup table or calibration curve to set τ based on read depth and MAF level of interest. Absent this information we set $\tau = 0$.

Posterior Somatic Test.

Posterior Somatic Test is a two-sided posterior difference test using paired control-case sample. To identify somatic mutations, we consider scenarios when the case(tumor) error rate is lower than the control(germline) error rate (e.g. loss-of-heterozygosity) as well as scenarios when the case(tumor) error rate is higher than the control(germline) error rate (e.g. homozygous somatic mutation). The two hypothesis tests are then $\Pr(\mu_j^{\text{case}} - \mu_j^{\text{control}} > \tau) > 1 - \alpha$ and $\Pr(\mu_j^{\text{case}} - \mu_j^{\text{control}} < \tau) > 1 - \alpha$. Threshold τ is set at zero.

Posterior Germline Test.

Posterior Germline Test is a one-sided posterior distribution test using a single control sample. We call a germline mutation if $\mu_j^{control} \geq \tau$ with high confidence,

$$\Pr(\mu_j^{\text{control}} \geq \tau) \approx \frac{1}{N_{\text{MH}}} \sum_{k=1}^{N_{\text{MH}}} \mathbb{1}_{\tilde{\mu}_{jk}^{\text{control}} \geq \tau} > 1 - \alpha, \tag{2.14}$$

2.4.2 χ^2 test for non-uniform base distribution

An abundance of non-reference bases at a position called by the posterior density test may be due to a true mutation or due to a random sequencing error; we would like to differentiate these two scenarios. We assume non-reference read counts caused by a non-biological mechanism results in a uniform distribution over three non-reference bases. In contrast, the distribution of counts among three non-reference bases caused by biological mutation would not be uniform.

We use a χ^2 goodness-of-fit test on a multinomial distribution over the non-reference bases to distinguish these two possible scenarios. The null hypothesis is $H_0 : p = (p_1, p_2, p_3)$ where $p_1 = p_2 = p_3 = 1/3$. Cressie and Read [12] identified a power-divergence family of statistics, indexed by λ, that includes as special cases

Pearson's $\chi^2(\lambda = 1)$ statistic, the log likelihood ratio statistic ($\lambda = 0$), the Freeman-Tukey statistic ($\lambda = -1/2$), and the Neyman modified statistic $X^2(\lambda = -2)$. The test statistic is

$$2nI^{\lambda} = \frac{2}{\lambda(\lambda+1)} \sum_{k=1}^{3} r_{ji}^{(k)} \left[\left(\frac{r_{ji}^{(k)}}{E_{ji}^{(k)}} \right)^{\lambda} - 1 \right]; \lambda \in R, \tag{2.15}$$

where $r_{ji}^{(k)}$ is the observed frequency for non-reference base k at position j in replicate i and $E_{ji}^{(k)}$ is the corresponding expected frequency under the null hypothesis. Cressie and Read [12] recommended $\lambda = 2/3$ when no knowledge of the alternative distribution is available; we choose that value.

We control for multiple hypothesis testing in two ways. We use Fisher's combined probability test [19] to combine the p-values for N replicates into a single p-value at position j,

$$X_j^2 = -2 \sum_{i=1}^{N} \ln(p_{ji}). \tag{2.16}$$

Equation (2.16) gives a test statistic that follows a χ^2 distribution with $2N$ degrees of freedom when the null hypothesis is true. If the sample average depth is higher than 500, we use the Benjamini-Hochberg method to control the family-wise error rate (FWER) over positions that have been called by the posterior distribution test [3, 16]. The average depth threshold is set because Benjamini-Hochberg method is a highly conservative method and will remove many true calls when the read depth is not high enough.

Chapter 3

Data

We used two independent data sets to evaluate the performance of RVD2 and compare it with other variant calling algorithms. Synthetic DNA sequence data provides true positive and true negative positions as well as define minor allele fractions. HCC1187 data is used to test the performance on a sequenced cancer genome with less than 100% tumor purity.

3.0.3 Synthetic DNA Sequence Data

Experimental process.

Two 400bp DNA sequences(Appendix A) that are identical except at 14 loci with variant bases were synthesized and clonally isolated and labeled case and control(https://www.dna20.com). Sample of the case and control DNA were mixed at defined fractions to yield defined MAFs of 0.1%, 0.3%, 1%, 10%, and 100%. The experimental DNA synthesis and sequencing process is shown in Figure A.1. More details of the experimental protocol are available from the original publication [20].

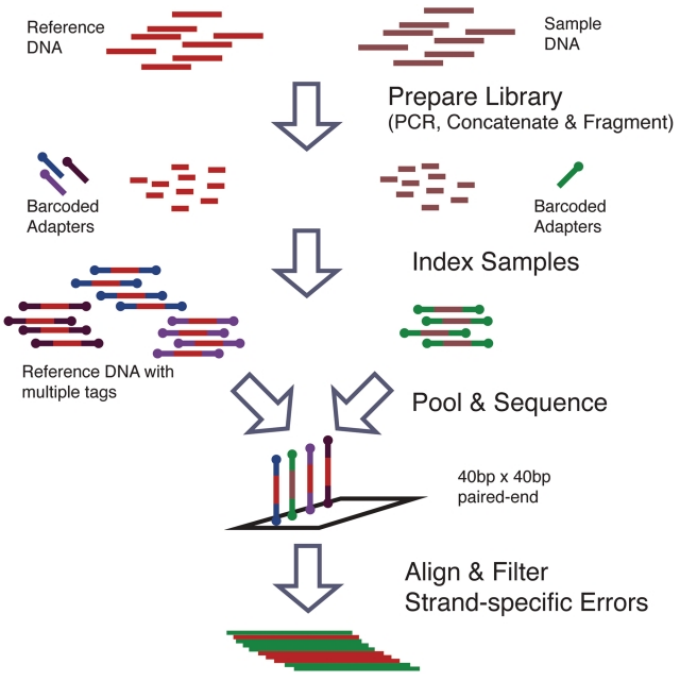


Figure 3.1: Synthetic DNA sequence DNA sequence synthesis and sequencing flowchart. Sample and reference DNA are independently prepared and tagged with indexed adapters. The reference and sample libraries are pooled and sequenced on the same lane. The reads are aligned and preprocessed to filter out strand-specific errors [20].

Computational process.

We aligned the reads to the reference sequence using BWA v0.7.4 with the -C50 option to filter for high mapping quality reads. To simulate lower coverage data while retaining the error structure of real NGS data, BAM files for the synthetic DNA data were downsampled $10\times$, $100\times$, $1,000\times$, and $10,000\times$ using Picard v1.96. The final data set contains read pairs for three replicates of each case and pairs of reads three replicates for the control sample giving $N = 6$ replicates for the control and each MAF level.

3.0.4 HCC1187 Sequence Data

Experimental process.

The HCC1187 dataset is a well-recognized baseline dataset from Illumina for evaluating sequence analysis algorithms [29, 30, 51]. The HCC1187 cell line was derived from epithelial cells from primary breast tissue from a 41 y/o adult with TNM stage IIA primary ductal carcinoma. The estimated tumor purity was reported to be 0.8. Matched normal cells were derived from lymphoblastoid cells from peripheral blood.

Computational process.

Sequencing libraries were prepared according to the protocol described in the original technical report [2]. The raw FASTQ read files were aligned to hg19 using the Isaac aligner to generate BAM files [55]. The aligned data had an average read depth of 40x for the normal sample and 90x for the tumor sample with about 96% coverage with 10 or more reads. We used samtools mpileup to generate pileup files using hg19 as reference sequence [49].

Chapter 4

Result — MCMC Inference

4.1 Synthetic dataset

We tested RVD2 using synthetic DNA and data from a primary ductal carcinoma sample. The Metropolis-within-Gibbs inference algorithm parameters were set to yield $4,000$ Gibbs samples with a 20% burn-in and $2\times$ tinning rate for a final total of $1,600$ samples. We drew $1,000$ samples from $\tilde{\mu}^{\Delta} = \tilde{\mu}_j^{\text{case}} - \tilde{\mu}_j^{\text{control}}$ to estimate the posterior probability of a variant.

We performed posterior difference test on synthetic data to identify mutations given it is a haploid DNA sequence. We set the threshold $\tau = 0$ and the size of the test $\alpha = 0.05$. We used RVD2 to identify somatic and germline mutations in the diploid HCC1187 sample. In the posterior somatic test, we set the threshold $\tau = 0$ and the size of the test $\alpha = 0.05$. In the posterior germline test, We set the threshold $\tau = 0.05$ considering the low average coverage (40x). The size of the test is set at $\alpha = 0.15$, higher than the size of somatic test. We are less confident on the germline test because we only use control sample in the germline test compared to normal-tumor paired sample in somatic test. We performed χ^2 non-uniformity test along all the posterior density test.

4.1.1 Estimated MAF for variants.

Figure 4.1 shows the posterior mean and 95% credible intervals for μ_j for called variant positions with $\bar{n} = 5584$ and MAF = 1.0%. All of the called positions show a clear difference between the case and control error rates. The posterior mean estimates are all shrunken towards the global error rate parameter $\mu_0 = 0.0023$ due to the hierarchical structure of the model.

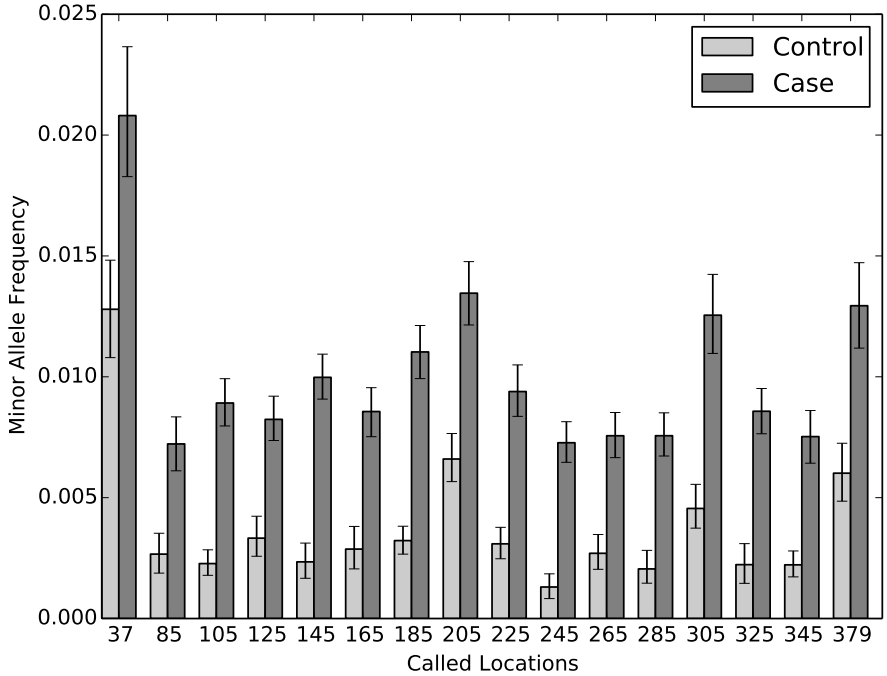


Figure 4.1: Estimated minor allele fraction for called variants in 1.0% dilution.

4.1.2 Performance with read depth

We generated receiver-operating characteristic curves (ROCs) for a range of median read depth and a range of minor allele frequencies (MAFs). For these ROC curves, we used the posterior density test without the χ^2 test to evaluate the performance of posterior density test individually. Figure 4.2 shows ROC curves generated by varying the threshold τ with a fixed $\alpha = 0.05$. Figure 4.2A shows ROC curves for

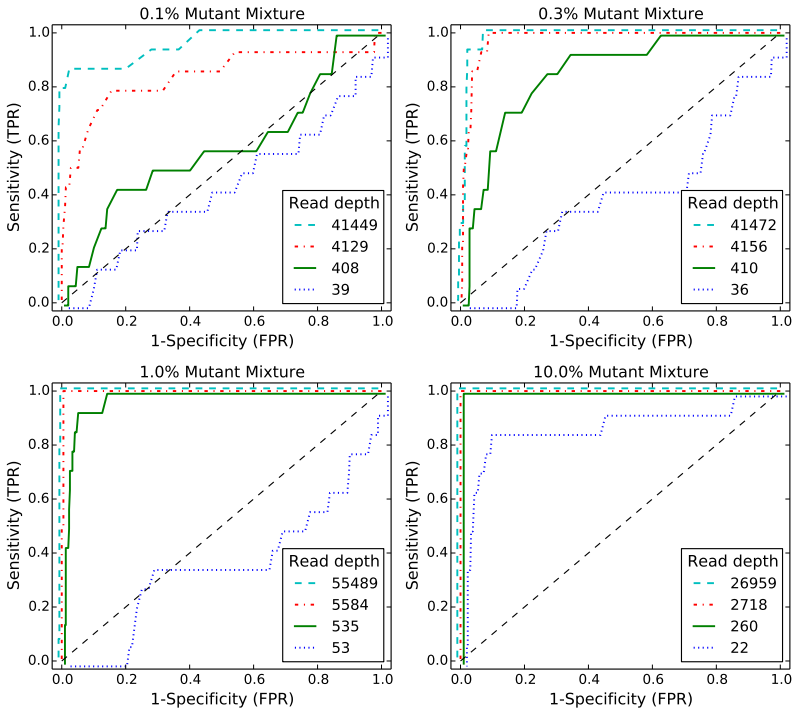


Figure 4.2: ROC curve varying read depth showing detection performance. Each subfigure shows ROC curves across four different read depths for one MAF level. Within one subfigurethe performance improves monotonically with read depth. Across different subfigures, the performance improves with MAF level.

a true 0.1% MAF for a range of median coverage depths. At the lowest depth the sensitivity and specificity is no better than random. However, we would not expect to be able to call a 1 in 1000 variant base with a coverage of only 43. The performance improves monotonically with read depth. Figures 4.2B-C show a similar relationship between coverage depth and accuracy for higher MAFs.

4.1.3 Performance comparison with other algorithms

We compare the empirical performance of RVD2 to other variant calling algorithms using the synthetic DNA data sets using the false discovery rate as well as sensitivity/specificity. Among these algorithms, Samtools and GATK are optimized for homogeneous samples, while RVD, VarScan2 Somatic, Strelka and muTect are designed to call variants in heterogeneous samples, which serve as better comparison

to RVD2. In a research applications, the false discovery rate is a more relevant performance metrics because the aim is generally to identify interesting variants. The sensitivity/specificity metric is more relevant in clinical applications where one is more interested in correctly calling all of the positive variants and none of the negatives. GATK, Varscan2, Strelka and muTect are only able to make use of one case and cone control sample, so we provide results of RVD2 with the same data ($N = 1$) for a fair comparison.

Sensitivity/Specificity Comparison

Figure 4.3 shows that samtools, GATK and VarScan2-mpileup all have similar performance. They call the 100% MAF experiment well even at low depth, but are unable to identify true variants in mixed samples. GATK, samtools and VarScan2-mpileup are optimized to call genotypes on pure samples. Therefore, those algorithms are expected to perform well on the 100% dilution (pure mutant) sample and poorly on heterogeneous samples. VarScan2-somatic is able to call more mixed samples. However, as the read depth increases the specificity degrades. Strelka is able to call 10% MAF variants with good performance, but is limited at 1% MAF and below. muTect has good performance across a wide range of MAF levels. But even at the highest depth only has around 0.5 sensitivity for low MAF levels. The statistics for RVD is an average statistics as RVD uses six control replicates to estimate the model but returns statistics separately for three sets of pair-end case replicates. RVD performed the best among all algorithms when the read depth is as high as 40000x. RVD called all the mutated positions across all MAF levels with no false positives when MAF level is 0.3% or lower. RVD calls some false positives when the MAF level is 1.0% or higher, which causes specificity slightly lower than 1.00. RVD program fails and can not call any mutations when the depth is unable

MAF	Median Depth	SAMtools	GATK	VarScan2 mpileup	VarScan2 somatic	Strelka	MuTect	RVD	N=1		N=6	
									RVD2 (T=0)	RVD2 (T*)	RVD2 (T=0)	RVD2 (T*)
0.1%	39	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/0.99	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	408	0.00/1.00	0.00/1.00	0.00/1.00	0.07/0.92	0.00/1.00	0.29/0.91	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	4129	0.00/1.00	0.00/1.00	0.00/1.00	0.57/0.52	0.00/1.00	0.64/0.86	0.00/1.00	0.00/1.00	0.00/1.00	0.14/1.00	0.29/1.00
	41449	0.00/1.00	0.00/1.00	0.00/1.00	0.64/0.79	0.00/1.00	0.14/0.93	1.00/1.00	0.43/1.00	0.57/1.00	0.86/0.97	0.79/1.00
0.3%	36	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.43/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	410	0.00/1.00	0.00/1.00	0.00/1.00	0.21/0.95	0.00/1.00	0.50/0.94	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	4156	0.00/1.00	0.00/1.00	0.00/1.00	0.57/0.53	0.00/1.00	0.36/0.91	0.00/1.00	0.14/1.00	0.29/1.00	1.00/0.99	1.00/0.99
	41472	0.00/1.00	0.00/1.00	0.00/1.00	0.64/0.75	0.00/1.00	0.43/0.90	1.00/1.00	0.93/0.97	0.93/0.99	1.00/0.85	0.93/0.97
1.0%	53	0.00/1.00	0.00/1.00	0.00/1.00	0.00/0.99	0.00/1.00	0.29/0.98	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	535	0.00/1.00	0.00/1.00	0.00/1.00	0.43/0.89	0.00/1.00	0.71/0.91	0.00/1.00	0.00/1.00	0.00/1.00	0.21/1.00	0.21/1.00
	5584	0.00/1.00	0.00/1.00	0.00/1.00	0.57/0.47	0.00/1.00	0.64/0.95	0.00/1.00	0.93/0.99	1.00/0.99	1.00/0.98	1.00/1.00
	55489	0.00/1.00	0.00/1.00	0.00/1.00	0.64/0.69	0.00/1.00	0.86/0.90	1.00/0.99	1.00/0.95	1.00/0.99	1.00/0.87	1.00/0.99
10.0%	22	0.21/1.00	0.00/1.00	0.00/1.00	0.36/1.00	0.29/1.00	0.86/0.99	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00	0.00/1.00
	260	0.00/1.00	0.00/1.00	0.00/1.00	0.86/1.00	1.00/1.00	1.00/0.99	0.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
	2718	0.00/1.00	0.00/1.00	0.00/1.00	0.57/0.78	1.00/1.00	1.00/0.98	0.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
	26959	0.00/1.00	0.00/1.00	0.00/1.00	0.64/0.53	1.00/0.99	1.00/0.98	1.00/0.98	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
100.0%	27	1.00/0.99	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/0.98	0.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
	298	1.00/0.99	1.00/1.00	1.00/1.00	1.00/0.99	1.00/0.99	1.00/0.98	0.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
	3089	0.86/1.00	1.00/1.00	1.00/1.00	1.00/0.65	1.00/0.99	1.00/0.98	0.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00
	30590	0.71/1.00	1.00/1.00	1.00/1.00	1.00/0.39	1.00/1.00	1.00/0.99	1.00/0.98	1.00/1.00	1.00/1.00	1.00/1.00	1.00/1.00

Figure 4.3: Sensitivity/Specificity comparison of RVD2 with other variant calling algorithms using synthetic sequence data.

to measure the baseline error rate.

The sensitivity for RVD2 with $\tau = 0$ is low for low read depths and MAF levels and $N = 1$ case and control sample. The sensitivity increases considerably with read depth at a slight expense to specificity. With τ^* the performance is much better with high sensitivity and specificity across a wide range of read depths and MAFs. However, in practice one may not know the optimal τ^* a-priori. With $N = 6$ replicates, the sensitivity increases considerably for low MAF variants with a slight degradation in specificity due to false positives. When the median read depth is at least 10× the MAF, RVD2 has higher specificity than all of the other algorithms tested and has a lower sensitivity in only three cases.

False Discovery Rate Comparison

Figure 4.4 shows the false discovery rate for RVD2 compared to samtools, GATK, varscan, Strelka and muTect. Blank cells indicate no positive calls were made.

Samtools performs well on 100% MAF sample and performance improves for read depths 3,089 and 30,590. GATK performs well on both the 10% and 100%

MAF	Median Depth	SAMtools	GATK	VarScan2 mpileup	VarScan2 somatic	Strelka	MuTect	RVD	N=1		N=6	
									RVD2 (T=0)	RVD2 (T*)	RVD2 (T=0)	RVD2 (T*)
0.1%	39						1.00					
	408				0.97		0.89					
	4129				0.96		0.86				0.00	0.00
	41449				0.90		0.93	0.04	0.14	0.11	0.50	0.08
0.3%	36						0.14					
	410				0.86		0.76					
	4156				0.96		0.87		0.00	0.00	0.26	0.26
	41472				0.92		0.87	0.08	0.43	0.28	0.80	0.43
1.0%	53				1.00		0.67					
	535				0.87		0.78				0.00	0.00
	5584				0.96		0.70		0.19	0.18	0.30	0.07
	55489				0.93	1.00	0.76	0.19	0.59	0.22	0.78	0.12
10.0%	22	0.00			0.00	0.00	0.25					
	260				0.08	0.00	0.18		0.00	0.00	0.00	0.00
	2718				0.91	0.07	0.36		0.00	0.00	0.00	0.00
	26959				0.95	0.18	0.33	0.31	0.00	0.00	0.00	0.00
100.0%	27	0.12	0.07	0.07	0.00	0.07	0.36		0.00	0.00	0.00	0.00
	298	0.12	0.07	0.00	0.12	0.18	0.39		0.00	0.00	0.00	0.00
	3089	0.00	0.07	0.00	0.91	0.18	0.33		0.00	0.00	0.00	0.00
	30590	0.00	0.07	0.00	0.94	0.00	0.26	0.3	0.00	0.00	0.00	0.00

Figure 4.4: False discovery rate comparison of RVD2 with other variant calling algorithms using synthetic sequence data. Blank cells indicate no locations were called variant.

variants, but makes a false positive call at the 100% MAF level for all read depth levels. VarScan2-pileup performs perfectly for all but the lowest depth for the 100% MAF.

VarScan2-somatic is able to make calls for all but the lowest MAF and coverage level. However, the FDR is high due to many false positives. Interestingly, at a MAF of 100% the FDR is zero for lowest read depth and over 0.9 for the highest read depth. Strelka has a better FDR than the samtools, GATK or Varscan2-somatic algorithms for almost all read depths at the 10% and 100% MAF. However, it does not call any variants at or below 1% MAF. muTect has the best FDR performance of the other algorithms we tested over a wide range of MAF and depths. But the FDR level is relatively high at around 0.7 for 0.1% – 1% MAF and 0.3 for 10% – 100% MAF. RVD has best FDR performance in the high read depth for 0.1% – 1% MAF levels. The FDR increases to around 0.3 for 10% – 100% MAF in the high read depth.

RVD2 has a lower FDR than other algorithms when the read depth is greater than 10× the MAF with $N = 1$ and τ set to the default value of zero or to the optimal

value. The FDR is higher when $N = 6$ because the variance of the control error rate distribution $P(\mu_j^{\text{control}}|r^{\text{control}})$ is smaller. The smaller variance yields improvements in sensitivity at the expense of more false positives. Since the FDR only considers positive calls, the performance by that measure degrades.

Matthews Correlation Coefficient Comparison

Figure 4.5 compares RVD2 with samtools, GATK, varscan, strelka and muTect using Matthews Correlation Coefficient (MCC) [45].

									N=1		N=6	
MAF	Median Depth	SAMtools	GATK	VarScan2 mpileup	VarScan2 somatic	Strelka	MuTect	RVD	RVD2 (T=0)	RVD2 (T*)	RVD2 (T=0)	RVD2 (T*)
0.1%	39						-0.02					
	408				-0.00		0.12					
	4129				0.03		0.25				0.37	0.53
	41449				0.19		0.05	0.98	0.60	0.70	0.64	0.84
0.3%	36						0.60					
	410				0.14		0.31					
	4156				0.04		0.17		0.37	0.53	0.85	0.85
	41472				0.16		0.19	0.95	0.71	0.81	0.41	0.71
1.0%	53				-0.02		0.29					
	535				0.18		0.36				0.46	0.46
	5584				0.01		0.41		0.86	0.90	0.83	0.96
	55489				0.13	-0.01	0.43	0.9	0.62	0.88	0.43	0.93
10.0%	22	0.46			0.59	0.53	0.79					
	260				0.89	1.00	0.90		1.00	1.00	1.00	1.00
	2718				0.16	0.96	0.79		1.00	1.00	1.00	1.00
	26959				0.06	0.90	0.81	0.82	1.00	1.00	1.00	1.00
100.0%	27	0.93	0.96	0.96	1.00	0.96	0.79		1.00	1.00	1.00	1.00
	298	0.93	0.96	1.00	0.93	0.90	0.77		1.00	1.00	1.00	1.00
	3089	0.92	0.96	1.00	0.25	0.90	0.81		1.00	1.00	1.00	1.00
	30590	0.84	0.96	1.00	0.15	1.00	0.85	0.83	1.00	1.00	1.00	1.00

Figure 4.5: Matthews correlation coefficient (MCC) comparison with other variant calling algorithms.

Samtools and VarScan2-mpileup achieved MCC value generally higher than 0.90 on 100% MAF sample across all read depthes, with 1.0 represents for a perfect prediction. However, both of them detected no variant when MAF is 10.0% or lower, with only one exception for samtools when MAF is 10.0% and read depth 22. GATK, Varscan2-somatic, Strelka and GATK outperformed Samtools and VarScan2-mpile on the 10.0% MAF sample, while approximately tied in other cases. Strelka achieved best MCC on 10% MAF sample comparing to Varscan2-somatic and GATK, more specifically around 1.00 when read depth is 260 or higher. There is a very obvious but unconventional decreasing trend in VarScan2-somatic MCC value across differ-

ent read depth and MAF level, a phenomenon also observed by 68. It is because VarScan2-somatic tends to call more false positives as read depth gets higher. Mutect seems to performs the best among all the algorithms expect RVD2 when MAF is 1.0% or lower. It achieves MCC values varying from -0.02 to 0.43, though too low to be practically meaningful. However, muTect achieved relatively lower MCC values when the MAF level is 10% and 100%, as a counteractive of being oversensitive.

RVD2 achieved MCC value 1.00 when the MAF is 100.0% at all read depth and 10% when read depth is not lower than 260. This indicates that RVD2($\tau = 0, N = 1$) is more accurate than the other algorithms when the median read depth is at least 10× the MAF.

4.2 HCC1187 primary ductal carcinoma sample

4.2.1 Performance of RVD2.

RVD2 identified twelve variants in the 44kbp PAXIP1 gene from chr7:154738059 to chr7:154782774. There were eight germline variants and eight somatic mutations in the twelve variants. RVD2 identified twelve variants in the 44kbp PAXIP1 gene from chr7:154738059 to chr7:154782774. There were eight germline variants and eight somatic mutations. Figure 4.6 shows the estimated minor allele frequencies for the normal and tumor samples at the called locations. Positions chr7:154743899C>T, chr7:154749704G>A, chr7:154753635T>C, chr7:154754371T>C, chr7:154758813G>A, chr7:154766700C>A, chr7:154780960C>T, and chr7:154781769G >T were called germline mutations. Positions chr7:154749704G>G, chr7:154753635T>C, chr7:154754371T>C, chr7:154758813G>A, chr7:154760439A>C, chr7:154766732T>G, chr7:154766832A>C, chr7:154777118A>C were identified as significantly different in tumor and normal sample MAF and called somatic mutation. Positions chr7:154754371 and chr7:154758813

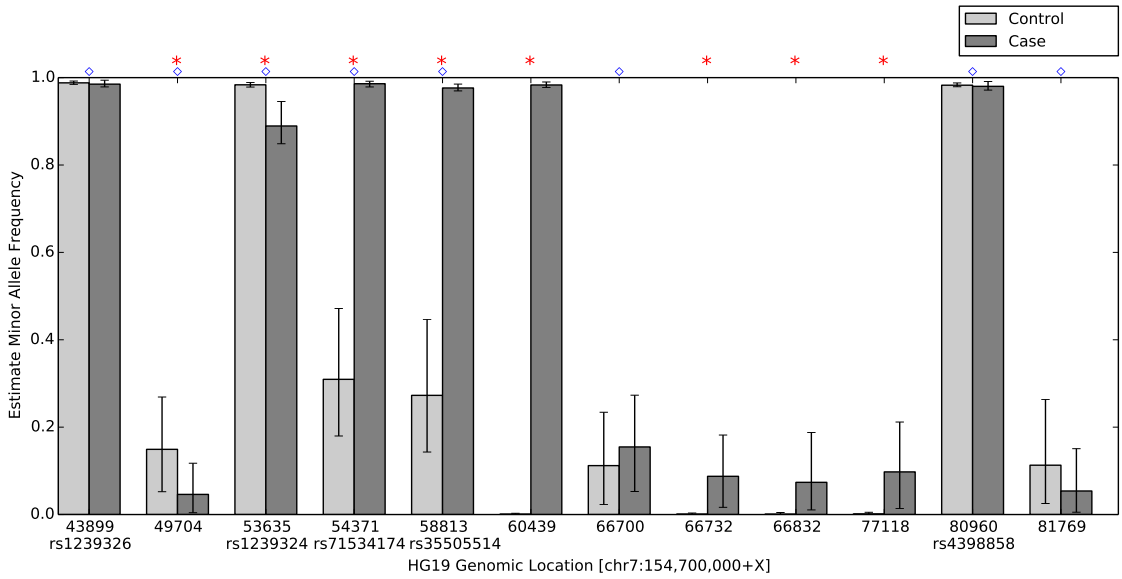
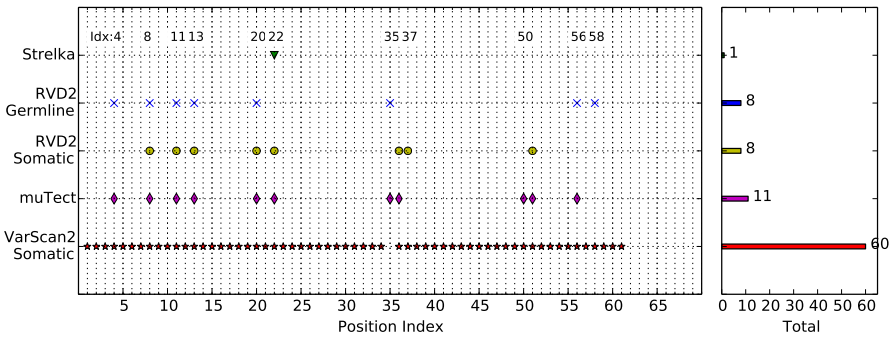


Figure 4.6: Estimated minor allele fraction for Germline and Somatic mutations called by RVD2 in the 44kbp PAXIP1 gene from chr7:154738059 to chr7:154782774. Blue diamonds (◇) indicates germline mutations, where μ^{control} is significantly different from the reference sequence. Red stars (*) indicates somatic mutations, where μ^{case} is significantly different from μ^{control}. The vertical lines represent 95% credible interval around posterior mean MAF. Five positions are common populations SNPs according to dbSNPv138, and the identities are shown below the positions.

appears to be loss-of-heterozygosity events. Some of these mutations are also found to be common population SNPs according to dbSNPv138. The corresponding identities are shown in the Figure 4.6. The read depth distribution for positions called by RVD2 are provided in Appendix I.

4.2.2 Performance comparison with other algorithms.

We ran muTect and VarScan2-somatic to call mutations in the PAXIP1 gene in HCC1187 sample. We also compared to the result shown in original research report where Strelka was used to identify mutations in the same sample [2]. Figure 4.7a shows mutation detection result from Strelka, RVD2, muTect, and VarScan2-somatic, the state-of-art algorithms able to call mutation from heterogeneous samples. For notation simplicity, we use position index to present actual positions in Figure 4.7, while the correspondence is provided in Appendix I.



Index	REF	Normal	Case
		A:C:G:T	A:C:G:T
4	C	0:0:0:45	0:0:0:71
8	G	8:0:36:0	2:0:42:0
11	T	0:38:0:0	0:63:0:1
13	T	0:19:0:31	0:61:0:0
20	G	14:0:28:0	54:0:0:0
22	A	37:0:0:0	0:38:0:0
35	C	4:21:0:0	10:42:0:0
36	T	0:0:0:35	0:0:5:44
37	A	34:0:0:0	42:4:0:0
50	C	0:49:0:0	0:64:2:0
51	A	46:0:0:0	31:4:0:0
56	C	0:0:0:42	0:0:0:56
58	G	0:0:21:4	0:0:31:2

(a) (b)

Figure 4.7: (a)Positions called by VarScan2-somatic, muTect, RVD2 and Strelka in the 44kbp PAXIP1 gene from chr7:154738059 to chr7:154782774. VarScan2-somatic reported 60 positions, muTect 11 positions, RVD2 12 positions and Strelka only 1 position. This figure uses position index to show the correspondence of positions called by different algorithms for notation simplicity. Complete actual positions and depth distribution are provided in Supplementary Table 1 for validation. (b) Depth distribution for positions called by RVD2 and muTect.

The mutations called by RVD2 and muTect are the most consistent among all the techniques. RVD2 detected twelve germline and somatic mutations, while muTect reported eleven, ten in common. In the disagreements, RVD2 did not call position 50 while muTect did not call position 37 and 58. Referring to the depth distribution shown in Figure 4.7b, it can be seen that position 37 and 58 are more likely mutated while position 50 is less likely mutated.

Strelka was the least sensitive algorithms among all the algorithms. According to the technical report, Strelka identified position 22 (chr7:154760439) as variant, but did not call any other variants. In particular Strelka missed the two LOH events called by RVD2. On the contrary, VarScan2-somatic called most positions among all algorithms, sixty positions as shown in Figure 4.7a. VarScan2-somatic detected all the positions called by RVD2 except position 35, which turns out to be a very likely mutation given the depth distribution in Figure 4.7b. On the other side, VarScan2-somatic reported fifty positions which were not called by any other three algorithms. The read depth in Supplementary Table 1 suggestions that these positions are very likely to be false positives. The fact that VarScan2-somatic can be over-sensitive has appeared in synthetic dataset analysis. As shown in Figure 4.4, the False Discovery

Rate for VarScan2-somatic at read depth 53 MAF level 1.0% is as high as 1.00. Spencer et al. [66] also mentioned that VarScan2 has tendency to call many false positives at high read depth.

Chapter 5

Alternative Approaches in MCMC inference

5.1 Proposal distribution in Metropolis-Hasting sampling

5.1.1 Detailed balance

The principle of detailed balance is that each elementary process should be equilibrated by its reverse process in the stationary state of a system. Detailed balance has been applied to many field including various MCMC methods, where equilibrium distributions are target posterior distributions [47].

In order to satisfy detailed balance in Metropolis-Hasting sampling process, a normal distribution $Q(\mu_j^*|\mu_j^{(p)}) \sim \mathcal{N}(\mu_j^{(p)}, \sigma_j^2)$ is used as proposal distribution to generate candidate samples for Metropolis-Hasting sampling, where $\mu_j^{(p)}$ is the pth sample from the posterior distribution. We fixed the proposal distribution variance for all the Metropolis-Hastings steps within a Gibbs iteration to $\sigma_j = 0.1 \cdot \hat{\mu}_j \cdot (1 - \hat{\mu}_j)$ if $\hat{\mu}_j \in (10^{-3}, 1 - 10^{-3})$ and $\sigma_j = 10^{-4}$ otherwise, where $\hat{\mu}_j$ is the MoM estimator of

μ_j.

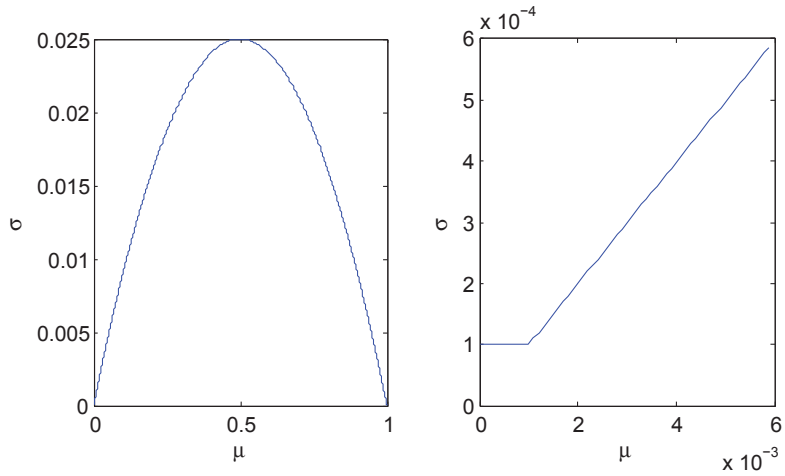


Figure 5.1: Standard deviation with respect to mean in proposal distribution $Q(\mu_j^*|\mu_j^{(p)}) \sim \mathcal{N}(\mu_j^{(p)}, \sigma_j^2)$ for Metropolis-Hasting sampling. The left plot is the overall plot with $\mu_j^{(p)}$ across (0,1), while the right plot is a magnification of $\mu_j^{(p)}$ at the one of the two break points.

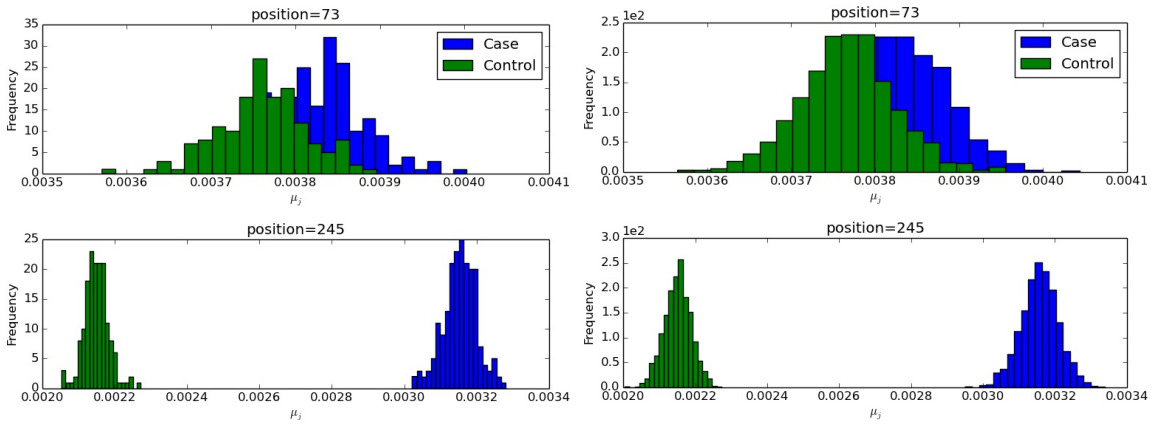
5.1.2 Obtain σ_j from $\hat{\mu}_j$

Prior to the proposal distribution described above, several other options were explored. First, in stead of $\hat{\mu}_j$, the MoM estimate of μ_j to provide σ, we tried it out whether it is feasible to use the $\mu_j^{(p)}$, the pth sample from the posterior distribution. This means the proposal distribution is not fixed in standard deviation and the sampling process will violate detailed balance requirement. This is distribution was still under consideration because of two reasons: we believe it makes better proposal distribution to adjust how wide the proposal distribution is according to the mean; and Manousiouthakis and Deem [42] stated that strict detailed balance is unnecessary in Monte Carlo simulation. However, as it turned out the performance of RVD2 is slightly better when the detailed balance is met, we vetoed the proposal distribution adjusting standard deviation according to the mean.

Second, we explored to how to find the standard deviation σ_j for proposal distri-

bution from the MoM estimate $\hat{\mu}_j$. We found out that the symmetric relationship $\sigma_j = 0.1 \cdot \hat{\mu}_j \cdot (1 - \hat{\mu}_j)$ if $\hat{\mu}_j \in (10^{-3}, 1 - 10^{-3})$ and $\sigma_j = 10^{-4}$ gives desirable acceptance rate between 0.2-0.8.

5.2 Gibbs Sampling size



(a) (b)

Figure 5.2: Histogram of $\mu_j^{control}$ and μ_j^{case} to evaluate the sufficiency of Gibbs sampling size. The Metropolis-Hasting sampling size is 50 in this experiment.(a)Gibbs sampling size at 400; (b)Gibbs sampling size at 4000.

The sampling inference uses Metropolis-within-Gibbs sampling approach to estimate the RVD2 hierarchical empirical Bayes model. However, it might require many Gibbs samples to achieve convergence and guarantee global optimal parameter settings. Presently there is no very good and simple ways to evaluate the convergence Gibbs sampling size. Therefore, we provide histograms of Gibbs samples of $\mu_j^{control}$ and μ_j^{case} to intuitively evaluate the convergence of Gibbs sampling, as shown in Figure 5.2.

Figure 5.2a shows the histograms of $\mu_j^{control}$ and μ_j^{case} at Gibbs sample size 400, and Figure 5.2b shows the histograms at sample size at 4000 for position 73 and position 245, respectively. Compare the histograms in Figure 5.2a and Figure 5.2b, it can be seen that histogram at Gibbs sampling size is much smoother, which

indicating the posterior distribution is much more sufficiently sampled.

Increasing the Gibbs sampling size can improve accuracy but at a cost of speed and memory. We opt this balance by setting Gibbs sampling size at 4000, where the time and space complexity is acceptable, and the accuracy is satisfactory as well.

5.3 Metropolis-Hasting sampling size

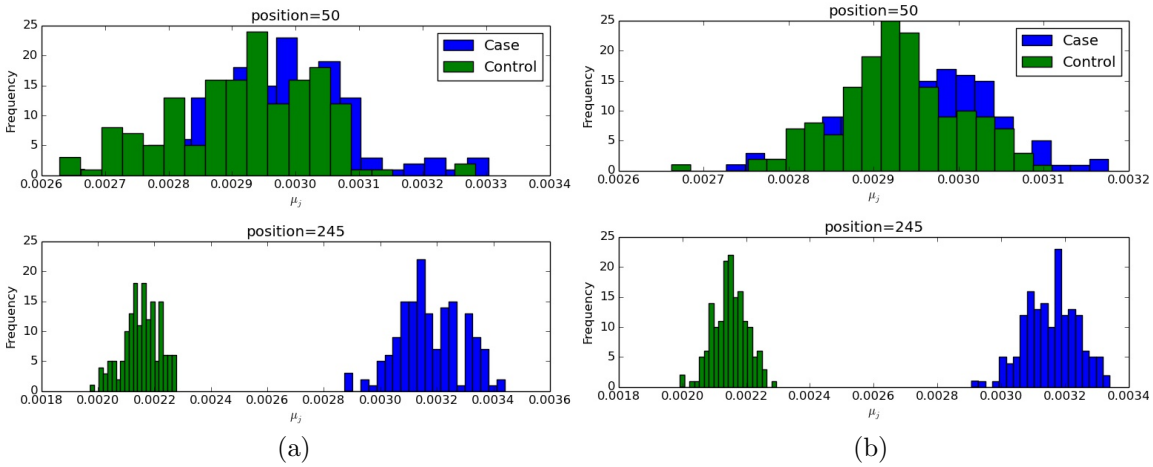


Figure 5.3: Histogram of $\mu_j^{control}$ and μ_j^{case} to evaluate the sufficiency of Metropolis-Hasting sampling size. The Gibbs sampling size is 400 in this experiment.(a)Metropolis-Hasting sampling size at 1; (b)Metropolis-Hasting sampling size at 10.

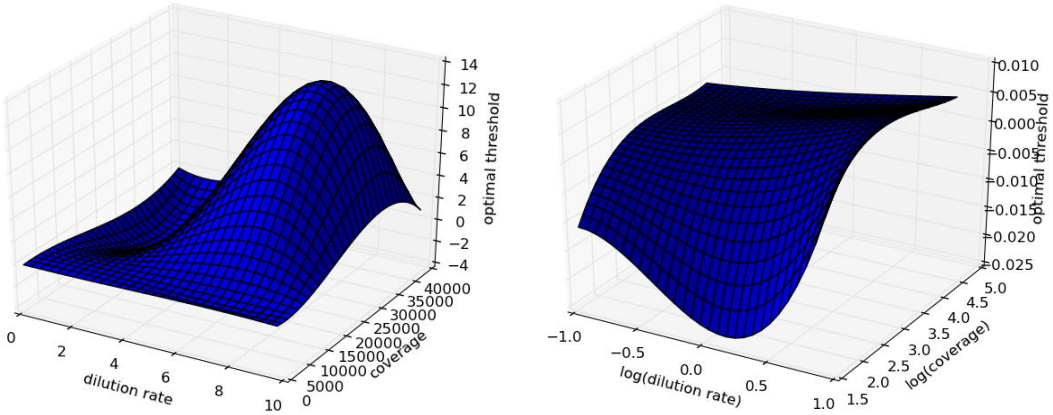
Within one Gibbs sampling iteration, the median of Metropolis-Hasting samples is returned as a single Gibbs step for posterior distribution. From Figure 5.3 it can be seen that appropriate large Metropolis-Hasting sampling size facilitates the convergence of the sampling process. Considering that only one Gibbs step is obtained from one Gibbs iteration, very large Metropolis-Hasting size may lead to high cost of speed in order to acquire enough Gibbs samples. Therefore, Metropolis-Hasting size between 5 to 10 is recommended in RVD2 sampling procedure.

5.4 Optimal Threshold τ^* in posterior density test

As shown in Section 4.1.3, optimal threshold τ^* can yield better Sensitivity, Specificity and FDR than arbitrary threshold $\tau = 0$ in the synthetic dataset. The optimal threshold τ^* can be obtained by maximizing MCC in a test data set; however, we would not have access to τ^* in general.

We have tried to fit a calibration curve to set τ based on read depth and MAF level of interest in the synthetic dataset. Figure 5.4 shows the three dimensional splined curve of τ^* with respect to dilution rate and coverage in linear or logarithmic scale.

The calibration curves in Figure 5.4 visualizes some trends in the relationship of τ^* with dilution rate and coverage. In general, optimal threshold is less variable over dilution rate than coverage; the optimal threshold constantly increases over dilution rate. However, limited by the few number of data points, the splined curves is not a effective calibration curve to find optimal threshold τ^*. Therefore, before we have access to more data, we currently use $\tau = 0$ as threshold in practice.



(a) (b)

Figure 5.4: Three dimensional Splined calibration plot for optimal threshold τ^* across different dilution rate (0.1%, 0.3%, 1.0%, 10.0%). (a)Linear spline calibration; (b)Log scale spline calibration.

Chapter 6

Discussion and Future work

6.1 Discussion

We describe here a novel statistical tool, RVD2, for model estimation and hypothesis testing for identifying single-nucleotide variants in heterogeneous samples using next-generation sequencing data. RVD2 using sampling inference has been implemented in python, and it has a higher sensitivity and specificity than many other approaches for a range of read depths and minor allele frequencies. The variational implementation of RVD2 is still under development.

Our inference algorithm uses Gibbs sampling to estimate the RVD2 hierarchical empirical Bayes model. This sampling procedure provides a guarantee to identify the global optimal parameter settings asymptotically. However, it may require many samples to achieve that guarantee causing the algorithm to be slower than other deterministic approaches. We opted for this balance of speed and accuracy because computational time is often not limiting and the cost of a false positive or false negative greatly outweighs the cost of more computation. Another factor that can affect the speed of RVD2 is the number of Metropolis-Hasting sample within one Gibbs sampling run. However, RVD2 is able to use multiple cores in parallel (

no more than the number of Metropolis-Hasting samples), which can significantly improve time efficiency. Using a computational mode with 64 2.34GHz processor and 256Gb of RAM, RVD2 takes around 10 minutes to analyze the 400bp long synthetic data for one dilution rate and one downsampling rate. The memory requirement is no higher than the size of the gene sequence times the number of Gibbs samples. We are currently developing a variational method to estimate RVD2, aiming at decreasing time complexity.

We have focused on the statistical model and hypothesis test in this study and our results do not include any pre-filtration of erroneous reads or post-filtration of mutation calls beyond a simple quality score threshold. Incorporation of such data-cleaning steps will likely improve the accuracy of the algorithm.

Our approach does not address identification of indels, structural variants or copy number variants. Those mutations typically require specific data analysis models and tests that are different than those for single-nucleotide variants. Furthermore, analysis of RNA-seq data or other data generated on the NGS platform may require different models that are more appropriately tuned to the particular noise feature of that data.

The availability of clinical sequence data is increasing as the technical capability to sequence clinical samples at low cost improves. Consequently, we require statistically accurate algorithms that are able to call germline and somatic point mutations in heterogeneous samples with low purity. Such accurate algorithms are a step towards greater access to genomics for clinical diagnostics.

6.2 Future Work

One future aspect to work on is to improve the model structure of RVD2. A possible direction is to assign appropriate priors on local precision parameter M_j, which is obtained using MoM method at present stage. A prior on M_j can regulate the experimental precision, which might be able to improve the performance. Moreover, a prior on M_j will solve the problem that M_j is not numerically computable when there is only one sample replicate. Some prior choices that is under exploring right now is Jeffrey's prior, Log-normal prior and Gamma prior.

Another direction of this project is to implement the MCMC sampling approach in using probabilistic programming language Stan [67]. As one of the major advantages, Stan is able to significantly improve the speed of RVD2 program.

A third field to work on RVD2 is to implement the variational RVD2 algorithm. The algorithm has been partially implemented in Python till now. Also, the variational algorithms involves several numerical integration to compute the expectations. It will greatly improve the speed of variational RVD2 if it can be improved to analytical process.

With a working algorithm now, a very important direction to work on is to apply RVD2 to many other clinical datasets. This will make scientific discoveries on the dataset, along which the RVD2 could be further improved.

Appendix A

Synthetic gene sequence

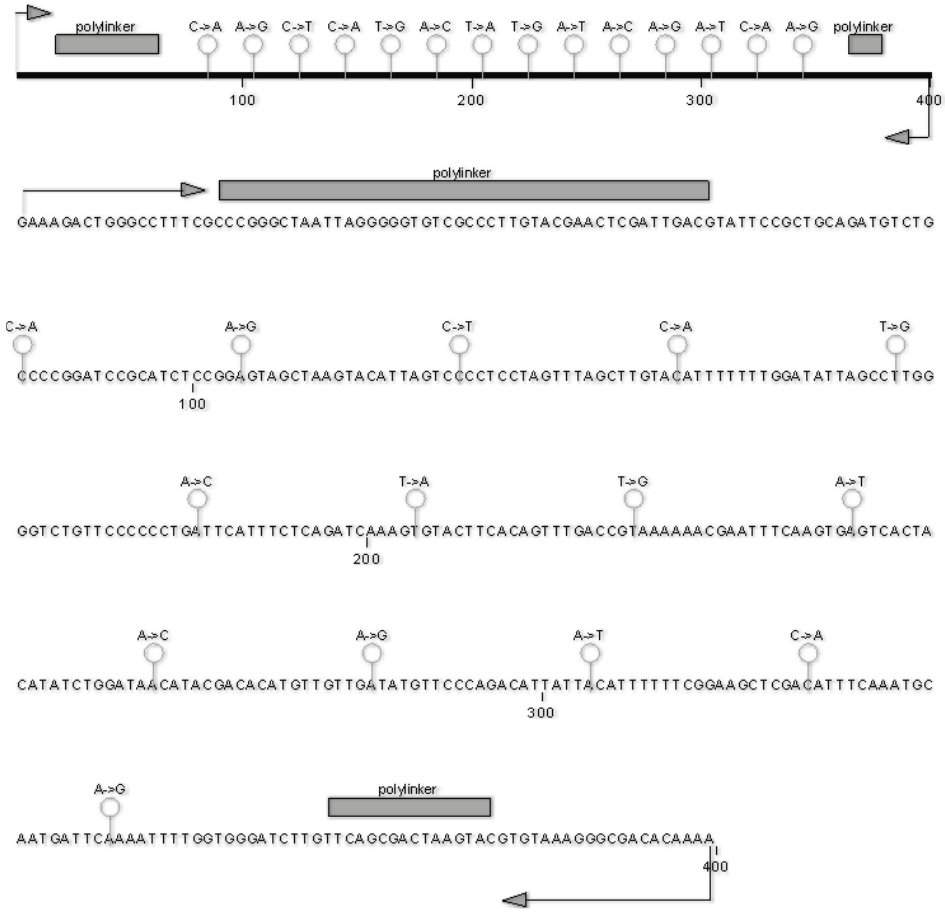


Figure A.1: Synthetic gene sequence. The synthetic gene reference is shown and the companion sequence with 14 known mutant positions is shown marked on the sequence. A polylinker sequence facilitates cloning into a DNA vector. [20]

Appendix B

Simulation Example

Figure B.1 shows an simulation example to illustrate the proposal graphical model and generative process.

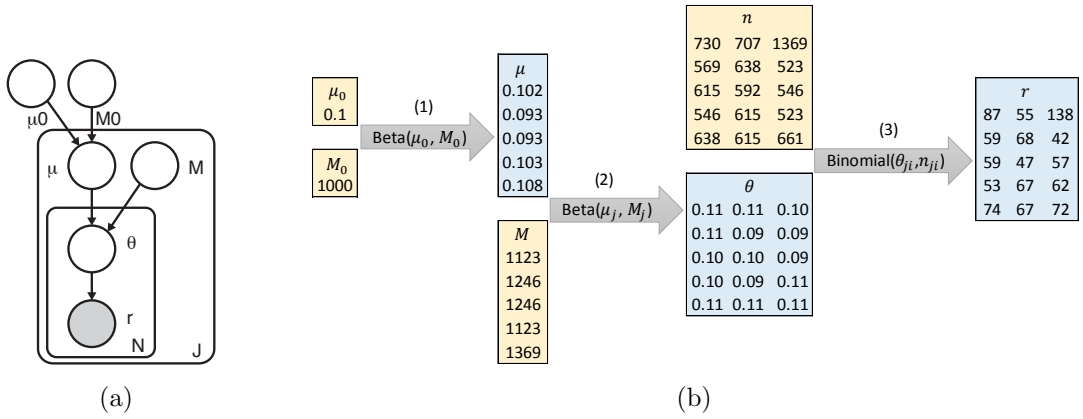


Figure B.1: (A) RVD2 Graphical Model.(B) An simulation example to illustrate the proposal graphical model and generative process. A generative process is to generate randomly observable data following your generative model, which is the reverse process of doing inference from the actual data. Here we want to generate a simulation sample with 3 replicates, and each replicate is a 5bp long sequence. Therefore, we have $J = 5, N = 3$. The numbers in orange are known, including $\mu_0 \in \mathbb{R}^{1\times 1}$, $M_0 \in \mathbb{R}^{1\times 1}$, $M \in \mathbb{R}^{5\times 1}$, $n \in \mathbb{R}^{5\times 3}$. (1) We sample local error rate $\mu \in \mathbb{R}^{5\times 1}$ by drawing random numbers $\mu_j \sim \text{Beta}(\mu_0, M_0)$; (2) and then use μ and M to sample $\theta \in \mathbb{R}^{5\times 3}$ by drawing random samples $\theta_{ji} \sim \text{Beta}(\mu_j, M_j)$; (3) finally we use $\theta \in \mathbb{R}^{5\times 3}$ and $n \in \mathbb{R}^{5\times 3}$ to sample $r \in \mathbb{R}^{5\times 3}$ by drawing random numbers $r_{ji} \sim \text{Binomial}(\theta_{ji}, n_{ji})$. In actual practice, non-reference read counts r and coverage n are the data we use to do inference.

Appendix C

Parameter Initialization Derivation

Since $r_{ji} \sim \text{Binomial}(n_{ji}, \theta_{ji})$, the first population moment is $E[r_{ji}] = \theta_{ji} n_{ji}$ and the first sample moment is simply $m_1 = r_{ji}$. Therefore the MoM estimator is

$$\hat{\theta}_{ji} = \frac{r_{ji}}{n_{ji}}$$

We take the MoM estimate, $\hat{\theta}_{ji}$, as data for the next conditional distribution in the hierarchical model. The distribution is $\theta_{ji} \sim \text{Beta}(\mu_j M_j, (1-\mu_j)M_j)$. The first and second population moments are

$$E[\theta_{ji}] = \mu_j, \qquad \text{Var}[\theta_{ji}] = \frac{\mu_j(1-\mu_j)}{M_j+1}.$$

The first and second sample moments are $m_1 = \frac{1}{N}\sum_{i=1}^{N}\theta_{ji}$ and $m_2 = \frac{1}{N}\sum_{i=1}^{N}\theta_{ji}^2$. Setting the population moments equal to the sample moments and solving for μ_j

and M_j gives

$$\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_{ji}, \qquad \hat{M}_j = \frac{\hat{\mu}_j(1-\hat{\mu}_j)}{\frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_{ji}^2} - 1.$$

Following the same procedure for the parameters of $\mu_j \sim \text{Beta}(\mu_0, M_0)$ gives the following MoM estimates

$$\hat{\mu}_0 = \frac{1}{J}\sum_{j=1}^{J}\hat{\mu}_j, \qquad \hat{M}_0 = \frac{\hat{\mu}_0(1-\hat{\mu}_0)}{\frac{1}{J}\sum_{j=1}^{J}\hat{\mu}_j^2} - 1.$$

Appendix D

RVD2 Estimated Parameters

The RVD2 algorithm provides estimates of model parameters and latent variables given the data. We show several of these parameters in Figure D.1.

The left column of Figure D.1 shows the read depth for each of the six bam files (three replicates each with two read pairs) for each data set. Because the DNA was not sheared and ligated prior to sequencing, the read depth drops to zero at the boundaries. For the 100% mutant data set, the read depth drops at the mutant locations. This is due to the parameters imposed at the alignment stage. The reads are 36bp long and we required no more than 2 mismatches. Therefore, reads that overlapped two mutations (spaced 20bp apart by design) and included one additional mutation would not align.

The right column of Figure D.1 shows the parameter estimates $\hat{M}_j$ and $\hat{M}_0$ for each data set. M_j measures the variance between replicates at location j. There is little variability across positions indicating that the replication variance does not change greatly across position. Furthermore, we see that M_j does not change with read depth (except where the depth goes to zero) indicating that M_j because M_j is capturing a different process than the read depth.

The error rate across positions is captured by the M_0 parameter shown as a

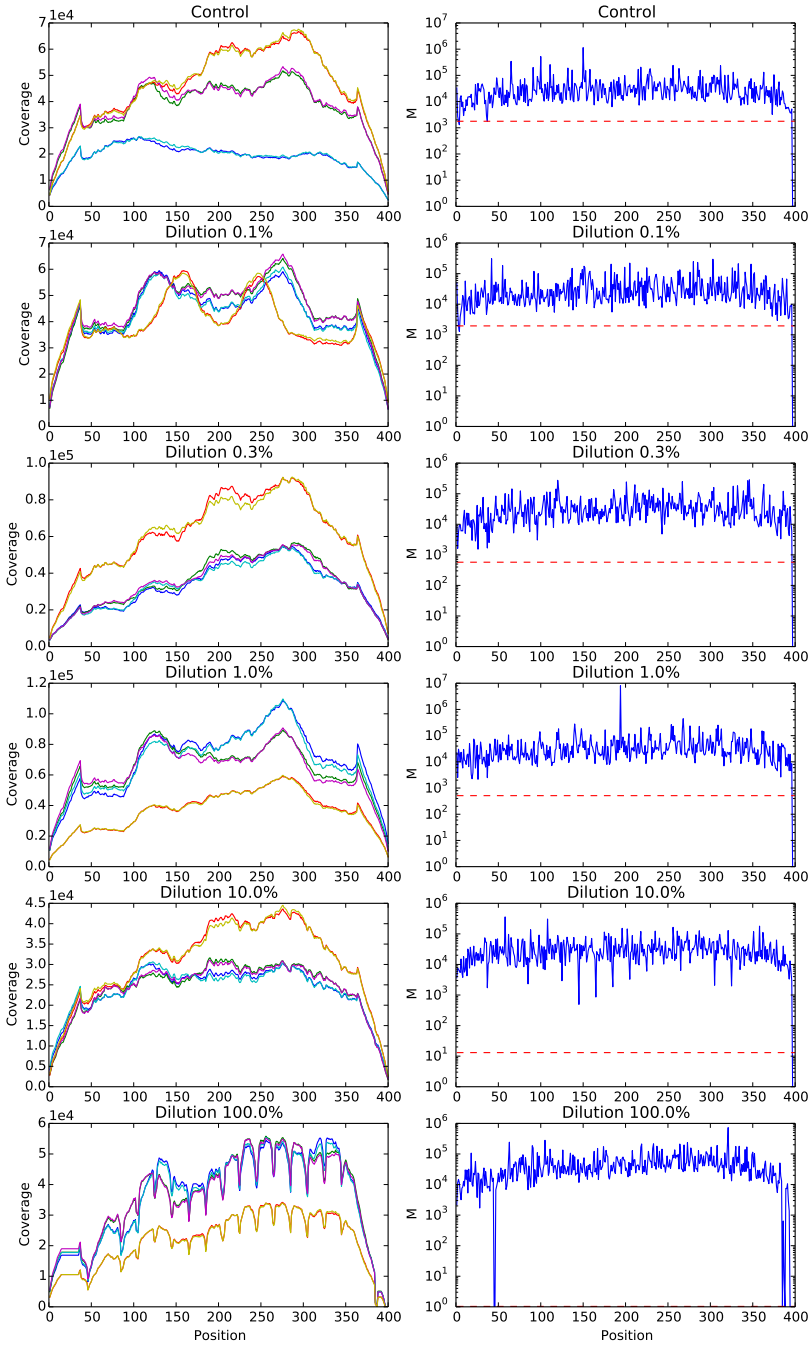


Figure D.1: Key parameters for RVD2 model for synthetic DNA data sets.

horizontal dotted line in the plots in the right column. We see that the variation between replicates is smaller than the variation between location. M_j and M_0 are precision parameters, they are inversely proportional to the variance. Where M_j is greater than M_0 the precision between replicates is higher than the precision across positions.

Appendix E

MCMC trace and autocorrelation analysis

One of the problematic attribute of a Markov chain in sampling approach is autocorrelation. Short-run autocorrelated samples are unrepresentative of the true underlying posterior distribution. In order to reduce autocorrelation among the samples, we thin the Markov chain by a factor of 2 to achieve a more precise estimate of the posterior.

In order to evaluate autocorrelation in the thinned chain, we visualize the MCMC trace, autocorrelation and Lag1 plot for Gibbs samples of θ and μ in Figure E.1. The MCMC trace indicates that θ and μ is well sampled; the autocorrelation and Lag1 plot shows that there is no significant autocorrelation in the Gibbs samples chain.

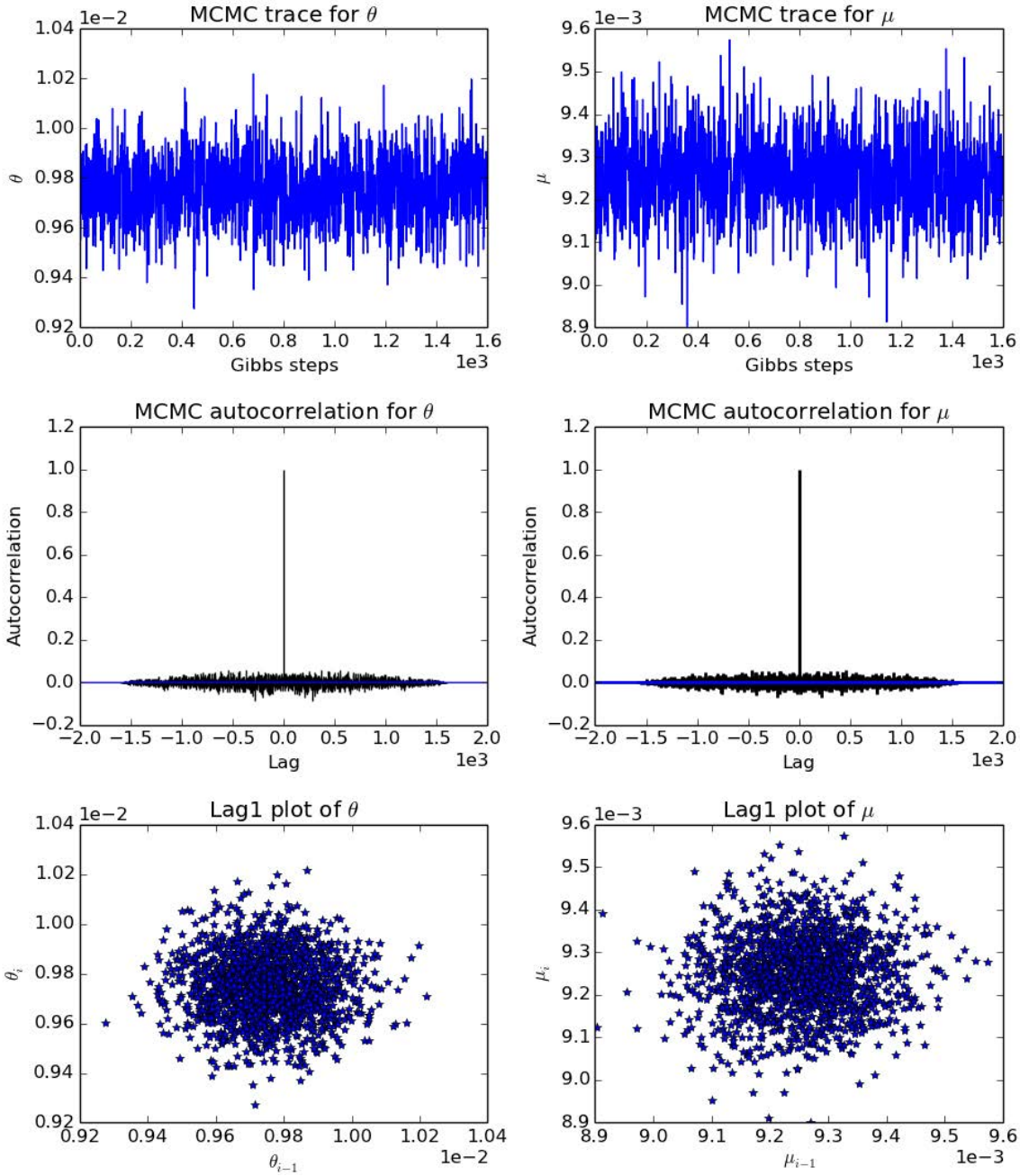


Figure E.1: MCMC traces and autocorrelation evaluation plot (sample: dilution rate at 0.3%, replicate 2, position 225).

Appendix F

ROC curve with χ^2 test

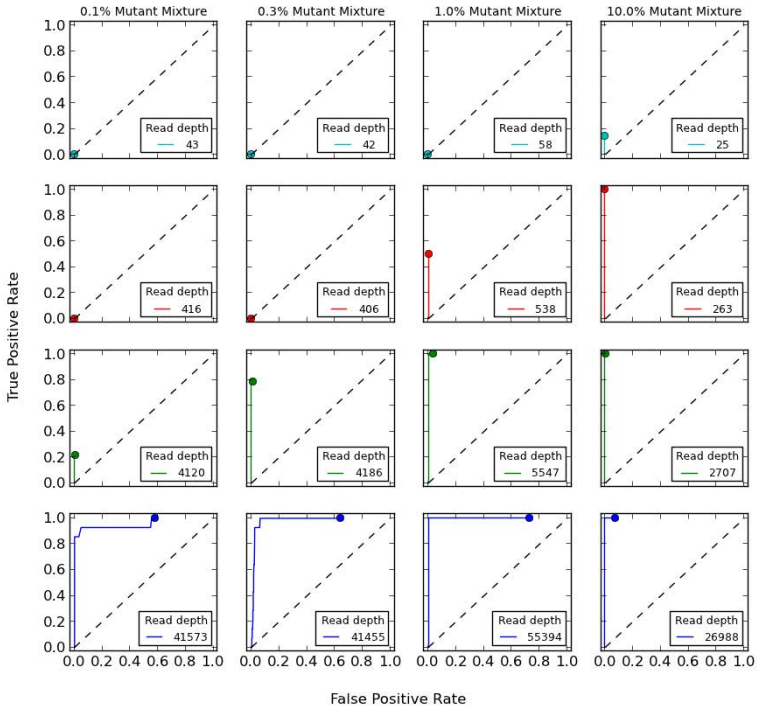


Figure F.1: ROC curve for posterior density test and χ^2 test in synthetic dataset.

Comparing to the ROC curve in Figure 4.2, where only Bayesian posterior test was evaluated, Figure F.1 shows that the χ^2 test is a very stringent test and removes most of the false positives. The χ^2 test generally guaranteed high specificity, especially at low MAF level, even though maybe at a high cost of sensitivity.

Appendix G

Somatic mutations posterior histogram

Figure G.1 and Figure G.2 shows the histograms of $\hat{\mu}_j$ for somatic mutations called by RVD2 in the 44kbp PAXIP1 gene in HCC1187 dataset

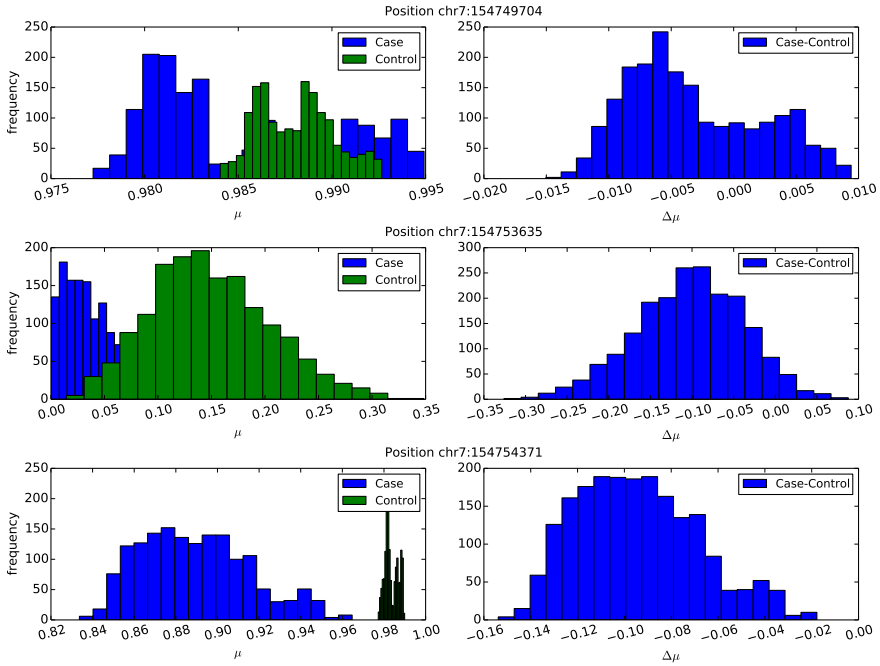


Figure G.1: Histogram of $\hat{\mu}_j$ for positions where μ^{case} is significantly lower than μ^{control}, namely $\Pr(\Delta\hat{\mu}_j < \tau) > 1 - \alpha$, where $\tau = 0, \alpha = 0.05$.

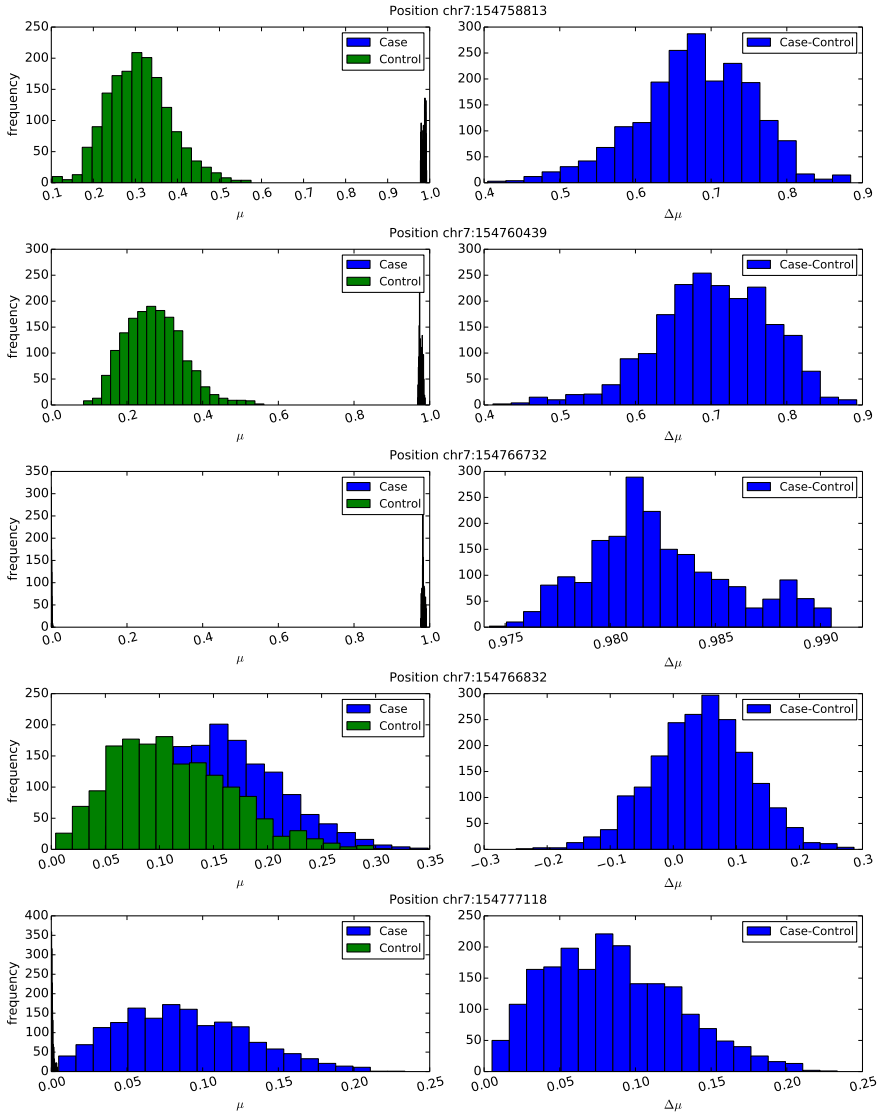


Figure G.2: Histogram of $\hat{\mu}_j$ for positions where μ^{case} is significantly higher than μ^{control}, namely $\Pr(\Delta\hat{\mu}_j > \tau) > 1 - \alpha$, where $\tau = 0, \alpha = 0.05$.

Appendix H

Parameter settings for other variant calling algorithms

Samtools We used samtools (v0.1.19) function mpileup to call variants and bcftools to save the result in standard VCF files. In mpileup, we set the -d option sufficiently high at 10^6 to avoid truncating read deapth. Option -u was enabled to make sure the output bcf files were uncompressed.

GATK We used GATK (v2.1-8) UnifiedGenotyper function to detect mutations on our synthetic data following the recommended workflow. Due to some format incompatibility, we applied Picard to format read group and GATK for realignment. In UnifiedGenotyper, -ploid (Number of samples in each pool $\times$ Sample Ploidy) was set at 1 because our synthetic data is haploid; -dcov was set at 10^6 to avoid downsampling coverage within GATK.

VarScan2-mpileup VarScan2 (v2.3.4) mpileup2snp is a SNP calling program which takes multi-samples from samtools mpileup pipeline.We assigned parameter -C value 50 as the synthetic data was aligned using BWA and set -d at 10^6. In

mpileup2snp, –min-var-freq, the only non-default parameter, was set low enough at 10^{-5} because the variant frequency can be as low as 10^{-3}.

VarScan2-somatic We tested VarScan2 somatic on our synthetic dataset. The parameter –normal-purity set was at 1.00, –tumor-purity at the dilution rate. The parameter –min-var-freq was set at 10^{-5}. We combined all the positions VarScan2-somatic called regardless the somatic status (Germline/LOH/Somatic/Unknown) to compare with performance of RVD2.

Strelka and muTect Since configuration and Analysis for Strelka and muTect is standardized and no parameter needs to be specified, we installed these two programs and ran them on our data set separately.

Samtools mpileup and GATK can accept multiple "tumor" replicates for variant calling, so we fed six bam files from each case replicate group to mpileup. VarScan2-mpileup takes multiple "tumor-normal" pair replicates so we passed six pair replicates to each algorithm. Varscan2-somatic, strelka and muTect do not accept replicate data for the "normal" or "tumor" bam files so we used a single bam file from each replicate group with a read depth that most closely matched the overall median depth of the replicates.

Appendix I

Positions Look-up chart in HCC1187 sample

Supplemetary Table 1. This table shows the sum of positions called by VarScan Somatic, RVD2, muTect and Strelka in 44kbp PAXIP1 gene from chr7:154738059 to chr7:154782774. The first column is the position index corresponding to Fig 6 in the paper. The second column is the actual position in PAXIP1 gene with respect to index. The Third column is the reference base. The forth and fifth columns are the depth matrix for Normal and Case sample repectively. The last four columns are the mutation detection status from the four algorithms. Blank cell indicates this position is not called by the algorithm in the header.

Index	Actual Position	REF	Control	Case	VarScan2 Somatic	RVD2	muTect	Strelka
Format			**A:C:G:T**	**A:C:G:T**				
1	chr7:154739981	T	0:0:0:50	0:0:2:58	Somatic			
2	chr7:154740723	T	0:0:0:60	0:0:2:69	Somatic			
3	chr7:154740848	A	37:0:0:0	56:2:0:0	Somatic			
4	chr7:154743899	C	0:0:0:45	0:0:0:71	Germline	Germline	Call	
5	chr7:154745094	G	0:0:47:0	2:0:68:0	Somatic			
6	chr7:154747295	T	0:0:0:41	0:0:2:81	Somatic			
7	chr7:154749452	T	0:0:3:29	0:0:2:37	Germline			
8	chr7:154749704	G	8:0:36:0	2:0:42:0	Germline	Germline/Somatic	Call	
9	chr7:154751731	A	53:0:0:0	73:2:0:0	Somatic			
10	chr7:154751818	T	0:0:0:57	0:1:4:75	Somatic			
11	chr7:154753635	T	0:38:0:0	0:63:0:1	Germline	Germline	Call	
12	chr7:154754218	G	0:0:49:0	0:0:65:2	Somatic			
13	chr7:154754371	T	0:19:0:31	0:61:0:0	LOH	LOH	Call	
14	chr7:154754860	T	0:0:0:49	0:0:2:61	Somatic			
15	chr7:154755810	T	0:0:0:49	0:2:0:58	Somatic			
16	chr7:154755836	A	57:0:0:0	64:3:0:0	Somatic			
17	chr7:154757232	A	40:0:0:0	54:2:0:0	Somatic			
18	chr7:154757503	T	0:0:0:50	0:0:2:69	Somatic			
19	chr7:154757988	A	57:0:0:0	69:3:0:0	Somatic			
20	chr7:154758813	G	14:0:28:0	54:0:0:0	LOH	LOH	Call	
21	chr7:154759833	T	0:0:0:48	0:0:2:62	Somatic			
22	chr7:154760439	A	37:0:0:0	0:38:0:0	Somatic	Somatic	Call	Somatic
23	chr7:154760538	G	0:0:62:0	0:0:42:2	Somatic			
24	chr7:154760592	T	0:0:0:47	0:0:2:38	Somatic			
25	chr7:154760658	T	0:0:0:52	0:0:2:47	Somatic			
26	chr7:154760790	A	55:0:0:0	66:2:0:0	Somatic			
27	chr7:154762296	T	0:0:0:51	0:0:2:74	Somatic			
28	chr7:154762409	T	0:0:0:32	0:0:2:51	Somatic			
29	chr7:154762415	A	27:2:0:0	47:3:0:0	Germline			
30	chr7:154762957	T	0:0:0:37	2:0:0:54	Somatic			
31	chr7:154763136	A	61:0:0:0	62:0:2:0	Somatic			
32	chr7:154763337	T	0:0:0:43	0:0:2:80	Somatic			
33	chr7:154766365	A	53:0:0:0	57:2:0:0	Somatic			
34	chr7:154766388	A	47:0:0:0	66:2:0:0	Somatic			
35	chr7:154766700	C	4:21:0:0	10:42:0:0		Germline	Call	
36	chr7:154766732	T	0:0:0:35	0:0:5:44	Somatic	Somatic	Call	
37	chr7:154766832	A	34:0:0:0	42:4:0:0	Somatic	Somatic		

Index	Actual Position	REF	Control	Case	VarScan2 Somatic	RVD2	muTect	Strelka
38	chr7:154766834	T	0:0:0:34	0:2:0:43	Somatic			
39	chr7:154766857	A	29:0:0:0	36:2:0:0	Somatic			
40	chr7:154767108	T	0:0:0:55	0:0:1:64	Somatic			
41	chr7:154767456	T	0:0:0:40	0:0:2:74	Somatic			
42	chr7:154767801	A	47:0:0:0	67:2:0:0	Somatic			
43	chr7:154768640	T	0:0:0:66	0:0:2:97	Somatic			
44	chr7:154769094	G	0:0:46:0	0:0:63:2	Somatic			
45	chr7:154771488	T	0:0:0:50	0:0:2:60	Somatic			
46	chr7:154772261	A	47:0:0:0	71:2:0:0	Somatic			
47	chr7:154775220	A	47:0:0:0	84:2:1:0	Somatic			
48	chr7:154775236	T	0:0:0:47	0:0:2:88	Somatic			
49	chr7:154775872	A	55:0:0:0	71:2:0:1	Somatic			
50	chr7:154777014	C	0:49:0:0	0:64:2:0	Somatic		Call	
51	chr7:154777118	A	46:0:0:0	31:4:0:0	Somatic	Somatic	Call	
52	chr7:154777687	G	0:0:45:0	0:0:50:2	Somatic			
53	chr7:154777955	T	0:0:0:52	0:0:2:47	Somatic			
54	chr7:154778892	A	55:0:0:0	44:2:0:0	Somatic			
55	chr7:154779690	A	45:0:0:0	52:2:0:0	Somatic			
56	chr7:154780960	C	0:0:0:42	0:0:0:56	Germline	Germline	Call	
57	chr7:154781700	A	55:3:0:0	82:2:0:0	Germline			
58	chr7:154781769	G	0:0:21:4	0:0:31:2	LOH	Germline		
59	chr7:154781944	T	0:0:0:55	0:0:2:47	Somatic			
60	chr7:154782120	A	47:0:0:0	42:2:0:0	Somatic			
61	chr7:154782770	A	58:0:0:0	91:2:0:0	Somatic			

Appendix J

Variational Inference Derivation

J.1 Factorization

We firstly fully factorize the exact posterior distribution using the naive mean-field method,

$$q(\mu, \theta) = q(\mu)q(\theta) = \prod_{j=1}^{J} q(\mu_j) \prod_{i=1}^{N} q(\theta_{ji}). \tag{J.1}$$

J.2 Writing out the ELBO

As shown in Section 2.3.2, the ELBO $\mathcal{L}(q, \phi)$ is

$$\begin{aligned}\mathcal{L}(q, \phi) = {} & E_q \left[\log p \left(r|\theta, n\right)\right] + E_q \left[\log p \left(\theta|\mu; M\right)\right] + E_q \left[\log p \left(\mu; \mu_0, M_0\right)\right] \\ & - E_q \left[\log q \left(\mu\right)\right] - E_q \left[\log q \left(\theta\right)\right]\end{aligned}$$

Writing out each component, we have

$$
\begin{aligned}
E_q\left[\log p\left(r|\theta,n\right)\right] &= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log p\left(r_{ji}|\theta_{ji},n_{ji}\right)\right] \\
&= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log\left(\frac{\Gamma(n_{ji}+1)}{\Gamma(r_{ji}+1)\Gamma(n_{ji}-r_{ji}+1)}\theta_{ji}^{r_{ji}}(1-\theta_{ji})^{n_{ji}-r_{ji}}\right)\right] \\
&= \sum_{j=1}^{J}\sum_{i=1}^{N} \log\left(\frac{\Gamma(n_{ji}+1)}{\Gamma(r_{ji}+1)\Gamma(n_{ji}-r_{ji}+1)}\right) \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{r_{ji}E_q\left[\log\theta_{ji}\right] + (n_{ji}-r_{ji})E_q\left[\log(1-\theta_{ji})\right]\right\}
\end{aligned}
$$

$$
\begin{aligned}
E_q\left[\log p\left(\theta|\mu;M\right)\right] &= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log p\left(\theta_{ji}|\mu_j;M_j\right)\right] \\
&= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\theta_{ji}^{M_j\mu_j-1}(1-\theta_{ji})^{M_j(1-\mu_j)-1}\right)\right] \\
&= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right] \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{E_q\left[(M_j\mu_j-1)\log\theta_{ji}\right] + E_q\left[(M_j(1-\mu_j)-1)\log\left(1-\theta_{ji}\right)\right]\right\} \\
&= N * \sum_{j=1}^{J} E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right] \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{(M_j E_q\left[\mu_j\right]-1)E_q\left[\log\theta_{ji}\right]\right\} \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{(M_j(1-E_q[\mu_j])-1)\,E_q\left[\log\left(1-\theta_{ji}\right)\right]\right\}
\end{aligned}
$$

$$
\begin{aligned}
E_q\left[\log p\left(\mu;\mu_0,M_0\right)\right] &= \sum_{j=1}^{J} E_q\left[\log p\left(\mu_j;\mu_0,M_0\right)\right] \\
&= \sum_{j=1}^{J} E_q\left[\log\left(\frac{\Gamma(M_0)}{\Gamma(\mu_0 M_0)\Gamma(M_0(1-\mu_0))}\mu_j^{M_0\mu_0-1}(1-\mu_j)^{M_0(1-\mu_0)-1}\right)\right] \\
&= J * \log\frac{\Gamma(M_0)}{\Gamma(\mu_0 M_0)\Gamma(M_0(1-\mu_0))} \\
&\quad + \sum_{j=1}^{J}\left\{(M_0\mu_0-1)E_q\left[\log\mu_j\right] + (M_0(1-\mu_0)-1)E_q\left[\log(1-\mu_j)\right]\right\}
\end{aligned}
$$

Therefore, in order to compute ELBO, we need to obtain the following expectations with respect to variational distribution: $E_q\left[\log\theta_{ji}\right]$, $E_q\left[\log\left(1-\theta_{ji}\right)\right]$, $E_q\left[\log\mu_j\right]$, $E_q\left[\log(1-\mu_j)\right]$, $E_q\left[\mu_j\right]$ and $E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right]$.

J.3 Variational Expectation (E-step)

J.3.1 Variational distributions

Variational distributions for θ_{ji} Section 2.2.2 gives the posterior conditional distribution of θ given other variables,

$$
p(\theta_{ji}|r_{ji},n_{ji},\mu_j,M_j) \sim \text{Beta}\left(r_{ji}+M_j\mu_j, n_{ji}-r_{ji}+M_j(1-\mu_j)\right). \tag{J.2}
$$

As posterior conditional distribution of θ_{ji} is a Beta distribution, the optimal variational distribution is Beta distribution. Therefore, we propose to use Beta distribution with parameter vector δ_{ji} as variational distribution.

$$
q(\theta_{ji}) \sim \text{Beta}\left(\delta_{ji}\right). \tag{J.3}
$$

Variational distributions for μ_j Section 2.2.3 provides the posterior distribution of μ_j given its Markov blanket

$$p(\mu_j|\theta_{ji}, M_j, \mu_0, M_0) \propto p(\mu_j|\mu_0, M_0)p(\theta_{ji}|\mu_j, M_j). \tag{J.4}$$

Posterior conditional distribution of μ_j is a product of two Beta distributions, which is not in the form of any known distribution. Therefore, we propose two different variational distribution to approximate true posterior distribution of μ_j.

The first proposal variational distribution is Beta distribution with parameter vector γ_{ji},

$$q(\mu_j) \sim \text{Beta}\left(\gamma_j\right), \tag{J.5}$$

as it would greatly simplify the variational deviation process.

The second proposal variational distribution is Laplace approximation distribution [72].

$$\mu_j \sim \mathcal{N}(\hat{\mu}_j, -f''(\hat{\mu}_j)^{-1}) \tag{J.6}$$

Where $\hat{\mu}_j$ is the stationary point which maximize the ELBO, and $f(\mu_j)$ isolates all the terms involves μ_j. As μ_j is within $[0, 1]$, μ_j follows a truncated normal distribution.

Isolating items in the ELBO that depends on μ_j gives

$$
\begin{aligned}
f(\mu_j) = & -N \log \Gamma(\mu_j M_j) - N \log \Gamma(M_j(1-\mu_j)) \\
& + \sum_{i=1}^{N} M_j \mu_j \left\{ E_q \left[\log \theta_{ji} \right] - E_q \left[\log \left(1 - \theta_{ji} \right) \right] \right\} \\
& + \left\{ (M_0 \mu_0 - 1) \log \mu_j + (M_0(1-\mu_0) - 1) \log(1-\mu_j) \right\}
\end{aligned}
\tag{J.7}
$$

Take the first derivative with respect to μ_j,

$$
\begin{aligned}
f'(\mu_j) = & -N M_j \psi(\mu_j M_j) + N M_j \psi(M_j(1-\mu_j)) \\
& + \sum_{i=1}^{N} M_j \left\{ E_q \left[\log \theta_{ji} \right] - E_q \left[\log \left(1 - \theta_{ji} \right) \right] \right\} \\
& + \left\{ \frac{M_0 \mu_0 - 1}{\mu_j} - \frac{M_0(1-\mu_0) - 1}{1-\mu_j} \right\}
\end{aligned}
\tag{J.8}
$$

The second derivative with respect to μ_j is,

$$
\begin{aligned}
f''(\mu_j) = & -N M_j^2 \psi'(\mu_j M_j) - N M_j^2 \psi'(M_j(1-\mu_j)) \\
& + \left\{ \frac{1 - M_0 \mu_0}{\mu_j^2} - \frac{M_0(1-\mu_0) - 1}{(1-\mu_j)^2} \right\}
\end{aligned}
\tag{J.9}
$$

where $\psi(\mu)$ is the Digamma function, and $\psi'(\mu) = \frac{\partial \psi(\mu)}{\partial \mu}$ is the Trigamma function.

J.3.2 E-step using Beta-Beta variational distribution

As stated in Section J.3.1, the first set of proposal variational distribution families are

$$\theta_{ji} \sim \text{Beta}(\delta_{ji})$$
$$\mu_j \sim \text{Beta}(\gamma_j)$$

Given the variational distributions we have

$$\begin{aligned}
E_q\left[\log \theta_{ji}\right] &= \psi(\delta_{ji1}) - \psi(\delta_{ji1} + \delta_{ji2}) \\
E_q\left[\log\left(1 - \theta_{ji}\right)\right] &= \psi(\delta_{ji2}) - \psi(\delta_{ji1} + \delta_{ji2}) \\
E_q\left[\mu_j\right] &= \frac{\gamma_{j1}}{\gamma_{j1} + \gamma_{j2}} \\
E_q\left[\log \mu_j\right] &= \psi(\gamma_{j1}) - \psi(\gamma_{j1} + \gamma_{j2}) \\
E_q\left[\log(1 - \mu_j)\right] &= \psi(\gamma_{j2}) - \psi(\gamma_{j1} + \gamma_{j2})
\end{aligned} \tag{J.10}$$

There is no analytical representation for $E_q\left[\log \Gamma(\mu_j M_j)\right]$ and $E_q\left[\log \Gamma(M_j(1 - \mu_j))\right]$. Therefore, we propose to use trapezoidal numerical integration to approximate these two expectations.

Moreover, according to the entropy of beta distribution random variable,

$$\begin{aligned}
E_q\left[\log q\left(\mu\right)\right] &= \sum_{j=1}^{J} E_q\left[\log q(\mu_j)\right] \\
&= -\sum_{j=1}^{J} \left\{\log(B(\gamma_{j1}, \gamma_{j2})) - (\gamma_{j1} - 1)\psi(\gamma_{j1}) - (\gamma_{j2} - 1)\psi(\gamma_{j2})\right\} \\
&\quad + \left\{(\gamma_{j1} + \gamma_{j2} - 2)\psi(\gamma_{j1} + \gamma_{j2})\right\}
\end{aligned} \tag{J.11}$$

$$
\begin{aligned}
E_q\left[\log q\left(\theta\right)\right] &= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q\left[\log q(\theta_{ji})\right] \\
&= -\sum_{j=1}^{J}\sum_{i=1}^{N}\left\{\log(B(\delta_{ji1},\delta_{ji2})) + (\delta_{ji1}+\delta_{ji2}-2)\psi(\delta_{ji1}+\delta_{ji2})\right\} \quad \text{(J.12)} \\
&\quad + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{(\delta_{ji1}-1)\psi(\delta_{ji1}) + (\delta_{ji2}-1)\psi(\delta_{ji2})\right\}
\end{aligned}
$$

where function $B(x_1, x_2)$ is Beta function.

J.3.3 E-step using Beta-Laplace variational distribution

From attributes of distribution $\theta_{ji} \sim \text{Beta}(\delta_{ji})$ we have

$$
\begin{aligned}
E_q\left[\log \theta_{ji}\right] &= \psi(\delta_{ji1}) - \psi(\delta_{ji1}+\delta_{ji2}) \\
E_q\left[\log\left(1-\theta_{ji}\right)\right] &= \psi(\delta_{ji2}) - \psi(\delta_{ji1}+\delta_{ji2})
\end{aligned}
$$

From attributes of truncated normal distribution $\mu_j \sim \mathcal{N}(\hat{\mu}_j, -f''(\hat{\mu}_j)^{-1}), \mu_j \in [0,1]$, assuming $\gamma_{j2} = \frac{0-\hat{\mu}_j}{\sqrt{-f''(\hat{\mu}_j)^{-1}}} = \frac{-\hat{\mu}_j}{\sqrt{-f''(\hat{\mu}_j)^{-1}}}$, $\gamma_{j1} = \frac{1-\hat{\mu}_j}{\sqrt{-f''(\hat{\mu}_j)^{-1}}}$, $Z = \Phi(\gamma_{j1}) - \Phi(\gamma_{j2})$,

We have

$$
E_q\left[\mu_j\right] = \hat{\mu}_j + \frac{\phi(\gamma_{j2}) - \phi(\gamma_{j1})}{Z}\sqrt{-f''(\hat{\mu}_j)^{-1}} \tag{J.13}
$$

Here, $\phi(x) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}x^2)$ is the probability density function of the standard normal distribution and $\Phi(\cdot)$ is its cumulative distribution function.

We are not able to analytically compute $E_q\left[\log\Gamma(\mu_j M_j)\right]$, $E_q\left[\log\Gamma(M_j(1-\mu_j))\right]$,

$E_q[\log \mu_j]$ and $E_q[\log(1-\mu_j)]$. Considering μ_j is bounded in $[0,1]$, we propose to use trapezoidal numerical integration to approximate these expectations.

Moreover, according to the entropy of beta distribution random variable,

$$\begin{aligned} E_q[\log q(\mu)] &= \sum_{j=1}^{J} E_q[\log q(\mu_j)] \\ &= \sum_{j=1}^{J} \left\{ \log(\sqrt{-2\pi e f''(\hat{\mu}_j)^{-1}} Z) + \frac{\gamma_{j2}\phi(\gamma_{j2}) - \gamma_{j1}\phi(\gamma_{j1})}{2Z} \right\} \end{aligned} \tag{J.14}$$

$$\begin{aligned} E_q[\log q(\theta)] &= \sum_{j=1}^{J}\sum_{i=1}^{N} E_q[\log q(\theta_{ji})] \\ &= -\sum_{j=1}^{J}\sum_{i=1}^{N} \{\log(B(\delta_{ji1}, \delta_{ji2})) - (\delta_{ji1}-1)\psi(\delta_{ji1}) \\ &- (\delta_{ji2}-1)\psi(\delta_{ji2}) + (\delta_{ji1}+\delta_{ji2}-2)\psi(\delta_{ji1}+\delta_{ji2})\} \end{aligned} \tag{J.15}$$

J.4 Optimizing Model Parameters $\phi = \{\mu_0, M_0, M\}$ (M-step)

J.4.1 Optimizing μ_0

The ELBO with respect to μ_0 is

$$\begin{aligned} \mathcal{L}_{[\mu_0]} = &-J * \log\Gamma(\mu_0 M_0) - J * \log\Gamma(M_0(1-\mu_0)) \\ &+ M_0\mu_0 \sum_{j=1}^{J} \{E_q[\log \mu_j] - E_q[\log(1-\mu_j)]\}. \end{aligned} \tag{J.16}$$

Take the derivative with respect to μ_0 and set it equal to zero,

$$\begin{aligned}\mathcal{L}'_{[\mu_0]} &= -J * M_0\psi(\mu_0 M_0) + J * M_0\psi(M_0(1-\mu_0)) \\ &\quad + M_0 \sum_{j=1}^{J} \{E_q[\log \mu_j] - E_q[\log(1-\mu_j)]\} \\ &= 0.\end{aligned} \tag{J.17}$$

The update for μ_0 can be numerically computed.

J.4.2 Optimizing M_0

The ELBO with respect to M_0 is

$$\mathcal{L}_{[M_0]} = J * \log \frac{\Gamma(M_0)}{\Gamma(\mu_0 M_0)\Gamma(M_0(1-\mu_0))} + M_0 \sum_{j=1}^{J} \{\mu_0 E_q[\log \mu_j] + (1-\mu_0)E_q[\log(1-\mu_j)]\} \tag{J.18}$$

Take the derivative with respect to M_0 and set it equal to zero,

$$\begin{aligned}\mathcal{L}'_{[M_0]} &= \log \frac{\Gamma(M_0)}{\Gamma(\mu_0 M_0)\Gamma(M_0(1-\mu_0))} + M_0 \sum_{j=1}^{J} \{\mu_0 E_q[\log \mu_j] + (1-\mu_0)E_q[\log(1-\mu_j)]\} \\ &= \psi(M_0) - \mu_0\psi(\mu_0 M_0) - (1-\mu_0)\psi(M_0(1-\mu_0)) \\ &+ \sum_{j=1}^{J} \{\mu_0 E_q[\log \mu_j] + (1-\mu_0)E_q[\log(1-\mu_j)]\} \\ &= 0\end{aligned} \tag{J.19}$$

the update for M_0 can be numerically computed.

J.4.3 Optimizing M

$$\begin{aligned}\mathcal{L}_{[M]} = N \sum_{j=1}^{J} E_q \left[\log \left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))} \right) \right] \\ + \sum_{j=1}^{J} \sum_{i=1}^{N} M_j \left\{ E_q\left[\mu_j\right] E_q\left[\log \theta_{ji}\right] + \left(1 - E_q\left[\mu_j\right]\right) E_q\left[\log\left(1-\theta_{ji}\right)\right] \right\}\end{aligned} \tag{J.20}$$

$$f(\mu) = \log \left(\frac{\Gamma(M)}{\Gamma(\mu M)\Gamma(M(1-\mu))} \right)$$

then

$$\begin{aligned} f'(\mu) &= -M\psi(\mu M) + M\psi(M(1-\mu)) \\ f''(\mu) &= -M^2\psi'(\mu M) - M^2\psi'(M(1-\mu)) < 0 \end{aligned} \tag{J.21}$$

where $\psi(\mu)$ is the Digamma function, and $\psi'(\mu) = \frac{\partial \psi(\mu)}{\partial \mu}$ is the Trigamma function. As trigamma function $\psi'(\mu)$ is positive, $f''(\mu)$ is negative. Thus, $f(\mu)$ is a concave function. We can approximate $f(\mu)$ using first-order Taylor expansion around point μ°, which is

$$\begin{aligned} f(\mu) &\leq f(\mu^\circ) + f'(\mu^\circ) \cdot (\mu - \mu^\circ) \\ &= \log \left(\frac{\Gamma(M)}{\Gamma(\mu^\circ M)\Gamma(M(1-\mu^\circ))} \right) + (-M\psi(\mu^\circ M) + M\psi(M(1-\mu^\circ))) \cdot (\mu - \mu^\circ). \end{aligned}$$

A upper bound approximation for $E_q \left[\log \left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))} \right) \right]$ around point μ_j°

can be represented as

$$E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right] \leq \log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j^\circ M_j)\Gamma(M_j(1-\mu_j^\circ))}\right) + \left(-M_j\psi(\mu_j^\circ M_j) + M_j\psi(M_j(1-\mu_j^\circ))\right)\cdot(E_q(\mu_j)-\mu_j^\circ).$$

The equality holds if and only if $\mu_j^\circ = E_q(\mu_j)$. Therefore, at this particular point,

$$E_q\left[\log\left(\frac{\Gamma(M_j)}{\Gamma(\mu_j M_j)\Gamma(M_j(1-\mu_j))}\right)\right] = \log\left(\frac{\Gamma(M_j)}{\Gamma(E_q(\mu_j)M_j)\Gamma(M_j(1-E_q(\mu_j)))}\right).$$

Then

$$\begin{aligned}\mathcal{L}_{[M]} = N\sum_{j=1}^{J}\log\left(\frac{\Gamma(M_j)}{\Gamma(E_q(\mu_j)M_j)\Gamma(M_j(1-E_q(\mu_j)))}\right) \\ + \sum_{j=1}^{J}\sum_{i=1}^{N} M_j\left\{E_q\left[\mu_j\right]E_q\left[\log\theta_{ji}\right] + \left(1-E_q\left[\mu_j\right]\right)E_q\left[\log\left(1-\theta_{ji}\right)\right]\right\}\end{aligned} \tag{J.22}$$

The partial derivative is

$$\begin{aligned}\frac{\partial\mathcal{L}_{[M]}}{\partial\ M_j} = \psi(M_j) - E_q(\mu_j)\psi(E_q(\mu_j)M_j) - (1-E_q(\mu_j))\psi((1-E_q(\mu_j))M_j) \\ + \sum_{j=1}^{J}\sum_{i=1}^{N}\left\{E_q\left[\mu_j\right]E_q\left[\log\theta_{ji}\right] + \left(1-E_q\left[\mu_j\right]\right)E_q\left[\log\left(1-\theta_{ji}\right)\right]\right\}\end{aligned} \tag{J.23}$$

The update for M_j can be numerically computed.

Appendix K

RVD2 python source code (MCMC sampling approach)

RVD2 Source Code in Python

```
#!/usr/bin/env python

"""rvd27.py: Compute MAP estimates for RVD2.7 model."""

from __future__ import print_function
from __future__ import division

import numpy as np

import scipy.stats as ss
from scipy.special import gammaln

import logging
import multiprocessing as mp
from itertools import repeat
import h5py
import tempfile

import sys
import os
import subprocess
from datetime import date

import re
import pdb
import time
def main():
    import argparse

    # Populate our options, -h/--help is already there.
    description = """
                    RVD is a hierarchical bayesian model for identifying
                    rare variants from short-read sequence data. """

    # create the top-level parser
    argp = argparse.ArgumentParser(prog='rvd', description=description)
    argp.add_argument('--version', action='version', version='%(prog)s 2.7')
    argp.add_argument('-v', '--verbose', dest='verbose', action='count',
                help="increase verbosity (specify multiple times for more)")

    # argp.add_argument('cmd', action='store', nargs='*',
    #             choices=['gen', 'gibbs'])
    subparsers = argp.add_subparsers(help='sub-command help')

    # create subparser for gibbs fitting
    argpGibbs = subparsers.add_parser('gibbs',
                        help='fit the RVD model using Gibbs sampling')
```

```
argpGibbs.add_argument('dcfile', nargs='+',
                    help='depth chart file name')
argpGibbs.add_argument('-o', dest='outputFile',
            default='output.hdf5',
            help='output HDF5 file name')
argpGibbs.add_argument('-p', '--pool', type=int, default=None,
            help='number of workers in multithread pool')
argpGibbs.set_defaults(func=gibbs)

# create subparser to compare two model files
argpTest = subparsers.add_parser('test_main',
                    help='RVD2 algorithm to find mutations in tumor-normal
                        -paired sample. A posteriror difference test and a
                        somatic test will be done in this program.')

argpTest.add_argument('alpha', type=float, default=0.95,
            help='poseterior difference test probability threshold')
argpTest.add_argument('controlHDF5Name', default=None,
            help='control model file (HDF5)')
argpTest.add_argument('caseHDF5Name', default=None,
            help='case model file (HDF5)')

argpTest.add_argument('somatic_tau', default=(0.05,0.95),
            help='Two thresholds for somatic test. \
            Mu lower than the lower threshold will be classified as
                reference (non-mutation), \
            Mu between the two thresholds will be classified as
                heterozygote, \
            Mu higher than the higher threshold will be classified as
                homozygote')

argpTest.add_argument('diffroi', default=(0,np.inf),
            help='region of interest in posterior differece distribution (
                tuple as interval)')

argpTest.add_argument('-N', type=int, default=1000,
            help='Monte-Carlo sample size (default=1000)')

argpTest.add_argument('-o', '--output', dest='outputFile', nargs='?',
            default=None)

argpTest.set_defaults(func=test_main)

# create subparser to sample the model
argpGen = subparsers.add_parser('gen',
                    help='sample data from the RVD model')
argpGen.add_argument('input', nargs='+')
argpGen.add_argument('-o', '--output', dest='outputFile', nargs='?',
            default='output.hdf5')

# Parse the arguments (defaults to parsing sys.argv)
args = argp.parse_args()
```

```
    # TODO check what came in on the command line and call optp.error("Useful
        message") to exit if all is not well

    log_level = logging.WARNING  # default
    if args.verbose == 1:
        log_level = logging.INFO
    elif args.verbose >= 2:
        log_level = logging.DEBUG

    # Set up basic configuration, out to stderr with a reasonable format
    logging.basicConfig(level=log_level,
                        format='%(levelname)s:%(module)s:%(message)s')

    # Do actual work here
    args.func(args)


def gibbs(args):
    """ Top-level function to use gibbs sampling on a set of depth chart files
    """
    (r, n, loc, refb) = load_depth(args.dcfile)
    (phi, theta_s, mu_s) = mh_sample(r, n)
    save_model(args.outputfile, phi, mu=mu_s, theta=theta_s, r=r, n=n, loc=loc
        ,
            refb=refb)

def test_main(args):
    test(args.alpha, args.controlHDF5Name, args.caseHDF5Name,
         args.somatic_tau, args.diffroi,
         args.N, args.outputFile)

def test(somatic_alpha=0.95, germline_alpha=0.85, controlHDF5Name=None,
    caseHDF5Name=None,
    somatic_tau=0, germline_tau=0.05, N=1000, outputFile=None):

    if outputfile == None:
        germline_test(controlHDF5Name, caseHDF5Name, germline_alpha,
            germline_tau)
        somatic_test(controlHDF5Name, caseHDF5Name, somatic_alpha, somatic_tau
            , N)
    else:
        germline_test(controlHDF5Name, caseHDF5Name, germline_alpha,
            germline_tau, outputFile)
        somatic_test(controlHDF5Name, caseHDF5Name, somatic_alpha, somatic_tau
            , N, outputFile)


def germline_test(controlHDF5Name, caseHDF5Name, alpha = 0.15, tau = 0.05,
    outputFile='germlinecalltable.vcf'):
    # only test if control sample is mutated
    germline_roi = [tau, np.inf]
```

```
    loc, refb, altb, controlMu, controlMu0, controlR, controlN, \
             caseMu, caseMu0, caseR, caseN = load_dualmodel(controlHDF5Name,
                 caseHDF5Name)

    logging.debug('Running germline test on posterior distribution of sample %
        s and %s)'\
                    %(controlHDF5Name, caseHDF5Name))

    ## chi2 test for case and control sample independently
    J =len(controlR)

    postP = bayes_test(controlMu, [germline_roi], type = 'close')
    bayescall = postP > 1-alpha

    # chi2 test on  sample
    controlchi2call , chi2P = chi2combinetest(controlR, controlN, bayescall)

    call = np.logical_and(bayescall, controlchi2call)

    altb = ['.' for b in altb]

    write_dualvcf(outputFile,loc, call, refb, altb, np.mean(controlMu, axis=1)
        , \
                np.median(controlR,0), controlN, \
                np.mean(caseMu, axis=1), np.median(caseR,0), caseN, tau,
                    alpha)


    h5Filename = 'germlinecall.hdf5'

    h5file = h5py.File(h5Filename, 'w')

    h5file.create_dataset('call', data=call)
    h5file.create_dataset('refb', data=refb)
    h5file.create_dataset('loc', data=loc,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('controlMu', data=controlMu,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('caseMu', data=caseMu,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('controlN', data=controlN,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('caseN', data=caseN,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.close()
    return loc, call, controlMu, caseMu, controlN, caseN


def somatic_test(controlHDF5Name, caseHDF5Name, alpha=0.05, tau= 0, N=1000,
    outputFile='somaticcalltable.vcf'):
    # Two sided difference test. Somatic status will be more specific
        classified. Threshold hold and alpha should be assign
```

```
    logging.debug('Running two sided posterior somatic test on sample %s and %
        s)'%(controlHDF5Name, caseHDF5Name))

    loc, refb, altb, controlMu, controlMu0, controlR, controlN, \
             caseMu, caseMu0, caseR, caseN = load_dualmodel(controlHDF5Name,
                 caseHDF5Name)

    somatic_roi = [tau, np.inf]

    # test if caseMu is significantly higher than controlMu
    [call1, _ , _, _, _] = one_side_diff_test(alpha, loc, refb, altb,
        controlMu, controlMu0, controlR, controlN, \
             caseMu, caseMu0, caseR, caseN, N, somatic_roi)

    # test if controlMu is significantly higher than caseMu
    [call2, _ , _, _, _] = one_side_diff_test(alpha, loc, refb, altb, caseMu,
        caseMu0, caseR, caseN, \
             controlMu, controlMu0, controlR, controlN, N, somatic_roi)

    call = [call1,call2]
    call = np.sum(call,0) > 0

    # write_dualvcf(outputFile,loc, call, refb, altb, np.mean(controlMu, axis
        =1), np.median(controlR,0), controlN, \
    #               np.mean(caseMu, axis=1), np.median(caseR,0), caseN, tau,
        alpha)

    write_vcf(outputFile, loc, call, refb, altb, np.mean(caseMu, axis=1),  np.
        mean(controlMu, axis=1))

    h5Filename = 'somaticcall.hdf5'

    h5file = h5py.File(h5Filename, 'w')

    h5file.create_dataset('call', data=call)
    h5file.create_dataset('refb', data=refb)
    h5file.create_dataset('loc', data=loc,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('controlMu', data=controlMu,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('caseMu', data=caseMu,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('controlN', data=controlN,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.create_dataset('caseN', data=caseN,
                          chunks=True, fletcher32=True, compression='gzip')
    h5file.close()

    return loc, call, controlMu, caseMu, controlN, caseN


def disparity_test(controlHDF5Name, caseHDF5Name, alpha=0.05, tau= 0, N=1000,
```

```
 outputFile='somaticcalltable.vcf'):
    # Oneside posterior difference test in case when somatic test is not
        appropriate. Also compatible to synthetic data
    # The function will test if the caseMu is significantly higher than
        controlMu.
    # If we want to test if controlMu is significantly higher than caseMu, we
        need to switch the place for controlHDF5Name and caseHDF5Name
    # since chi2 test will only apply to the second HDF5 file.

    logging.debug('Running posterior disparity test on whether local mutation
        rate in sample %s is significantly higher than sample %s)'%(
        caseHDF5Name, controlHDF5Name))

    loc, refb, altb, controlMu, controlMu0, controlR, controlN, \
             caseMu, caseMu0, caseR, caseN = load_dualmodel(controlHDF5Name,
                 caseHDF5Name)

    disroi = [tau, np.inf]

    call, chi2call , chi2P, bayescall, postP = one_side_diff_test(alpha, loc,
        refb, altb, controlMu, controlMu0, controlR, controlN, \
             caseMu, caseMu0, caseR, caseN, N, disroi)

    write_dualvcf(outputFile,loc, call, refb, altb, np.mean(controlMu, axis=1)
        , np.median(controlR,0), controlN, \
                   np.mean(caseMu, axis=1), np.median(caseR,0), caseN, tau,
                       alpha)

    # write_vcf(outputFile, loc, call, refb, altb, np.mean(caseMu, axis=1), np
        .mean(controlMu, axis=1))

    return loc, call, controlMu, caseMu, controlN, caseN,  chi2call , chi2P,
        bayescall, postP

def one_side_diff_test(alpha, loc, refb, altb, Mu1, Mu01, R1, N1, Mu2, Mu02,
    R2, N2, N, diffroi):

    (Z, MuS2, MuS1) = sample_post_diff(Mu2-Mu02, Mu1-Mu01, N)
    # (Z, MuS2, MuS1) = sample_post_diff(Mu2, Mu1, N)

    if len(np.shape(diffroi))==1:
        diffroi = [diffroi]
    postP = bayes_test(Z, diffroi,'open')

    bayescall = postP > 1-alpha

    # chi2 test on  sample
    chi2call , chi2P = chi2combinetest(R2,N2,bayescall)

    call = np.logical_and(bayescall, chi2call)

    return call, chi2call , chi2P, bayescall, postP
```

```
def chi2combinetest(R, N, bayescall = 1, pvalue = 0.05):

    nRep = R.shape[0]
    J = R.shape[1]
    chi2Prep = np.zeros((J,nRep))
    chi2P = np.zeros((J,1))
    for j in xrange(J):
            chi2Prep[j,:] = np.array([chi2test(R[i,j,:] ) for i in xrange(nRep
                )] )
            if np.any(np.isnan(chi2Prep[j,:])):
                chi2P[j] = np.nan
            else:
                chi2P[j] = 1-ss.chi2.cdf(-2*np.sum(np.log(chi2Prep[j,:] + np.
                    finfo(float).eps)), 2*nRep) # combine p-values using Fisher
                    's Method

    nbayescall = sum(bayescall)
    if nbayescall < 1:
        nbayescall = 1

    if np.median(N) > 500: #Benjamini-Hochberg method FWER control
        chi2call = chi2P < pvalue/nbayescall
    else:
        chi2call = chi2P < pvalue

    chi2call = chi2call.flatten()
    chi2P = chi2P.flatten()

    return  chi2call, chi2P

def load_dualmodel(controlHDF5Name, caseHDF5Name):
    '''
        Load and synchonize the Case and Control Model files
    '''
    # Load the Case and Control Model files
    (controlPhi, controlTheta, controlMu, controlLoc, controlR, controlN) =
        load_model(controlHDF5Name)
    (casePhi, caseTheta, caseMu, caseLoc, caseR, caseN) = load_model(
        caseHDF5Name)

    # Extract the common locations in case and control
    caseLocIdx = [i for i in xrange(len(caseLoc)) if caseLoc[i] in controlLoc]
    controlLocIdx = [i for i in xrange(len(controlLoc)) if controlLoc[i] in
        caseLoc]

    caseMu = caseMu[caseLocIdx,:]
    controlMu = controlMu[controlLocIdx,:]
    caseR = caseR[:,caseLocIdx,:]
    controlR = controlR[:,controlLocIdx,:]

    loc = caseLoc[caseLocIdx]
```

```
    J = len(caseLoc)

    with h5py.File(controlHDF5Name, 'r') as f:
        refb = f['/refb'][...]
        f.close()
    refb = refb[controlLocIdx]

    altb = []
    acgt = {'A':0, 'C':1, 'G':2, 'T':3}
    for i in xrange(J):
        r = np.squeeze(caseR[:,i,:]) # replicates x bases

        # Make a list of the alternate bases for each replicate
        acgt_r = ['A','C','G','T']
        del acgt_r[ acgt[refb[i]] ]

        if not np.iterable(np.argmax(r, axis=-1)):
            altb_r = acgt_r[np.argmax(r, axis=-1)]
        else:
            altb_r = [acgt_r[x] for x in np.argmax(r, axis=-1)]
        altb.append(altb_r[0])

    return loc, refb, altb, controlMu, controlPhi['mu0'], controlR, controlN,\
           caseMu, casePhi['mu0'], caseR, caseN


def write_dualvcf(outputFile, loc, call, refb, altb, controlMu, controlR,
    controlN,\
                  caseMu, caseR, caseN, tau, alpha=0.95):
    '''
        Write high confidence variant calls from somatic test when there are
            both control and case sample to VCF 4.2 file.
    '''
    J = len(loc)

    today=date.today()

    chrom = [x.split(':')[0][3:] for x in loc]
    pos = [int(x.split(':')[1]) for x in loc]

    vcfF = open(outputFile,'w')

    print("##fileformat=VCFv4.1", file=vcfF)
    print("##fileDate=%0.4d%0.2d%0.2d" % (today.year, today.month, today.day),
         file=vcfF)

    print("##source=rvd2", file=vcfF)

    print('##Posterior test in cancer-normal-paired sample.', file=vcfF)

    print("##Posterior difference threshold = %0.2f" %tau, file=vcfF)
```

```
print("##Probability threshold alpha = %0.2f" %alpha, file=vcfF)

print("##Chi square test is included", file=vcfF)

uniquechrom = set(chrom)
uniquechrom = list(uniquechrom)

for i in xrange(len(uniquechrom)):
    seq = [idx for idx, name in enumerate(chrom) if name==uniquechrom[i]]
    seqlen = len(seq)
    print("##contig=<ID=%(chr)s,length=%(seqlen)d>" %{'chr': uniquechrom[i
        ],'seqlen': seqlen}, file=vcfF)


print("##INFO=<ID=COAF,Number=1,Type=Float,Description=\"Control Allele
    Frequency\">", file=vcfF)
print("##INFO=<ID=CAAF,Number=1,Type=Float,Description=\"Case Allele
    Frequency\">", file=vcfF)

print("##FORMAT=<ID=AU,Number=1,Type=Integer,Description=\"Number of 'A'
    alleles used in fitting the model\">", file=vcfF)
print("##FORMAT=<ID=CU,Number=1,Type=Integer,Description=\"Number of 'C'
    alleles used in fitting the model\">", file=vcfF)
print("##FORMAT=<ID=GU,Number=1,Type=Integer,Description=\"Number of 'G'
    alleles used in fitting the model\">", file=vcfF)
print("##FORMAT=<ID=TU,Number=1,Type=Integer,Description=\"Number of 'T'
    alleles used in fitting the model\">", file=vcfF)

print("#CHROM\tPOS\tID\tREF\tALT\tQUAL\tFILTER\tINFO\tFORMAT\tNormal\tCase
    ", file=vcfF)

for i in xrange(J):
    if call[i]:
        # restore R
        actg = ['A','C','G','T']

        idx = actg.index(refb[i])
        caseR4 = np.zeros(4)
        controlR4 = np.zeros(4)
        caseR4[idx] = np.median(caseN[:,i])-np.sum(caseR[i,:])
        controlR4[idx] = np.median(controlN[:,i])-np.sum(controlR[i,:])
        for d in xrange(idx):
            caseR4[d] = caseR[i,d]
            controlR4[d] = controlR[i,d]
        for d in xrange(3-idx):
            caseR4[d+idx+1] = caseR[i,d+idx]
            controlR4[d+idx+1] = controlR[i,d+idx]

        print ("chr%s\t%d\t.\t%s\t%s\t.\tPASS\tCOAF=%0.3f;CAAF=%0.3f\tAU:
            CU:GU:TU\t%d:%d:%d:%d\t%d:%d:%d:%d" \
               % (chrom[i], pos[i], refb[i], altb[i],controlMu[i]*100.0,
                   caseMu[i]*100.0,\
```

```
                    controlR4[0], controlR4[1], controlR4[2], controlR4[3],\
                    caseR4[0], caseR4[1], caseR4[2], caseR4[3]), file=vcfF)

    vcfF.close()

def write_vcf(outputFile, loc, call, refb, altb, caseMu, controlMu):
    """ Write high confidence variant calls to VCF 4.2 file.   //for
        compatible to previous programms before test functions adapted for
        somatic test.
    """

    #TODO: get dbSNP id for chrom:pos
    J = len(loc)

    today=date.today()

    chrom = [x.split(':')[0][3:] for x in loc]
    pos = [int(x.split(':')[1]) for x in loc]
    vcfF = open(outputFile,'w')

    print("##fileformat=VCFv4.1", file=vcfF)
    print("##fileDate=%0.4d%0.2d%0.2d" % (today.year, today.month, today.day),
         file=vcfF)

    print("##INFO=<ID=COAF,Number=1,Type=Float,Description=\"Control Allele
        Frequency\">", file=vcfF)
    print("##INFO=<ID=CAAF,Number=1,Type=Float,Description=\"Case Allele
        Frequency\">", file=vcfF)

    print("#CHROM\tPOS\tID\tREF\tALT\tQUAL\tFILTER\tINFO", file=vcfF)
    for i in xrange(J):
        if call[i]:
            print("%s\t%d\t.\t%c\t%s\t.\tPASS\tCOAF=%0.3f;CAAF=%0.3f" % (chrom
                [i], pos[i], refb[i], altb[i], controlMu[i]*100.0, caseMu[i
                ]*100.0), file=vcfF)

    vcfF.close()


def sample_run():
    n = 1000
    J = 10
    phi = {'mu0': 0.20, 'M0': 2e3, 'a': 1e6, 'b': 1}
    r, theta, mu, M = generate_sample(phi, n=n, J=J, seedint=10)
    r[:, int(J / 2)] = n * np.array([0.50, 0.55, 0.45])

    phi, theta_s, mu_s = mh_sample(r, n,
                                        nsample=100,
                                        thin=0,
                                        burnin=0)
```

```
def load_model(h5Filename):
    """ Returns the RVD2.7 model samples and parameters.
    Takes an hdf5 filename and returns phi and other parameters
    """

    out = []

    with h5py.File(h5Filename, 'r') as h5file:
        # Load phi - it always exists
        phi = {'mu0': h5file['phi/mu0'][()],
               'M0': h5file['phi/M0'][()],
               'M': h5file['phi/M'][...]}
        out.append(phi)

        # Load theta if it exists
        if u"theta" in h5file.keys():
            theta = h5file['theta'][...]
            out.append(theta)

        # Load mu if it exists
        if u"mu" in h5file.keys():
            mu = h5file['mu'][...]
            out.append(mu)

        # Load loc if it exists
        if u"loc" in h5file.keys():
            loc = h5file['loc'][...]
            out.append(loc)

        # Load r if it exists
        if u"r" in h5file.keys():
            r = h5file['r'][...]
            out.append(r)

        if u"n" in h5file.keys():
            n = h5file['n'][...]
            out.append(n)


    return tuple(out)



def save_model(h5Filename, phi, mu=None, theta=None, r=None, n=None, loc=None,
     refb=None):
    """ Save the RVD2.7 model samples and parameters """

    # TODO add attributes to hdf5 file
    h5file = h5py.File(h5Filename, 'w')

    # Save the model parameters (phi)
    h5file.create_group('phi')
```

```
    h5file['phi'].create_dataset('mu0', data=phi['mu0'])
    h5file['phi'].create_dataset('M0', data=phi['M0'])
    h5file['phi'].create_dataset('M', data=phi['M'],
                                     chunks=True,
                                     fletcher32=True,
                                     compression='gzip')

    # Save the latent variables if available.
    if mu is not None:
        h5file.create_dataset('mu', data=mu,
                              chunks=True, fletcher32=True, compression='gzip'
                                  )
    if theta is not None:
        h5file.create_dataset('theta', data=theta,
                              chunks=True, fletcher32=True, compression='gzip'
                                  )

    # Save the data used for fitting the model if available
    if r is not None:
        h5file.create_dataset('r', data=r,
                              chunks=True, fletcher32=True, compression='gzip'
                                  )
    if n is not None:
        h5file.create_dataset('n', data=n,
                              chunks=True, fletcher32=True, compression='gzip'
                                  )

    # Save the reference data
    if loc is not None:
        h5file.create_dataset('loc', data=loc,
                              chunks=True, fletcher32=True, compression='gzip'
                                  )
    if refb is not None:
        h5file.create_dataset('refb', data=refb)

    h5file.close()


def generate_sample(phi, n=100, N=3, J=100, seedint=None):
    """Returns a sample with n reads, N replicates, and
    J locations. The parameters of the model are in the structure phi.
    """

    if seedint is not None:
        np.random.seed(seedint)

    # Draw J location-specific error rates from a Beta
    alpha0 = phi['M0']*phi['mu0']
    beta0 = phi['M0']*(1-phi['mu0'])
    mu = ss.beta.rvs(alpha0, beta0, size=J)

    # Draw sample error rate and error count
```

```
    theta=np.zeros((N,J))
    r = np.zeros((N,J))
    for j in xrange(0, J):
        alpha = mu[j]*phi['M'][j]
        beta = (1-mu[j])*phi['M'][j]
        theta[:,j] = ss.beta.rvs(alpha, beta, size=N)
        r[:,j] = ss.binom.rvs(n, theta[:,j])
    return r, theta, mu

def complete_ll(phi, r, n, theta, mu):
    """ Return the complete data log-likelihood.
    """
    alpha0 = phi['M0']*phi['mu0'] + np.finfo(np.float).eps
    beta0 = phi['M0']*(1-phi['mu0']) + np.finfo(np.float).eps

    alpha = phi['M']*mu + np.finfo(np.float).eps
    beta = phi['M']*(1 - mu) + np.finfo(np.float).eps

    # Bound theta away from 0 or 1
    theta[theta < np.finfo(np.float).eps] = np.finfo(np.float).eps
    theta[theta > 1-np.finfo(np.float).eps] = 1 - np.finfo(np.float).eps

    logPmu = beta_log_pdf(mu, alpha0, beta0)
    logPtheta = beta_log_pdf(theta, alpha, beta)
    logPr = ss.binom.logpmf(r, n, theta)

    return np.sum(logPmu + logPtheta + logPr)

def estimate_mom(r, n):
    """ Return model parameter estimates using method-of-moments.
    """

    theta = r/(n + np.finfo(np.float).eps) # make sure this is non-truncating
        division
    if np.ndim(r) == 1: mu = theta
    elif np.ndim(r) > 1: mu = np.mean(theta, 0)

    mu0 = np.mean(mu)
    M0 = (mu0*(1-mu0))/(np.var(mu) + np.finfo(np.float).eps)

    # estimate M. If there is only one replicate, set M as 10 times of M0.
    # If there is multiple replicates, set M according to the moments of beta
        distribution
    if np.shape(theta)[0] is 1:
        M = 10*M0*np.ones_like(mu)
    else:
        M = (mu*(1-mu))/(np.var(theta, 0) + np.finfo(np.float).eps )

    phi = {'mu0':mu0, 'M0':M0, 'M':M}
    return phi, mu, theta

def sampleLocMuMH(args):
```

```
    # Sample from the proposal distribution at a particular location
    mu, Qsd, theta, M, alpha0, beta0 = args

    # TODO put an escape in here to avoid an infinite loop
    while True:
        mu_p = ss.norm.rvs(mu, Qsd)
        if (0 < mu_p < 1): break

    # Log-likelihood for the proposal mu
    alpha_p = mu_p*M + np.finfo(np.float).eps
    beta_p = (1-mu_p)*M + np.finfo(np.float).eps
    logPmu_p = beta_log_pdf(mu_p, alpha0, beta0) \
                + np.sum(beta_log_pdf(theta, alpha_p, beta_p))

    # Log-likelihood for the current mu
    alpha = mu*M + np.finfo(np.float).eps
    beta = (1-mu)*M + np.finfo(np.float).eps
    logPmu = beta_log_pdf(mu, alpha0, beta0) \
                + np.sum(beta_log_pdf(theta, alpha, beta))

    # Accept new mu if it increases posterior pdf or by probability
    loga = logPmu_p - logPmu
    if (loga > 0 or np.log(np.random.random()) < loga):
        mu = mu_p
    return mu

def sampleMuMH(theta, mu0, M0, M, mu=ss.beta.rvs(1, 1), Qsd=0.1, burnin=0,
    mh_nsample=1, thin=0, pool=None):
    """ Return a sample of mu with parameters mu0 and M0.
    """
    if np.ndim(theta) == 1: (N, J) = (1, np.shape(theta)[0])
    elif np.ndim(theta) > 1: (N, J) = np.shape(theta)

    alpha0 = mu0*M0 + np.finfo(np.float).eps
    beta0 = (1-mu0)*M0 + np.finfo(np.float).eps

    mu_s = np.zeros( (mh_nsample, J) )
    for ns in xrange(0, mh_nsample):
        if pool is not None:
            args = zip(mu, Qsd, theta.T, M, repeat(alpha0, J), repeat(beta0, J
                ))
            mu = pool.map(sampleLocMuMH, args)
        else:
            for j in xrange(0, J):
                args = (mu[j], Qsd[j], theta[:,j], M[j], alpha0, beta0)
                mu[j] = sampleLocMuMH(args)
        # Save the new sample
        mu_s[ns, :] = np.copy(mu)

    if burnin > 0.0:
        mu_s = np.delete(mu_s, np.s_[0:np.int(burnin*mh_nsample):], 0)
    if thin > 0:
```

```
        mu_s = np.delete(mu_s, np.s_[::thin], 0)

    return mu_s

def mh_sample(r, n, gibbs_nsample=10000,mh_nsample=10, burnin=0.2, thin=2,
    pool=None):
    """ Return MAP parameter and latent variable estimates obtained by

    Metropolis-Hastings sampling.
    By default, sample 10000 M-H with a 20% burn-in and thinning factor of 2.
    Stop when the change in complete data log-likelihood is less than 0.01%.
    """
    if np.ndim(r) == 1: N, J = (1, np.shape(r)[0])
    elif np.ndim(r) == 2: N, J = np.shape(r)
    elif np.ndim(r) == 3:
        r = np.sum(r, 2) # sum over non-reference bases
        N, J = np.shape(r)

    # Initialize a hdf5 file for logging model progress
    h5Filename = tempfile.NamedTemporaryFile(suffix='.hdf5').name
    logging.debug("Storing temp data in %s" % h5Filename)
    h5file = h5py.File(h5Filename, 'w')
    h5file.create_group('phi')
    h5file['phi'].create_dataset('mu0', (1,), dtype='f')
    h5file['phi'].create_dataset('M0', (1,), dtype='f')
    h5file['phi'].create_dataset('M', (J,), dtype='f')
    h5file.create_dataset('theta_s', (N, J, gibbs_nsample), dtype='f')
    h5file.create_dataset('mu_s', (J, gibbs_nsample), dtype='f')
    # Initialize estimates using MoM
    phi, mu, theta = estimate_mom(r, n)
    logging.debug("MoM: mu0 = %0.3e; M0 = %0.3e." % (phi['mu0'], phi['M0']) )

    # Correct MoM estimates to be non-trivial
    mu[mu < np.finfo(np.float).eps*1e4] = phi['mu0']
    theta[theta < np.finfo(np.float).eps*1e4] = phi['mu0']
    phi['M'][phi['M'] < np.finfo(np.float).eps *1e4] = 1

    # Set Qsd for proposal distribution according to mom of mu

    def boundfn(mu):
        bound = 0.001
        if bound < mu < 1-bound:
            return mu*(1-mu)/10
        else:
            return bound/10


##    def boundfn(mu):
##        bound = 0.001
##        if bound < mu < 1-bound:
##            return mu/10
##        else:
```

```
##            return bound/10

    Qsd = map(boundfn, mu)

    # Sample theta, mu by gibbs sampling
    theta_s = np.zeros( (N, J, gibbs_nsample) )
    mu_s = np.zeros( (J, gibbs_nsample) )
    for i in xrange(gibbs_nsample):
        if i % 100 == 0 and i > 0: logging.debug("Gibbs Iteration %d" % i)

        # Draw samples from p(theta | r, mu, M) by Gibbs
        alpha = r + mu*phi['M'] + np.finfo(np.float).eps
        beta = (n - r) + (1-mu)*phi['M'] + np.finfo(np.float).eps
        theta = ss.beta.rvs(alpha, beta)

        # Draw samples from p(mu | theta, mu0, M0) by Metropolis-Hastings
        mu_mh = sampleMuMH(theta, phi['mu0'], phi['M0'], phi['M'], mu=mu, Qsd=
            Qsd, mh_nsample=mh_nsample, pool=pool)
        mu = np.median(mu_mh, axis=0)
        # Store the sample
        theta_s[:,:,i] = np.copy(theta)
        mu_s[:,i] = np.copy(mu)
        # Update parameter estimates
        # phi['mu0'] = np.mean(mu)
        # phi['M0'] = (phi['mu0']*(1-phi['mu0']))/(np.var(mu) + np.finfo(np.
            float).eps)
        # TODO update for M

        # Store the current model
        h5file['phi']['mu0'][0] = phi['mu0']
        h5file['phi']['M0'][0] = phi['M0']
        h5file['phi']['M'][...] = phi['M']
        h5file['theta_s'][:,:,i] = theta
        h5file['mu_s'][:,i] = mu
        h5file.flush()
    # Apply the burn-in and thinning
    if burnin > 0.0:
        mu_s = np.delete(mu_s, np.s_[0:np.int(burnin*gibbs_nsample):], 1)
        theta_s = np.delete(theta_s, np.s_[0:np.int(burnin*gibbs_nsample):],
            2)
    if thin > 1:
        mu_s = np.delete(mu_s, np.s_[::thin], 1)
        theta_s = np.delete(theta_s, np.s_[::thin], 2)

    h5file.close()
    return (phi, theta_s, mu_s)

def beta_log_pdf(x, a, b):
    return gammaln(a+b) - gammaln(a) - gammaln(b) \
            + (a-1)*np.log(x+np.finfo(np.float).eps) \
            + (b-1)*np.log(1-x+np.finfo(np.float).eps)
```

```
def ll(phi, r):
    """ Return the log-likelihood of the data, r, under the model, phi.
    """
    pass

def make_pileup(bamFileName, fastaFileName, region):
    """ Creates a pileup file using samtools mpileup in /pileup directory.
    """

    # Check that BAM file exists
    assert os.path.isfile(bamFileName), "BAM file does not exist: %s" %
        bamFileName

    # Check that FASTA reference file exists
    assert os.path.isfile(fastaFileName), "FASTA file does not exist: %s" %
        fastaFileName

    # Create pileup directory if it doesn't exist
    if not os.path.isdir("pileup"):
        os.makedirs("pileup")

    # Format the samtools call
    callString = ["samtools", "mpileup", "-d", "1000000", "-r", "%s" % region,
                  "-f", "%s" % fastaFileName, "%s" % bamFileName]

    # Remove the extension from the bam filename and replace with .pileup
    pileupFileName = bamFileName.split("/")[-1]
    pileupFileName = os.path.join("pileup",
                                  "%s.pileup" % pileupFileName.split(".", 1)
                                      [0])

    # Run samtools pileup only if the file doesn't already exist.
    #try:
    #    with open(pileupFileName, 'r'):
    #        logging.debug("Pileup file exists: %s" % pileupFileName)
    #except IOError:
    logging.debug("[call] %s", " ".join(callString))
    with open(pileupFileName, 'w') as fout:
        subprocess.call(callString, stdout=fout)
    return pileupFileName

def make_depth(pileupFileName):
    """ Generates a depth chart file for each pileup file and stores it in the
        /depth_chart directory. The folder will be created if it doesn't exist
            .
    """

    if not os.path.isdir("depth_chart"):
        os.makedirs("depth_chart")

    # TODO replace this with a python version.
```

```
    callString = ["../../bin/pileup2dc", "%s" % pileupFileName]

    dcFileName = pileupFileName.split("/")[-1]
    dcFileName = os.path.join("depth_chart",
                                "%s.dc" % dcFileName.split(".", 1)[0])
    #try:
    #    with open(dcFileName, 'r'):
#  logging.debug("Depth chart file exists: %s" % dcFileName)
    #except IOError:
    logging.debug("Converting %s to depth chart." % pileupFileName)
    with open(dcFileName, 'w') as fout:
        subprocess.call(callString, stdout=fout)
    return dcFileName

def load_depth(dcFileNameList):
    """ Return (r, n, location, reference base) for a list of depth charts.
        The
        variable r is the error read depth and n is the total read depth.
    """
    r=[]; n=[]
    acgt = {'A':0, 'C':1, 'G':2, 'T':3}

    loc = []
    refb = {}
    cd = []
    for dcFileName in dcFileNameList:
        with open(dcFileName, 'r') as dcFile:
            header = dcFile.readline().strip()
            dc = dcFile.readlines()
            dc = [x.strip().split("\t") for x in dc]
            loc1 = [x[1]+':'+str(x[2]).strip('\000') for x in dc if x[4] in
                acgt.keys()]

     loc.append( loc1 )

            refb1 = dict(zip(loc1, [x[4] for x in dc if x[4] in acgt.keys()]))
            refb.update(refb1)
            cd.append( dict(zip(loc1, [map(int, x[5:9]) for x in dc if x[4] in
                 acgt.keys()])) )

    loc = list(reduce(set.intersection, map(set, loc)))

    def stringSplitByNumbers(x):
        r = re.compile('(\d+)')
        l = r.split(x)
        return [int(y) if y.isdigit() else y for y in l]

    loc = sorted(loc,key = stringSplitByNumbers)
    logging.debug(loc)
    refb = [refb[k] for k in loc]

    J = len(loc)
```

```
    N = len(dcFileNameList)
    for i in xrange(0, N):
        logging.debug("Processing %s" % dcFileNameList[i])
        c = np.array( [cd[i][k] for k in loc] )
        n1 = np.sum(c, 1)
        #r1 = np.zeros(J)
        refIdx=np.zeros(J)

        for j in xrange(0,J):
            #r1[j] = n1[j] - c[j, acgt[refb[j]]]
            refIdx[j] = 4*j+acgt[refb[j]]
        c = np.delete(c, refIdx, None)
        c = np.reshape(c, (J, 3) )
        #r.append(r1)
        n.append(n1)
        r.append(c)
    r = np.array(r)
    n = np.array(n)

    return (r, n, loc, refb)

def chi2test(X, lamda=2.0/3, pvector=np.array([1.0/3]*3)):
    """ Do chi2 test to decide how well the error reads fits uniform
        multinomial distribution. P-value returned.
        lamda=1 Pearson's chi-square
        lamda=0 the log likelihood ratio statistic/ G^2
        lamda=-1/2 Freeman-Tukey's F^2
        lamda=-1  Neyman modified chi-square
        lamda=-2  modified G^2
    """
    X=np.array(X)

    nsum=np.sum(X)
    if nsum == 0: return np.nan # return NaN if there are no counts
    E=nsum*pvector


    if lamda==0 or lamda==-1:
        C=2.0*np.sum(X*np.log(X*1.0/E))
    else:
        C=2.0/(lamda*(lamda+1))*np.sum(X*((X*1.0/E)**lamda-1))

    df=len(pvector)-1
    #p=scipy.special.gammainc(C,df)
    # p=1-gammainc(df/2,C/2)
    p = 1 - ss.chi2.cdf(C, df)
    return(p)

def sample_post_diff(muCaseG, muControlG, N):
    """ Return N samples from the posterior distribution for
         u_j|r_case - u_j|r_control. """
```

```
    nCase = muCaseG.shape[1]
    nControl = muControlG.shape[1]

    caseSample = np.random.choice(nCase, size=N, replace=True)
    controlSample = np.random.choice(nControl, size=N, replace=True)

    muCaseS = muCaseG[:, caseSample]
    muControlS = muControlG[:, controlSample]

    Z = muCaseS - muControlS

    return (Z, muCaseS, muControlS)

def bayes_test(Z, roi, type = 'close'):
    """ Return posterior probabilities in regions defined in list of tuples (
        roi)
        from samples in columns of Z. """
    (J,N)=np.shape(Z)

    nTest = len(roi) # get the number of regions to compute probabilities

    p = np.zeros((J,nTest))
    for i in xrange(nTest):
        for j in xrange(J):
                if type == 'close':
                    # somatic test
                    p[j,i] = np.float( np.sum( np.logical_and( (Z[j,:] >= roi[
                        i][0]), (Z[j,:] <= roi[i][1]) ) ) ) / N
                elif type == 'open':
                    # diff test
                    p[j,i] = np.float( np.sum( np.logical_and( (Z[j,:] > roi[i
                        ][0]), (Z[j,:] < roi[i][1]) ) ) ) / N

    p = np.sum(p,1) # add up all the regions in each position.

    return p # combine probabilities from all regions.

if __name__ == '__main__':
    main()
```

Bibliography

[1] Alkan, C., Kidd, J. M., Marques-Bonet, T., Aksay, G., Antonacci, F., Hormozdiari, F., Kitzman, J. O., Baker, C., Malig, M., Mutlu, O., Sahinalp, S. C., Gibbs, R. A., and Eichler, E. E. (2009). Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature Genetics*, 41(10):1061–1067.

[2] Allen, E. (2013). Molecular characterization of tumors using next-generation sequencing. Technical Report 770-2013-011, 2013 Illumina, Inc.

[3] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

[4] Blei, D. (2011). Variational inference.

[5] Blei, D. M. and Jordan, M. I. (2004). Variational methods for the dirichlet process. In *Proceedings of the twenty-first international conference on Machine learning*, page 12. ACM.

[6] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[7] Capobianchi, M. R., Giombini, E., and Rozera, G. (2012). Next-generation sequencing technology in clinical virology. *Clinical Microbiology and Infection*, 19(1):15–22.

[8] Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

[9] Cibulskis, K., Lawrence, M. S., and Carter, S. L. (2013). Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature*.

[10] Consortium, . G. P. et al. (2012). An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65.

[11] Consortium, T. H. M. P. (2013). A framework for human microbiome research. *Nature*, 486(7402):215–221.

[12] Cressie, N. and Read, T. R. (1984). Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 440–464.

[13] Cushing, A., Flaherty, P., Hopmans, E., Bell, J. M., and Ji, H. P. (2013). Rvd: a command-line program for ultrasensitive rare single nucleotide variant detection using targeted next-generation dna resequencing. *BMC research notes*, 6(1):206.

[14] DePristo, M. A., Banks, E., Poplin, R., Garimella, K. V., Maguire, J. R., Hartl, C., Philippakis, A. A., del Angel, G., Rivas, M. A., Hanna, M., et al. (2011). A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491–498.

[15] Devroye, L. (1986). Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, pages 260–265. ACM.

[16] Efron, B. (2010). *Large-scale inference: empirical Bayes methods for estimation, testing, and prediction*, volume 1. Cambridge University Press.

[17] Fan, H. C., Blumenfeld, Y. J., Chitkara, U., Hudgins, L., and Quake, S. R. (2008). Noninvasive diagnosis of fetal aneuploidy by shotgun sequencing DNA from maternal blood. *PNAS*, 105(42):16266–16271.

[18] Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE.

[19] Fisher, S. R. A., Genetiker, S., Fisher, R. A., Genetician, S., Britain, G., Fisher, R. A., and Généticien, S. (1970). *Statistical methods for research workers*, volume 14. Oliver and Boyd Edinburgh.

[20] Flaherty, P., Natsoulis, G., Muralidharan, O., Winters, M., Buenrostro, J., Bell, J., Brown, S., Holodniy, M., Zhang, N., and Ji, H. P. (2011). Ultrasensitive detection of rare mutations using next-generation targeted resequencing. *Nucleic Acids Research*.

[21] Gamerman, D. and Lopes, H. F. (2006). *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press.

[22] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. CRC press.

[23] Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical Science*, pages 473–483.

[24] Ghedin, E., Laplante, J., DePasse, J., Wentworth, D. E., Santos, R. P., Lepow, M. L., Porter, J., Stellrecht, K., Lin, X., Operario, D., Griesemer, S., Fitch, A., Halpin, R. A., Stockwell, T. B., Spiro, D. J., Holmes, E. C., and George, K. S. (2010). Deep Sequencing Reveals Mixed Infection with 2009 Pandemic Influenza A (H1N1) Virus Strains and the Emergence of Oseltamivir Resistance. *Journal of Infectious Diseases*, 203(2):168–174.

[25] Gilks, W. R. (2005). *Markov chain monte carlo*. Wiley Online Library.

[26] Gilks, W. R., Best, N., and Tan, K. (1995). Adaptive rejection metropolis sampling within gibbs sampling. *Applied Statistics*, pages 455–472.

[27] Glenn, T. C. (2011). Field guide to next-generation dna sequencers. *Molecular Ecology Resources*, 11(5):759–769.

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

[29] Howarth, K., Blood, K., Ng, B., Beavis, J., Chua, Y., Cooke, S., Raby, S., Ichimura, K., Collins, V., Carter, N., et al. (2007). Array painting reveals a high frequency of balanced translocations in breast cancer cell lines that break in cancer-relevant genes. *Oncogene*, 27(23):3345–3359.

[30] Howarth, K. D., Pole, J. C., Beavis, J. C., Batty, E. M., Newman, S., Bignell, G. R., and Edwards, P. A. (2011). Large duplications at reciprocal translocation breakpoints that might be the counterpart of large deletions and could arise from stalled replication bubbles. *Genome Research*, 21(4):525–534.

[31] Huang, R., Pavlovic, V., and Metaxas, D. N. (2004). A graphical model framework for coupling mrfs and deformable models. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–739. IEEE.

[32] Jensen, F. V. (1996). *An introduction to Bayesian networks*, volume 210. UCL press London.

[33] Jordan, M. I. (2004). Graphical models. *Statistical Science*, pages 140–155.

[34] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

[35] Kitzman, J. O., Snyder, M. W., Ventura, M., Lewis, A. P., Qiu, R., Simmons, L. E., Gammill, H. S., Rubens, C. E., Santillan, D. A., Murray, J. C., Tabor, H. K., Bamshad, M. J., Eichler, E. E., and Shendure, J. (2012). Noninvasive Whole-Genome Sequencing of a Human Fetus. *Science Translational Medicine*, 4(137):137ra76–137ra76.

[36] Koboldt, D. C., Steinberg, K. M., Larson, D. E., Wilson, R. K., and Mardis, E. R. (2013). The next-generation sequencing revolution and its impact on genomics. *Cell*, 155(1):27–38.

[37] Koboldt, D. C., Zhang, Q., Larson, D. E., Shen, D., McLellan, M. D., Lin, L., Miller, C. A., Mardis, E. R., Ding, L., and Wilson, R. K. K. (2012). VarScan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome Research*, 22(3):568–576.

[38] Laird, P. W. (2010). Principles and challenges of genomewide DNA methylation analysis. *Nature Reviews Genetics*, 11(3):191–203.

[39] Li, H. (2011). A statistical framework for snp calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993.

[40] Li, S. Z. (1995). *Markov random field modeling in computer vision*. Springer-Verlag New York, Inc.

[41] Li, S. Z. and Singh, S. (2009). *Markov random field modeling in image analysis*, volume 3. Springer.

[42] Manousiouthakis, V. I. and Deem, M. W. (1999). Strict detailed balance is unnecessary in monte carlo simulation. *The Journal of chemical physics*, 110(6):2753–2756.

[43] Mardis, E. R. (2008). The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141.

[44] Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., Berka, J., Braverman, M. S., Chen, Y.-J., Chen, Z., et al. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380.

[45] Matthews, D., Hosker, J., Rudenski, A., Naylor, B., Treacher, D., and Turner, R. (1985). Homeostasis model assessment: insulin resistance and β-cell function from fasting plasma glucose and insulin concentrations in man. *Diabetologia*, 28(7):412–419.

[46] McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., and DePristo, M. A. (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20(9):1297–1303.

[47] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (2004). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

[48] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

[49] Navin, N., Krasnitz, A., Rodgers, L., Cook, K., Meth, J., Kendall, J., Riggs, M., Eberling, Y., Troge, J., Grubor, V., Levy, D., Lundin, P., Maner, S., Zetterberg, A., Hicks, J., and Wigler, M. (2010). Inferring tumor progression from genomic heterogeneity. *Genome Research*, 20(1):68–80.

[50] Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

[51] Newman, S., Howarth, K. D., Greenman, C. D., Bignell, G. R., Tavaré, S., and Edwards, P. A. (2013). The relative timing of mutations in a breast cancer genome. *PloS one*, 8(6):e64991.

[52] Ouyang, Z., Zhou, Q., and Wong, W. H. (2009). ChIP-Seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *PNAS*, 106(51):21521–21526.

[53] Quackenbush, J. (2002). Microarray data normalization and transformation. *Nature genetics*, 32:496–501.

[54] Quail, M. A., Smith, M., Coupland, P., Otto, T. D., Harris, S. R., Connor, T. R., Bertoni, A., Swerdlow, H. P., and Gu, Y. (2012). A tale of three next generation sequencingplatforms: comparison of Ion Torrent, PacificBiosciences and Illumina MiSeq sequencers. *BMC Genomics*, 13(1):1–1.

[55] Raczy, C., Petrovski, R., Saunders, C. T., Chorny, I., Kruglyak, S., Margulies, E. H., Chuang, H.-Y., Källberg, M., Kumar, S. A., Liao, A., et al. (2013). Isaac: Ultra-fast whole genome secondary analysis on illumina sequencing platforms. *Bioinformatics*.

[56] Raftery, A. E. and Lewis, S. M. (1996). Implementing mcmc. In *Markov chain Monte Carlo in practice*, pages 115–130. Springer.

[57] Rivera, C. M. and Ren, B. (2013). Mapping Human Epigenomes. *Cell*, 155(1):39–55.

[58] Robasky, K., Lewis, N. E., and Church, G. M. (2013). The role of replicates for error mitigation in next-generation sequencing. *Nature Reviews Genetics*.

[59] Robert, C. P. and Casella, G. (2004). *Monte Carlo statistical methods*, volume 319. Citeseer.

[60] Ronaghi, M., Karamohamed, S., Pettersson, B., Uhlén, M., and Nyrén, P. (1996). Real-time dna sequencing using detection of pyrophosphate release. *Analytical biochemistry*, 242(1):84–89.

[61] Saddiki, H., McAuliffe, J., and Flaherty, P. (2013). Glad: A mixed-membership model for heterogeneous tumor subtype classification. *Submitted*.

[62] Saunders, C. T., Wong, W. S. W., Swamy, S., Becq, J., Murray, L. J., and Cheetham, R. K. (2012). Strelka: accurate somatic small-variant calling from sequenced tumor-normal sample pairs. *Bioinformatics*, 28(14):1811–1817.

[63] Schuster, S. C. (2007). Next-generation sequencing transforms todays biology. *Nature*, 200(8).

[64] Shendure, J., Porreca, G. J., Reppas, N. B., Lin, X., McCutcheon, J. P., Rosenbaum, A. M., Wang, M. D., Zhang, K., Mitra, R. D., and Church, G. M. (2005). Accurate multiplex polony sequencing of an evolved bacterial genome. *Science*, 309(5741):1728–1732.

[65] Shenoy, P. P. (1992). Valuation-based systems for bayesian decision analysis. *Operations research*, 40(3):463–484.

[66] Spencer, D. H., Tyagi, M., Vallania, F., Bredemeyer, A., Pfeifer, J. D., Mitra, R. D., and Duncavage, E. J. (2013). Performance of common analysis methods for detecting low-frequency single nucleotide variants in targeted next-generation sequence data. *The Journal of Molecular Diagnostics*.

[67] Stan Development Team (2014). Stan: A c++ library for probability and sampling, version 2.2.

[68] Stead, L. F., Sutton, K. M., Taylor, G. R., Quirke, P., and Rabbitts, P. (2013). Accurately Identifying Low-Allelic Fraction Variants in Single Samples with Next-Generation Sequencing: Applications in Tumor Subclone Resolution. *Human Mutation*, 34(10):1432–1438.

[69] Titterington, D. (2004). Bayesian methods for neural networks and related models. *Statistical Science*, pages 128–139.

[70] Wainwright, M. and Jordan, M. (2005). A variational principle for graphical models. *New Directions in Statistical Signal Processing*, page 155.

[71] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

[72] Wang, C. and Blei, D. (2013). Variational Inference in Nonconjugate Models. *The Journal of Machine Learning Research*.

[73] Yau, C., Mouradov, D., Jorissen, R. N., Colella, S., Mirza, G., Steers, G., Harris, A., Ragoussis, J., Sieber, O., Holmes, C. C., et al. (2010). A statistical approach for detecting genomic aberrations in heterogeneous tumor samples from single nucleotide polymorphism genotyping data. *Genome Biol*, 11(9):R92–R92.

[74] Zhang, Y., Brady, M., and Smith, S. (2001). Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *Medical Imaging, IEEE Transactions on*, 20(1):45–57.