School of Medicine

Oregon Health & Science University

Certificate of Approval

This is to certify that the Master's Thesis of

Benjamin A. Cordier

"Evaluation of Background Prediction for Variant Detection in a Clinical Context: Towards Improved NGS Monitoring of Minimal Residual Disease in Hematological Malignancies"

Has been approved

_____
Thesis Advisor

_____
Committee Member

_____
Committee Member

_____
Committee Member

_____
Committee Member

_____
Committee Member

I would like to give my thanks and gratitude to Dr. Shannon McWeeney (Thesis Mentor, Committee Chair) for her invaluable guidance, gentle nudging, and patience – this project would in no way have been possible without her; Dr. Christina Zheng (Committee Member) for lending her depth of knowledge on genomics data processing, managing the GATK pipeline, and aid in the experimental design; Dr. Beth Wilmot (Committee Member) for helping guide the problem formulation with probing questions; Dr. Richard Press (Committee Member) for his insight into the clinical context, practical considerations, and providing much of the original inspiration for this research; and Dr. Guanming Wu (Committee Member) for his early support and guidance throughout the project. I would also like to acknowledge my family – their chorus of support provided much needed motivation during some of the most challenging moments, and Molly, whose positivity helped reframe many seemingly insurmountable mountains into piddly hillocks.

Table of Contents

1. Abstract

With the growing value of next generation sequencing (NGS) assays for the determination of minimal residual disease (MRD) in the clinic, the confident and sensitive detection of low frequency variants is crucial to the treatment of cancer. Current *in silico* pipelines often lack the sensitivity to detect low frequency variants, whose variant allele frequencies (VAFs) covary with sample purity (i.e. tumor-normal and/or normal-tumor contamination), sample clonality, and copy number variations. Sensitivity is also confounded by the background error inherent to sequencing data, which may be introduced by systematic platform error, library amplification, and errors in sample preparation. Attempting to mitigate background error in sequencing data, researchers have developed many software error correction programs that model sources of error to mitigate its impact on downstream processing. While these models have been developed for *de novo* assembly, metagenomics research, and viral haplotype reconstruction, their application to the use case of low frequency variant detection has yet to be explored in-depth. For this research, we sought to develop a software framework for the evaluation of background error models in the low frequency variant use case, with a specific focus on their potential value to MRD monitoring.

2. Background

2.1 Next Generation Sequencing for Monitoring of Minimal Residual Disease

The sensitive detection of minimal residual disease (MRD), defined as the post-therapy disease burden, by next generation sequencing (NGS) represents a valuable prognostic tool when aiming to treat hematological malignancies with precision medicine strategies. Current research has found that high sensitivity measurements of MRD are predictive of clinical outcome [1], may be used to guide additional therapeutic interventions [1-2], allow for more granular risk stratification [2], improve the administration of precision therapies [2], and are highly valuable for prognostic assessments [3]. A notable example regarding the treatment of acute myeloid leukemia (AML) crystallizes the importance of MRD, where it has been asserted that, looking to the future, regular quantification of AML with NGS-based assays will be as important in determining a precision medicine treatment strategy as the determination of the disease subtype [4]. While this assertion may be especially true for AML, there is little reason to believe that the knowledge gained from repeated MRD assays would not benefit the standard of care for most, if not all, hematological malignancies. Owing to the early successes and potential future, interest has grown significantly around the use of NGS for the sensitive measurement of MRD – a problem domain that is characterized by somatic low frequency variation.

2.2 Contemporary Challenges to Somatic Low Frequency Variant Detection

The growing importance of NGS-based MRD assays emphasizes an existing need for accurate, sensitive methods for the detection of somatic low frequency variants. Currently, their detection is largely made challenging by three confounding factors. The first two arise from the cancer itself – the presence of multiple subclones in a sample (clonality) and the presence of copy number variations (CNVs), which includes both local CNVs and aneuploidy. The third confounding factor is the reduction of sample purity due to normal cell contamination [6].[1] The impact of the former two dynamics, in comparison to the latter, differs in the challenges they present. While reduced sample purity from contamination of normal tissue in a tumor sample is expected to result in a global reduction in variant allele frequencies (VAFs) [7-10], the clonal structure of a cancer and resident copy number variations may conspire to yield highly variable, low VAFs [11]. Combined, these three covariates of VAF form a complex problem space for variant detection and, in particular, low frequency variants; defined as variants with

[1] A similar, additional concern when analyzing tumor data with a paired normal is contamination of the tumor in the normal sample, however, this has been deemed outside the scope of the current iteration of this research.

an allele frequency of less than of equal to 5% VAF. However, this enumeration merely represents the primary biological components of the somatic low-frequency variant detection problem.

Other technical artifacts from sample processing and the selected NGS platform may also confound analyses. Library amplification with PCR, required by some NGS sequencing platforms (e.g. Illumina), can induce a reduction in the coverage of genomic regions enriched with G and C bases, known as GC content bias [12-14]. Systematic base-call errors, which include substitutions, insertion-deletions (indels), and homopolymer errors are also common across NGS sequencing data, with error rates varying not only by error type, but also by organism and platform [14-15]. Fortunately, the former has been found to be unimodal and amenable to comparatively simple modeling [13]. Less fortunately, the complexity of platform base call errors and their provenance enables no easy solution, even with the integration of large amounts of prior knowledge (e.g. platform, organism, error specific profiles, and/or other assumptions) [15]. This complexity, however, has not precluded the development of software to account for these errors within prescribed contexts.

2.3 Software Approaches to Modeling Background Error

Indeed, much software has been developed for the modeling of platform error and background 'noise' in sequencing data. While early examples were largely written for the *de novo* assembly use case (where error correction is critical to the genesis of accurate assemblies) and were often relatively rudimentary (e.g. failing to incorporate important Phred quality scores in to their models) [15], more recent examples have been designed for the reconstruction of viral haplotypes and complex genomes [16-20], the correction of long read NGS data (i.e. produced by either the Pacific Bioscience or Oxford Nanopore platforms) using short read NGS data (e.g. Illumina, Ion Torrent) [21-25], and applications with non-uniform coverage (e.g. single-cell sequencing and RNA-seq) [18, 26-32]. In addition to models designed primarily for a biological use case, a number of error models have been designed for specific platforms, with developers making consideration for the systematic base-call errors that a given platform tends to produce (this is especially true for 454 pyro sequencing) [18, 21-35, 28, 32-36]. Of specific interest to this research is the Illumina platform, which is largely characterized by substitution errors [15].

While many of these models have proven indispensable across a number of problem domains, error correction software designed with the aim of effectively targeting somatic low frequency variants and separating them from background error has yet to be developed. This gap may have much to do with the aforementioned biological challenges involved with the confident

detection of low frequency variants or, alternatively, the relatively novel use case. Nonetheless, it is possible that a number of existing models may yield a basic solution with the capacity to server as a guide for the development of a novel error model optimized for their detection; in this respect, models that have been developed for the study of complex genomes are of particular interest. However, questions regarding the extent to which the introduction of (additional) background error modeling software into already complex variant calling pipelines may influence downstream processing exist.

Almost all error correction occurs very early in the data processing pipeline, which may have negative effects on the performance of downstream pipeline procedures. With out a doubt, any successful adoption of such software into an existing variant calling pipeline will require, at the least, careful consideration of the parameterization of these other modules and extensive benchmarking. In regard of the need for improved low frequency variant detection, the existing variant detection pipelines, and current error correction software, a sensible first step may be to devise a way to systematically benchmark existing error models within the context of a contemporary variant calling pipeline. The aims of this approach would be two-fold: First, to ascertain the effect of existing error correction software on the calling of low frequency variants, and second, to understand to what degree existing error models may help guide the development of a novel error model designed principally for the low frequency variant detection use case.

2.4 The GATK Variant Calling Pipeline

The Genome Analysis Toolkit (GATK) is the dominant variant calling pipeline in contemporary genomics [38-39]. Much of the original development of the software behind the GATK pipeline was pursued as part of the 1000 Genomes Project, a landmark study in the field of population genetics [40-42]. As a result, the software consists of multiple software modules that have been designed primarily for the detection of germline variants from whole genome and whole exome sequencing (WGS and WES) data produced by the Illumina platform.

More recently, the development of a somatic variant caller, MuTect (and subsequently MuTect 2, currently in active development and available for beta testing) in conjunction with ContEst, a module to estimate sample cross-contamination, has expanded the scope of the GATK pipeline into the realm of somatic variant detection [43-44]. Despite this expansion in scope, the continued inclusion of modules upstream from MuTect in the GATK somatic best practices for WGS and WES sequencing is worth examining. This is due to the immediately apparent differences between the use case that these modules were originally developed for and the

novel use case enabled by the MuTect family of variant callers; i.e. population-level characterization of high-confidence germline polymorphisms versus the detection of somatic single nucleotide variants (SNVs) of an individual's cancer. Implicit in the latter use case are low frequency variants, however, the focus of the GATK on the detection of high-confidence variants during the development may represent technical debt that must be accounted for if we are to progress further towards the confident detection of low frequency variants.

In this respect, of primary interest is the GATKs' Base Quality Score Recalibration module (BQSR). This software module seeks to improve the Phred quality scores of sequencing data by targeting mismatching base calls that do not appear in the reference of one or more user-provided database files (e.g. VCFs from dbSNP [45] and 1000 Genomes [40-42]) and then modeling known covariates of read group and base quality, such as dinucleotide context, machine cycle, and lane. The modification of base quality scores is done with the intention of improving the performance of down stream analyses (e.g. variant calling), however, there are a number of concerns regarding the retention of the BQSR module in a low frequency variant pipeline. For example, it has been found that the BQSR module may compromise detection of *de novo* variants by reducing their base quality scores [46] and that the module may also reduce sensitivity in regions of high divergence (in the cited research, HLA) with little to no improvement in precision [47]. Furthermore, platform base calling algorithms have improved since the introduction of the BQSR module in 2011, potentially obviating its use entirely [48]. Given the questions surrounding the rationale of using the BQSR module for the detection of low frequency variants and its use outside of the limited, 'best practice' use cases, it is worth examining further whether the BQSR module should be included in variant detection pipelines that target low frequency variants – especially if we are to explore the inclusion of additional error modeling software.

2.5 Research Questions

The confident, sensitive detection of MRD represents a powerful motivation for the pursuit of methodological improvements to the challenging problem of low frequency variant detection. In consideration of the potential of background error modeling approaches to separate true low frequency variation from the background error inherent to sequencing data, in addition to the anticipated challenges in the application of the GATK pipeline (and particularly the BQSR module) to the task of low frequency variant detection, the focus of this research was to answer the following questions.

- Is the BQSR module diminishing the ability of the GATK pipeline to detect low frequency variants?
- Can existing error correction software be leveraged to accurately model background error in somatic sequencing data within the context of the GATK pipeline (with or without BQSR), potentially enabling the sensitive detection of somatic low frequency variants?
- Assuming the optimal error correction software for somatic low frequency variant detection does not already exist, can the evaluation of existing models serve as a guide towards the development of a novel model designed for their detection?

3. Methods

3.1 Selection of Background Error Models

To answer these research questions, journal articles for 61 background error models or parent software suites were culled from a current review of error models for sequencing data [15], in addition to literature covering research on error models for application to cancer genomics, metagenomics & single-cell sequencing, *de novo* assembly, and population genomics. A set of exclusion criteria was developed with the aim of evaluating each model in a principled way. In terms of biology, only one exclusion criterion was defined; that the model should make no known haploid/diploid assumption(s) in order to maintain theoretical compatibility with the somatic low frequency variant use case. Other exclusion criteria rested largely on technical feasibility, which eliminated the majority of models from consideration. The model should:

- Be compatible with modern Illumina and/or Ion Torrent NGS short-read platforms
- Be compatible with a Unix-based operating system
- Be able to receive input of either BAM, SAM, FASTQ, or FASTA/QUALA file formats
- Output either BAM, SAM, FASTQ, or FASTA/QUALA corrected file formats
- Expose a functional API through the command line (i.e. be CLI-accessible)
- Compile correctly with available compilers (GCC 4.8.2 or GCC 5.3.0)
- Have clear, fit to purpose documentation
- Have compatible dependencies that can feasibly be installed

- Have relatively simple usage & setup (e.g. minimal pre or post-processing required, a single error correction step)

- Be licensed under an open-source license

An evaluation of model properties, as available, is presented in Table 8.2 (Supplemental). Ultimately, three background error models were implemented for the described iteration of this study; BFC, Lighter, and Bloocoo [49-52]. Given a compressed timeframe, their selection for this evaluation rested largely on feasibility – each of these error models provides adequate documentation, was able to be compiled with relative ease on the target system, and implements one or more Bloom filter data structures to maximize time-space efficiency [53]. Two additional models, Quorum and BayesHammer, were tested extensively, however, their deployment was unsuccessful for this iteration [27-28]. Deployment of Quorum was unsuccessful due to the burden placed on the distributed file system of the HPC cluster used; while a solution was found to this limitation, the enabled condition was not available for analysis at the time of this writing. BayesHammer was not deployed due to an issue parsing the validated FASTQ sequence files; this issue is being explored further and a solution is anticipated.

3.2 Selected Background Error Models

All three selected background error models make use of a similar approach to error correction known as the *k*-mer spectrum approach, originally developed for the problem of *de novo* assembly [54]. The *k*-mer spectrum algorithm works by first taking genome reads from the sequencing data and fragmenting them into substrings of a predefined length *k*, with each fragment known as a *k*-mer. If a given *k*-mer belongs to more than *S* reads, a threshold defined by the user, the *k*-mer is considered solid. In contrast, if the *k*-mer belongs to fewer than *S* reads, the *k*-mer is considered weak. The fragmentation process results in a distribution of *k*-mers G_k, of which the distribution of solid *k*-mers (the so-called *k*-mer spectrum) approximates the genome *G*. At this point, the error correction problem is cast as a spectral alignment problem, where the aim is to find the minimum number of steps to transform each given weak *k*-mer substring *s* into a a strong *k*-mer. The approach to error minimization may be done according to any edit distance deemed appropriate (e.g. Hamming, Levenshtein). While all three selected background error models use a variation of the *k*-mer spectrum approach, they exhibit a number of distinctive properties.

BFC: BFC fragments reads into substrings with length *k* and proceeds to classify them by counting and comparison against a user-defined threshold. The software then iterates over

reads searching for the longest substring comprised of strong *k*-mers; if no strong *k*-mer is found, search for strong *k*-mers 1 mismatch away. When an eligible strong *k*-mer is found, undergo substring extension. Eligible *k*-mers 1 mismatch away are corrected, but only one such *k*-mer is allowed per read during the substring extension step [49].

Bloocoo: Similar to BFC, Bloocoo first classified *k*-mers as strong or weak by counting. The software then simply corrects unambiguous reads (i.e. those with a clear best available correction). For reads that are ambiguous, reads are corrected through read extension using eligible strong *k*-mers and a majority vote algorithm. The software also attempts to avoid false positive corrections by verifying that each correction is supported by multiple solid *k*-mers [50-51].

Lighter: Rather than counting, Lighter randomly subsamples *k*-mers with replacement from the *k*-mer spectrum, omitting reads with N bases and those randomly ignoring a fraction of read, as specified by the parameter α (the authors offer guidelines on how to set this). The *k*-mers are then classified as strong or weak with a threshold strong *k*-mers at consecutive positions are combined and stored in a secondary Bloom filter. Each read then is matched to eligible consecutive *k*-mers and is corrected through read extension. Reads with equally eligible corrections from the consecutive *k-mers* are left uncorrected [52].

3.3 Overview of Computational Pipeline Module Evaluation System Prototype

A software prototype was developed with the aim of evaluating software modules for the GATK pipeline, according to the Broad Institutes' Best Practices for Somatic SNV and Indel Discovery in Whole Genome and Exome Sequence [55] (along with any additional software modules as required by the pipeline or the experimental and control conditions), in an audit-able and systematic way. Three primary goals for the system were established:

1. To provide a simplified and consistent API abstraction for the multiple software packages and pipeline procedures required (e.g. GATK, Samtools, Picard Tools).

2. To automate the enforcement of experimental design requirements throughout the pipeline via appropriate scope management.

3. To manage and enforce logging of each pipeline procedure and requisite steps to ensure the audit-ability of an experiment through system logs.

To do this, the concept of a pipeline 'procedure' was developed. Here, a procedure is defined as a set of one or more steps required to realize a discrete conceptual process, to be applied

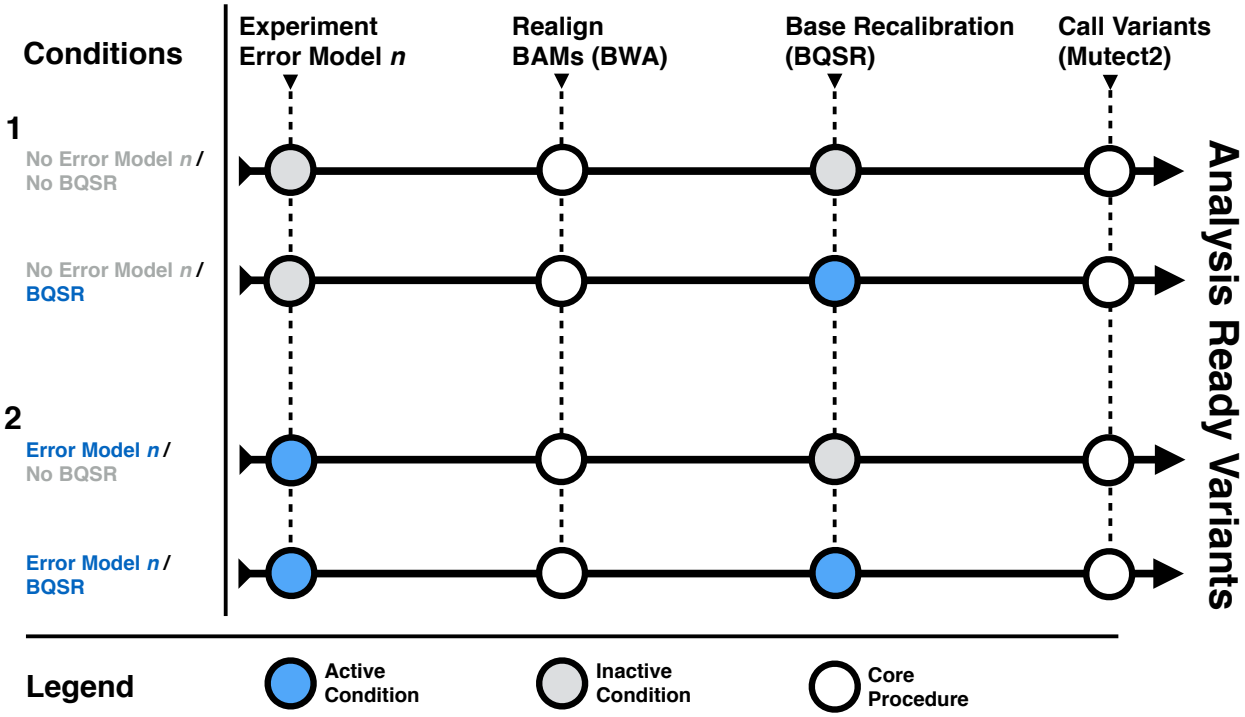
to the sequencing data passing through the pipeline; examples of procedures include ‘variant calling’, ‘error modeling’, ‘marking duplicate reads’, or ‘multiple sequence alignment’. While the focal point of a procedure is often a specific algorithm (e.g. BQSR for base recalibration), additional child steps may include data extraction, processing, transformation, and/or validation, in addition to basic custodial tasks such as file system management, data structuring, or the extraction of metadata.

Due to a lack of scoping features within the GNU bash environment, scope is managed almost entirely through namespacing that corresponds to the directory structure of the file system. Importantly, in addition to providing rudimentary scope management, the namespacing architecture also serves to enforce an appropriate and rigorous directory structure for each experimental condition. Leveraging the namespacing approach, a rudimentary state management system was also introduced to the prototype to prevent unnecessary duplication of data processing via a ledger of the pipeline management systems’ global state. This state management system functions by first validating that a step has yet to run given the user-defined experimental condition(s). This is done through a call to the state ledger prior to the initialization of each child step of a procedure. If the given state is not found, the pipeline executes the given step (within a procedure). Upon successful completion (verified by an exit status code of 0), the state is exported to the state ledger. Each state is represented via a unique string, the uniqueness of which is guaranteed by the inclusion of the entire namespace generated via each software API call required to initialize a procedure, along with an indication of the selected procedure, and an integer representing the completed step within the procedure.

The Prototype System was developed on the Exacloud HPC cluster at Oregon Health & Science University in a Unix environment (CentOS 6.5) using shell scripting (GNU bash 4.4.5) and Python (version 3.5.2). Core software dependencies for the pipeline include GATK 3.6 [38-39], Picard Tools 2.9.0 [56], Samtools 1.3.1 [57], BWA 0.7.15-r1140 [58], and BioPython 1.68 [59]. Language dependencies include GNU bash 4.4.5, Python 3.5.4, Java 8 (1.8.0_112) and Perl v5.24.0. Dependency management was handled using package management solutions, Linuxbrew 1.18 and Miniconda 4.3.11, when available. Depending on the requirements the software, compilation was performed using either GCC version 4.8.2 or 5.3.0. The prototype is publicly available under an open source BSD 3-clause license via a public source code repository hosted on Github (URL: https://github.com/greenstick/thesis-pipeline). All algorithm parameters can be found within their parent scripts as part of their invocations, located in the procedures directory (URL: https://github.com/greenstick/thesis-pipeline/tree/master/procedures).

3.4 Experimental Conditions

Figure 3.4.1 – Overview of Experimental Conditions Applied



Summary: The illustrated conditions were executed for each data set (Set 1, Set 2, & Set 3), on both tumor and normal. For aggregate condition 1, the goal was to understand whether BQSR influences the sensitivity of the GATK pipeline to low frequency variants when run according to the relevant GATK best practices. No error model was used, however, both child conditions were active (BQSR and no BQSR). For aggregate condition 2, *Error Model n* is representative of the given error model (Lighter, BFC, or Bloocoo), as each background error model condition would be subject to the procedures along the horizontal pipeline vectors (i.e. error modeling, realignment, the given base recalibration condition, and variant calling). The purpose of condition 2 was to assess the effect of each error model on low frequency variant detection both with and without BQSR. Overall, conditions 1 and 2 were designed to answer two questions, respectively: 1. Is the BQSR module diminishing the ability of the GATK pipeline to detect low frequency variants? and 2. Can existing error correction software be leveraged to accurately model background error in somatic sequencing data within the context of the GATK pipeline (with or without BQSR), potentially enabling the sensitive detection of somatic low frequency variants? Importantly, note that not all procedures are illustrated by this overview (hence the *Error model n* notation) and that realignment and variant calling are indicated to show roughly where in the pipeline each condition resides – i.e. the pipeline vectors do not represent a full enumeration of procedures run.

With the selection of three error models for evaluation, twenty-four multi-tiered (aggregate) conditions (6 control, 18 experimental) were designed for evaluation via the Prototype System.

The combination of conditions was designed to allow for the independent evaluation of the selected background error models, both within the context of the larger pipeline and in their relation to the BQSR module, in addition to the independent evaluation of the BQSR module itself. For each data set (Set 1, Set 2, & Set 3), the tumor normal pairs were run through the pipeline both with and without the BQSR module. Then, for each error model (Lighter, BFC, Bloocoo), a similar approach was taken. For example, under the BFC condition, each data set would be processed with BFC and both with and without BQSR (Figure 3.3.1).

3.5 Synthetic Whole Genome Sequencing Data

Table 3.5.1 – Synthetic Data Specifications

	Set 1	**Set 2**	**Set 3**
Aligner	BWA Backtrack	BWA Backtrack	BWA Backtrack
Mutation Types	SNV & SV (deletions, duplications, inversions)	SNV & SV (deletions, duplications, insertions, inversions)	SNV, SV (deletions, duplications, insertions, inversions) & INDEL
Cellularity	100%	80%	100%
Subclone VAFs	N/A	N/A	50%, 33%, 20%
Sex	Female	Female	Female
Original BAM	HCC1143 BL from TCGA Benchmark 4	HCC1954 BL from TCGA Benchmark 4	HCC1143 BL from TCGA Benchmark 4
Coverage (Tumor / Normal)	29.93x / 29.93x	30.97x / 30.98x	31.09x / 31.14x
***n* Reads (Tumor / Normal)**	964,638,036 / 964,739,976	1,122,018,486 / 1,122,503,732	1,127,400,640 / 1,129,059,456
% Positions with >= 20x Coverage (Tumor / Normal)	75.89% / 75.93%	78.09% / 78.15%	78.75% / 78.96%
BAMSurgeon Commit	https://github.com/adamewing/bamsurgeon/tree/12862f1127cd513186b91444cfb03dc02bbd69ee	https://github.com/adamewing/bamsurgeon/tree/df7bc9148b490ca2dbdc20ba192ee4c7d676b4d1	https://github.com/adamewing/bamsurgeon/tree/6485d9c756d5f7d5b2025d13a21615be8db4be20
Reference Genome	Homo_sapiens_assembly19.fasta	Homo_sapiens_assembly19.fasta	Homo_sapiens_assembly19.fasta

Source: Modified from: https://www.synapse.org/#!Synapse:syn312572/wiki/62018

The data analyzed with the Prototype System was comprised of three, interleaved paired-end tumor-normal WGS data pairs (Set 1, Set 2, & Set 3; 6 BAM files total), with each tumor file synthetically generated from its corresponding normal pair by the BAMSurgeon software for the *2014 ICGC-TCGA-DREAM Somatic Mutation Calling Challenge* [60]. For the challenge, normal files were sourced from the TCGA Mutation Calling Benchmark 4 [61]. Each set exhibited a coverage of approximately 30x, but varied in complexity. Ordered from least to most complex, Set 1 exhibited 100% purity and 3,537 somatic single nucleotide variants (SNVs), Set 2 reduced sample purity of 80% (20% normal contamination) with 4,332 SNVs, and Set 3 100% purity with 7,903 SNVs and variant allele fractions of 50%, 33%, and 20% (Table 3.4.1). The aligned data (released in BAM format [57]) was downloaded using the GeneTorrent download client [58].

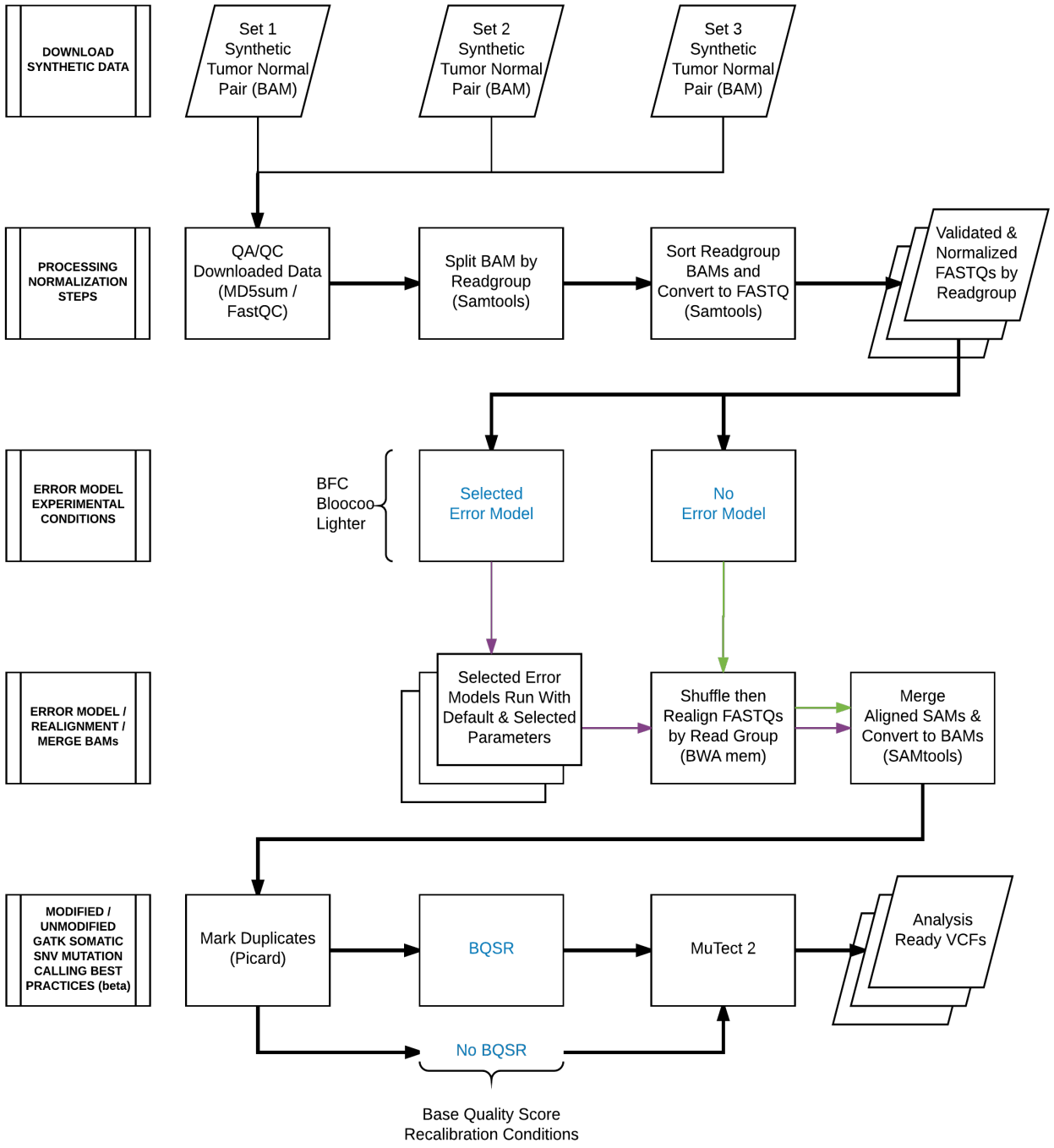
3.6 General Pipeline Implementation

The following procedures were performed for both the tumor and normal BAM files from all three sets (Set 1, Set 2, & Set 3) (Figure 3.5.1). A notable exception is that of MuTect 2, which receives both normal and tumor BAM files as inputs. MuTect 2 was run for each of the three sets (Set 1, Set 2, & Set 3) and was run twice for each set – once on data processed with BQSR, and once on data processed without BQSR. Also note that the hg19 reference was indexed by BWA once prior to the execution of any pipeline conditions.

BAM to FASTQ: The synthetic data downloaded for the study consisted of aligned BAM files originally aligned with BWA backtrack algorithm [59]. Due to the incompatibility of these original alignments with the targeted GATK Best Practices for Somatic SNV and Indel Discovery in Whole Genome and Exome Sequence, these alignments were removed via reversion to FASTQ format. This process was accomplished first by splitting the merged BAM files by read group, coordinate sorting the read group-level BAM files, and then reverting each read group-level BAM to FASTQ format. All these steps were executed using Samtools.

Apply Error Model n: Each read group-level FASTQ was then processed individually using the background error model selected for the given experimental condition. For Bloocoo, this procedure required the extra steps of separating the sequence and quality data, applying the error model, and then merging the output corrected sequences with quality data. Similarly, Lighter required the separation of the interleaved paired-end reads into single-end reads. Upon completion, the error corrected single-end reads were re-interleaved into a single paired-end FASTQ.

Figure 3.6.1 – Simplified Diagrammatic Overview of Experimental Pipeline



Summary: Starting at the top, Sets 1, 2, & 3 were first downloaded. Second, MD5 check sums were computed, FastQC was run, the BAMs were split by read group, and then converted to FASTQ format. Third, error modeling was applied as required by the given experimental condition, reads were then shuffled and aligned before the read group BAMs were merged back together. Fourth, the GATK pipeline was run using Picard Tools Mark Duplicates, the required BQSR condition (i.e. BQSR / No BQSR), and finally somatic variant calling with MuTect 2.

into paired-end FASTQs. BFC required no additional processing. These child steps were done using custom Python scripts available on the Prototype System's public repository (URL: https://github.com/greenstick/thesis-pipeline/tree/master/utils).

Alignment: Once a given error correction procedure had completed, the alignment procedure was initiated. For each experimental condition, each read group-level FASTQ was shuffled using a custom Python script, also available on the public repository (URL: https://github.com/greenstick/thesis-pipeline/tree/master/procedures/shuffle-fastq.py). This shuffling was done to avoid bias associated with the previous alignment embedded in the downloaded BAM files. After shuffling, the FASTQ files were aligned using the default parameters for the BWA-mem algorithm, the aligner recommended by the GATK best practices documentation. The resulting alignments were then annotated with their corresponding read groups using Samtools addreplacerg prior to being unified into a merged BAM using Samtools merge, with the resulting BAM now mimicking the original BAM file, but with a BWA-mem alignment and the designated background error correction.

Mark Duplicates: The error corrected, aligned, and merged BAM file was then passed to the core GATK pipeline. First, the BAM file was coordinate sorted using Samtools sort and then indexed using Samtools index. Next, duplicate reads were marked using Picard Tools mark duplicates algorithm. This was followed by a secondary indexing of the merge BAM file.[2]

Base Recalibration: For conditions requiring BQSR, the base recalibration procedure was then performed. This procedure consisted of four steps. First, run the BaseRecalibrator module to build a model of covariation for the data, as detailed in Section 2.4; second, run the BaseRecalibrator module again to build the second model of covariation; third, run the AnalyzeCovariates module to generate plots of the base recalibration; and finally, write the recalibrated base quality scores to a new BAM file. Again, the new file was indexed with Samtools.

Call Variants: The final step of the GATK pipeline is to call variants. To do this, cross sample contamination was first estimated using the recommended contamination estimation tool, ContEst [43]. The output contamination estimation file was parsed for the estimated value, which was then converted from a percentage to a proportion for input into the MuTect2 somatic variant caller [44]. In addition to the contamination estimation, the recommended

[2] After each step of the GATK pipeline, the BAM files were indexed using Samtools. Indexing of the BAM files was often superfluous, however, this was done to ensure no warnings would be generated by the GATK tools processing these files.

dbSNP and COSMIC database VCF files, available in the GATK resource bundle [45, 63-65], and the processed, paired tumor-normal BAM files were also input to MuTect2. The output of MuTect2 is a VCF file of both somatic and germline variant calls annotated accordingly.

All commands were executed using either their software defaults or, if available, recommended parameterization. All GATK, Picard Tools, and BWA commands were executed according to best practices documentation available on the GATK website. The complete set of commands issued for each software tool is available as part of the procedure scripts on the public repository (URL: https://github.com/greenstick/thesis-pipeline/tree/master/procedures).

3.7 VCF Analysis

VCF analyses were performed using custom Python scripts available on the analyses public repository (URL: https://github.com/greenstick/thesis-analyses/tree/master). For each error model condition, both the BQSR and No BQSR VCF files, along with the truth set provided by the *2014 ICGC-TCGA-DREAM Somatic Mutation Calling Challenge* [60], were parsed using pyVCF (URL: http://pyvcf.readthedocs.io).

Within a VCF file, each variant was assessed to be unique via the combination of chromosome, position, reference allele, and alternate allele values. These values were combined into a unique string and then hashed using the SHA1 message digest algorithm to enable time-space efficient set computations of VCF statistics and pipeline performance characteristics (e.g. sensitivity, specificity, accuracy). Sensitivity-specificity curves were computed along with their accompanying AUC to aid in the assessment of each conditions' performance using 100 tumor LOD score (tLOD) thresholds ranging from 0 to 100. The optimal tLOD for each pipeline, defined as the tLOD threshold that maximized the sum of sensitivity and specificity, was also computed.

4. Results

4.1 General Performance Evaluation by Condition

The performance of each condition was measured via sensitivity-specificity curve AUC across the full variant allele frequency range (0.0 - 1.0 VAF), computed from tLOD cutoff thresholds ranging from 0 to 100. The sensitivity-specificity curve AUCs for each dataset and aggregate condition were together summarized for comparison (Table 4.1.1). The Bloocoo BQSR aggregate condition demonstrated the best performance across all tLOD thresholds for Set 1 (AUC = 0.9746). For Set 2, the best performing was the Bloocoo No BQSR aggregate condition

(AUC = 0.9610). Similar to Set 1, the best performing Set 3 was the Bloocoo BQSR aggregate condition (AUC = 0.9339).

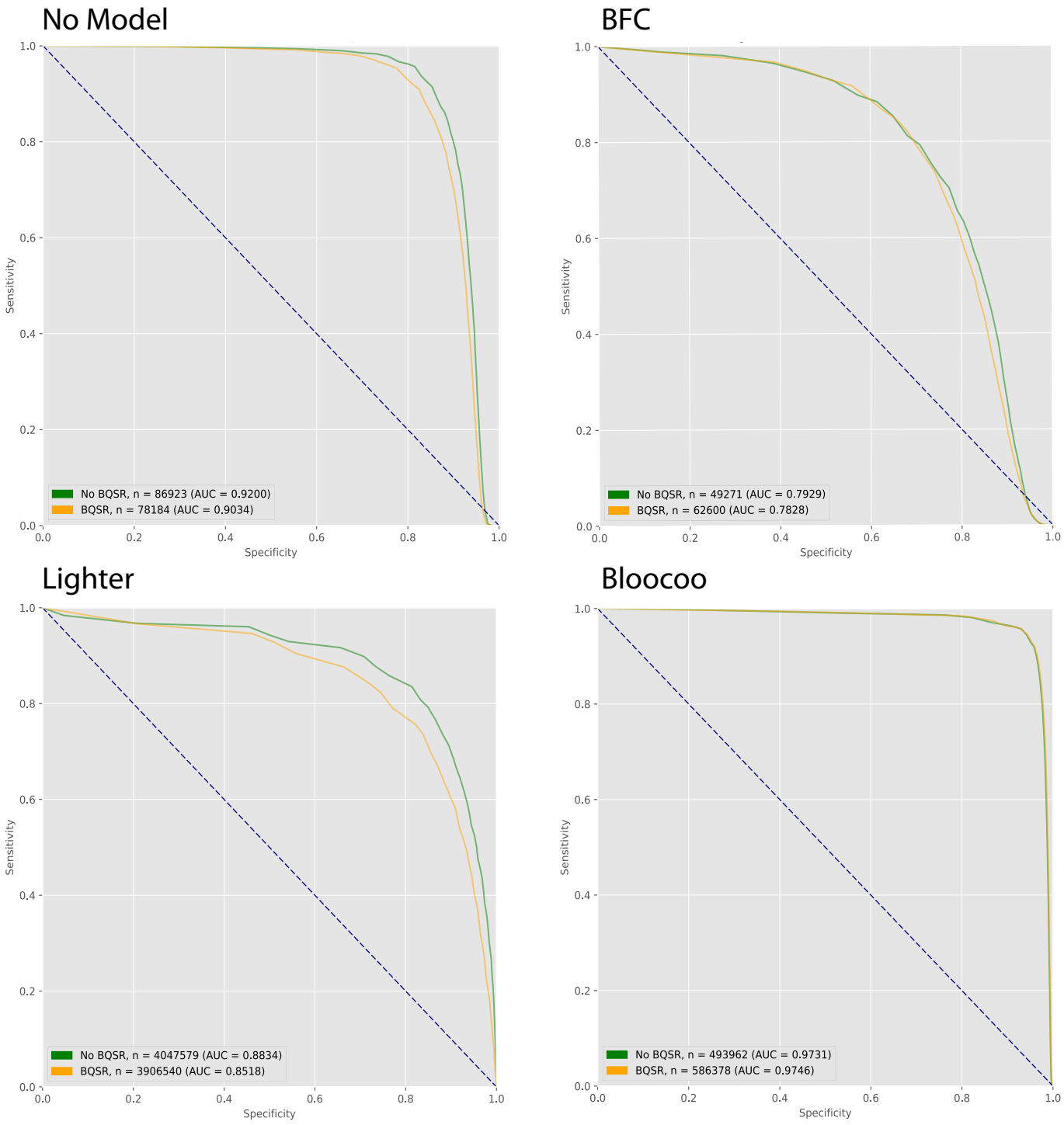
Table 4.1.1 – Sensitivity-Specificity AUC by Condition

Background Error Model	BQSR Condition	Set 1	Set 2	Set 3
No Model	**No BQSR**	0.9200	0.9466	0.8909
	BQSR	0.9034	0.8966	0.8333
BFC	**No BQSR**	0.7929	0.7723	0.7819
	BQSR	0.7828	0.8063	0.7961
Bloocoo	**No BQSR**	0.9731	**0.9610**	0.9335
	BQSR	**0.9746**	0.9558	**0.9339**
Lighter	**No BQSR**	0.8834	0.8596	0.8006
	BQSR	0.8518	0.8102	0.7996

For Set 1, the Bloocoo background error model, combined with the No BQSR and BQSR conditions, were found to exhibit the best performance in terms of sensitivity and specificity across all tLOD thresholds with AUCs of 0.9731 and 0.9746, respectively (Figure 4.1.1). An observable difference was seen between the BQSR and No BQSR conditions for the No Model and Lighter background error model conditions; however, no difference was seen between the BQSR and No BQSR conditions for the BFC and Bloocoo background error model conditions. It appears that in some cases, background error models may mitigate the effects of the BQSR module. This was especially true for Bloocoo, which demonstrated this mitigation behavior across all three datasets.

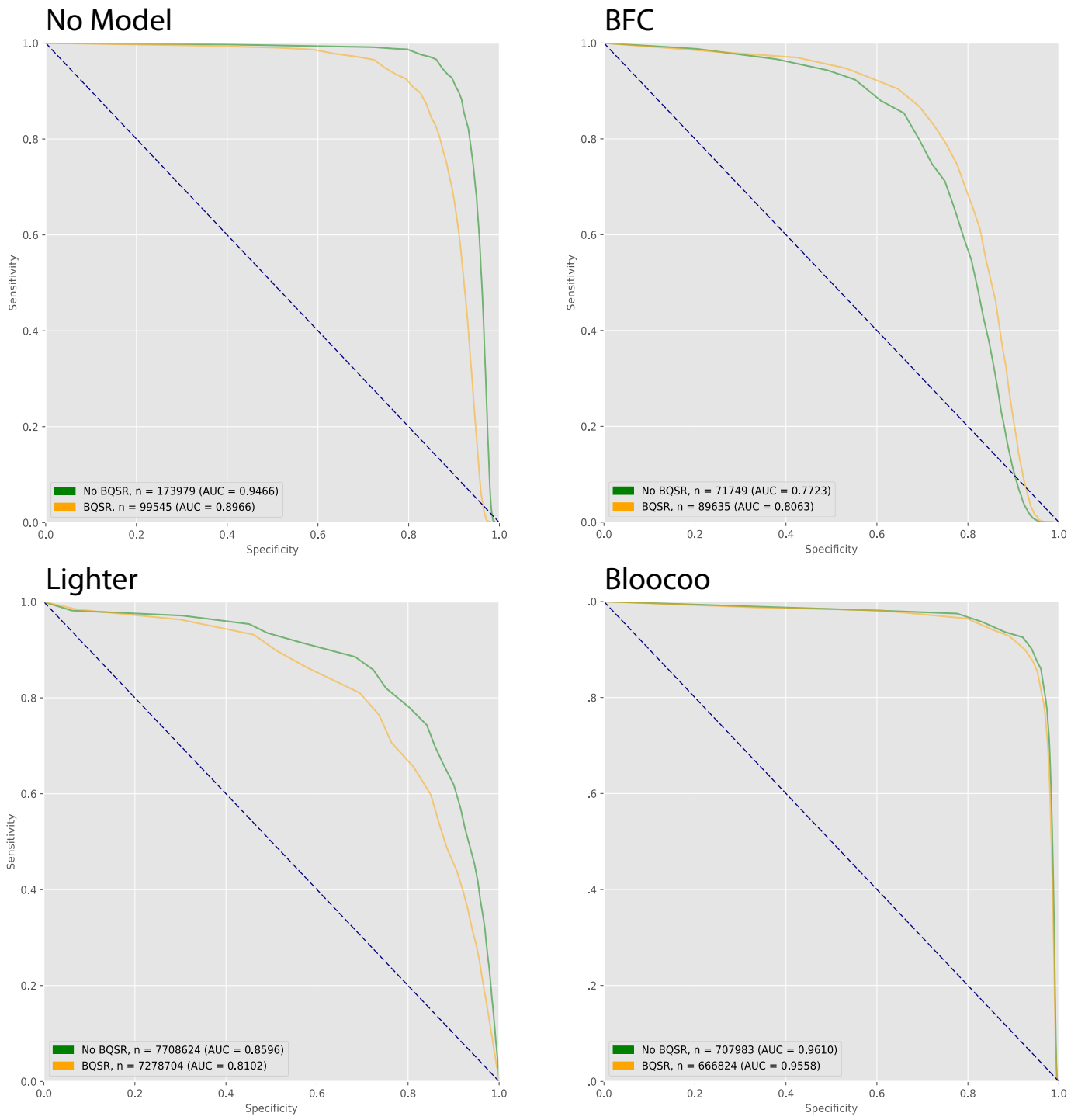
For Set 2, aggregate conditions incorporating the Bloocoo background error model conditions were again found to exhibit the best performance in terms of sensitivity and specificity across all tLOD thresholds, with rough equivalency between the No BQSR and BQSR conditions, yielding AUCs of 0.9610 and 0.9558, respectively (Figure 4.1.2). These aggregate conditions were followed closely by the No Model No BQSR aggregate condition (AUC = 0.9466). Notably, the BFC background error model exhibited improved performance in concert with the BQSR module in contrast to its No BQSR condition, with AUCs of 0.8063 and 0.7723, respectively – an improvement over both comparable Set 1 aggregate conditions. Generally, the 20% normal contamination present in Set 2 did not appear to meaningfully compromise sensitivity and specificity (as compared to comparison to Set 1) for the No Model condition.

Figure 4.1.1 – Set 1 Performance Comparison BQSR Versus No BQSR



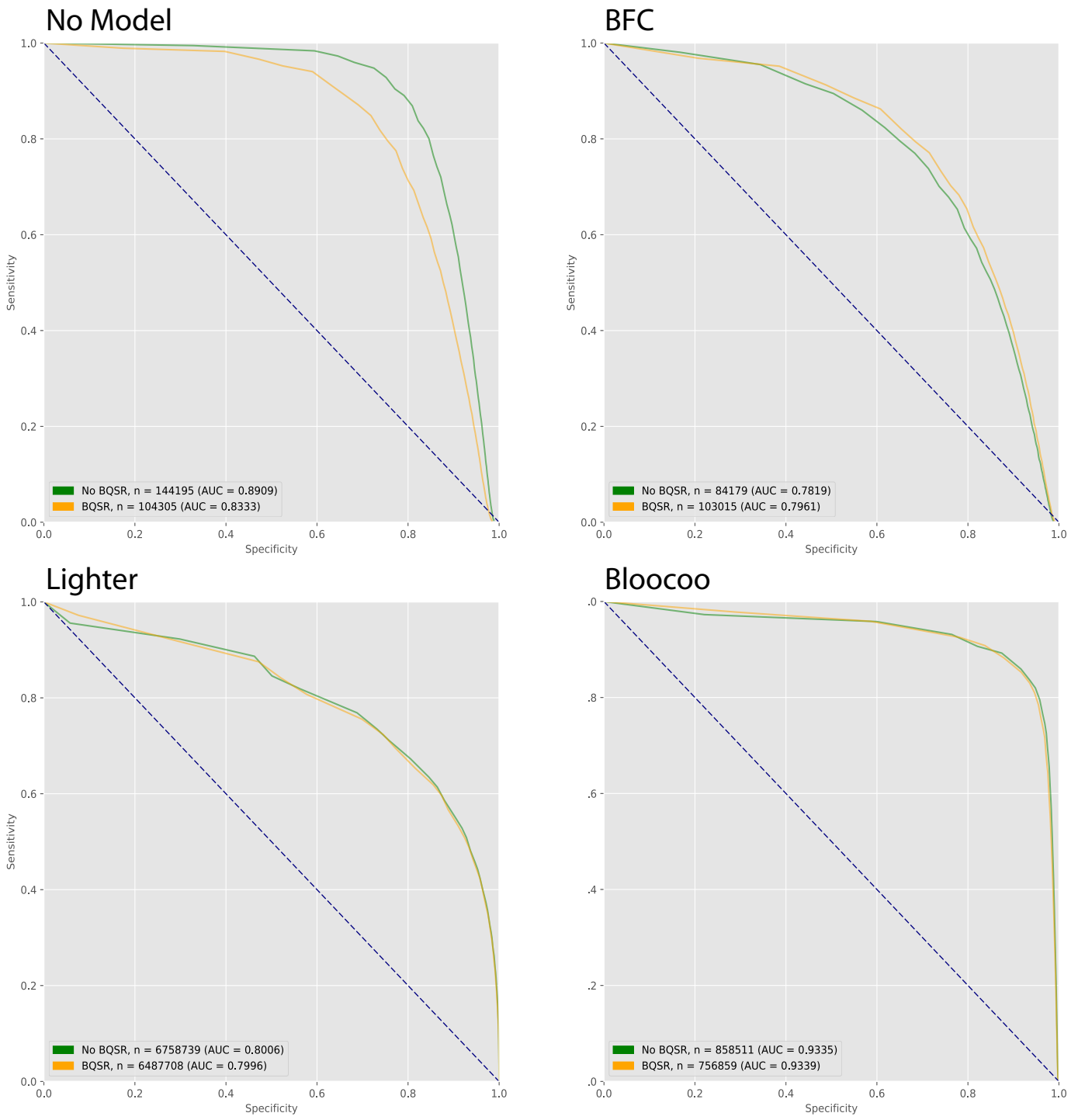
Summary: Clockwise from top left: No Model, BFC, Bloocoo, Lighter. Set 1 comparison sensitivity-specificity curves across all VAF ranges (0.0 - 1.0) for the No BQSR and BQSR conditions by Background Error Model Condition.

Figure 4.1.2 – Set 2 Performance Comparison BQSR Versus No BQSR



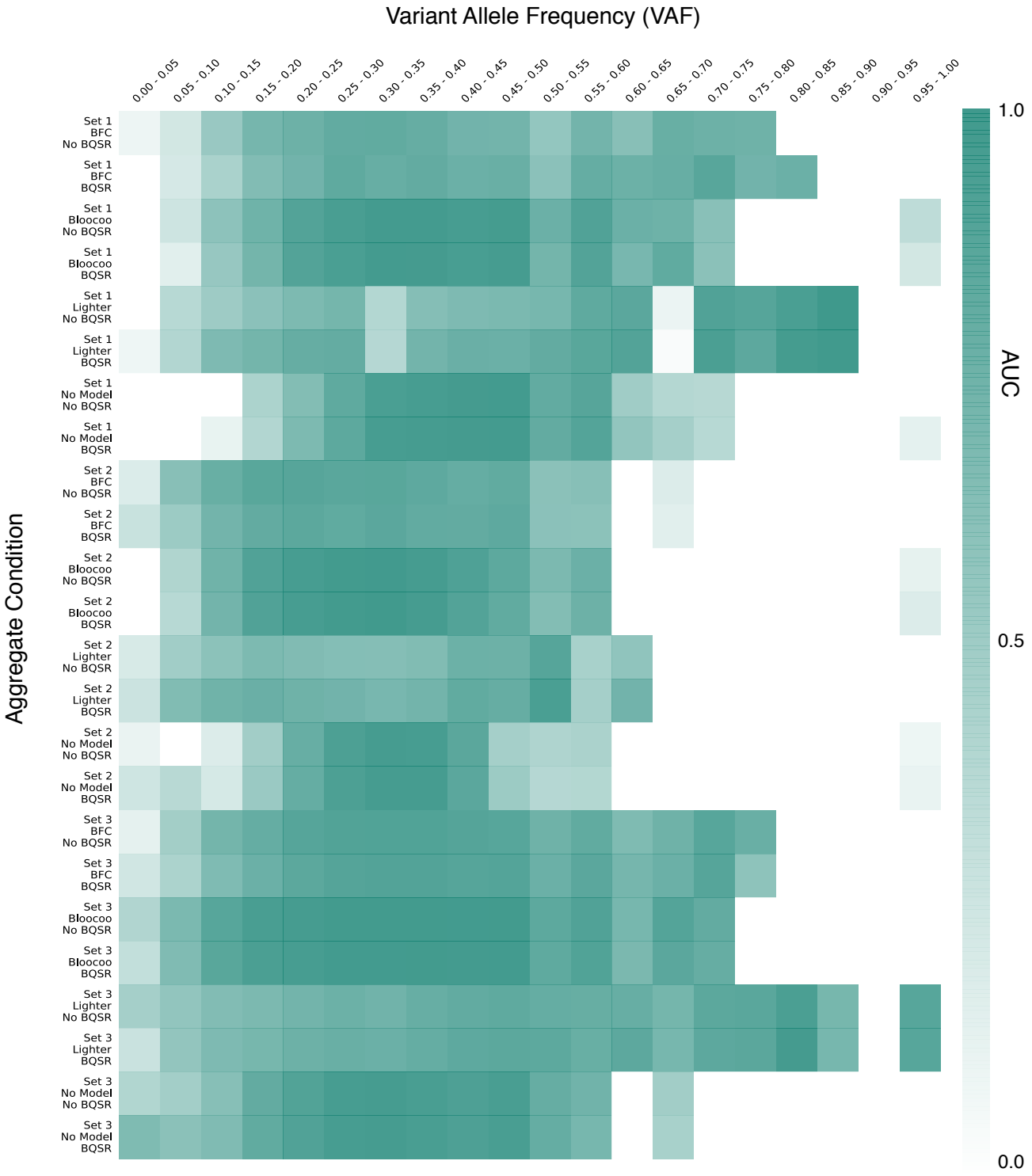
Summary: Clockwise from top left: No Model, BFC, Bloocoo, Lighter. Set 2 (exhibiting 20% tumor contamination) comparison sensitivity-specificity curves across all VAF ranges (0.0 - 1.0) for the No BQSR and BQSR conditions by Background Error Model Condition.

Figure 4.1.3 – Set 3 Performance Comparison BQSR Versus No BQSR



Summary: Clockwise from top left: No Model, BFC, Bloocoo, Lighter. Set 3 (exhibiting clonality) comparison sensitivity-specificity curves across all VAF ranges (0.0 - 1.0) for the No BQSR and BQSR conditions by Background Error Model Condition.

Figure 4.1.4 – Condition Sensitivity-Specificity AUC Across 0.05 VAF Ranges



Summary: A heat map of sensitivity-specificity curve AUC values for each aggregate condition by variant allele frequency range; each sensitivity-specificity curve was computed within a 0.05 VAF range interval. The teal color increases in saturation as AUC increases.

For Set 3, the difference between the No BQSR and BQSR conditions appeared to be effectively eliminated by the use of a background error model. That is, for the BFC, Bloocoo, and Lighter background error model conditions, the No BQSR and BQSR conditions exhibited comparable sensitivity and specificity (Figure 4.1.3). Again, the Bloocoo background error model condition yielded the highest AUCs of 0.9335 and 0.9339 for the No BQSR and BQSR conditions, respectively. The No Model and No BQSR aggregate condition exhibited superior performance to the No Model and BQSR aggregate condition, with AUCs of 0.8909 and 0.8333, respectively. Generally, AUCs were lower across Set 3 as compares to Sets 1 and 2, likely on account of the clonality exhibited.

To better understand sensitivity and specificity across VAF ranges for each condition, sensitivity-specificity curves were also computed across 0.05 VAF range intervals (e.g. variants with allele frequencies greater than 0.30 VAF and less than or equal to 0.35 VAF). The computed AUCs for each ranged curve were then projected as a heat map for examination of broad patterns in the performance of the aggregate conditions (Figure 4.1.4). In this visual, we see aggregate condition on the y-axis by VAF interval on the x-axis. The teal color represents AUC value, with increased color saturation serving as the mark of a greater AUC in the given range.

In general, the heat map is largely as expected – above 0.5 VAF we see much more empty space where AUCs of 0.0 are reported. This is in line with the biological assumption that the vast majority of variants will be heterozygous. In addition, we also see a tapering of the performance band below the 0.15 VAF mark, indicating either a lack of low frequency variants in the data, the challenges in their detection, or a combination. However, we also see that each background error model exhibits remarkably different performance characteristics. For example, the best performing background error model conditions, No Model and Bloocoo, each exhibit a clustering of predictions below the 0.5 VAF threshold. This clustering is less apparent for both BFC and Lighter, with Lighter especially exhibiting banding that extends much past the 0.5 VAF limit in both Sets 1 and 3. Regarding the BQSR and No BQSR conditions, we see that the large differences seen for some aggregate conditions in figures 4.1.1 - 4.1.3 are the result of a combined effect across VAF ranges rather than differences clustering in a specific VAF range.

4.2 Optimal Sensitivity-Specificity Threshold Analysis

While the AUC statistic is able to give a general sense of how each aggregate condition performed across a given range of tumor LOD thresholds by variant allele frequency, it fails to reveal which tumor LOD and aggregate condition combination offers the best performance.

Table 4.2.1 – Optimal Thresholds by Sensitivity-Specificity

Synthetic Data	Background Error Model	BQSR Condition	Tumor LOD Threshold	Sensitivity	Specificity	Accuracy
Set 1	**No Model**	**No BQSR**	16	0.9368	0.8296	0.8339
		BQSR	18	0.8844	0.8364	0.8385
	BFC	**No BQSR**	13	0.8122	0.6818	0.6906
		BQSR	14	0.8212	0.6821	0.6895
	Bloocoo	**No BQSR**	12	0.9299	**0.9532**	**0.9530**
		BQSR	12	**0.9448**	0.9421	0.9421
	Lighter	**No BQSR**	15	0.8083	0.8323	0.8322
		BQSR	15	0.7366	0.8381	0.8380
Set 2	**No Model**	**No BQSR**	12	**0.9472**	0.8740	0.8758
		BQSR	16	0.8748	0.8381	0.8397
	BFC	**No BQSR**	11	0.8006	0.6917	0.6979
		BQSR	11	0.8287	0.7251	0.7298
	Bloocoo	**No BQSR**	11	0.9011	0.9403	0.9400
		BQSR	11	0.8746	**0.9431**	**0.9427**
	Lighter	**No BQSR**	15	0.6993	0.8588	0.8587
		BQSR	11	0.7640	0.7365	0.7365
Set 3	**No Model**	**No BQSR**	13	**0.8690**	0.8093	0.8156
		BQSR	14	0.8180	0.7383	0.7499
	BFC	**No BQSR**	13	0.7379	0.7132	0.7172
		BQSR	13	0.7323	0.7407	0.7396
	Bloocoo	**No BQSR**	11	0.8353	**0.9364**	**0.9347**
		BQSR	11	0.8301	0.9357	0.9337
	Lighter	**No BQSR**	15	0.6136	0.8644	0.8639
		BQSR	15	0.5965	0.8737	0.8732

Summary: Statistics for tLOD that maximizes sensitivity and specificity by aggregate conditions. Conditions highlighted in blue exhibit the greatest sum of sensitivity and specificity by dataset. Bolded statistics indicate the greatest value for that statistic in given dataset.

Any implementation of a variant calling pipeline requires a validated (optimal) tumor LOD threshold for operation outside the research context (this is especially true given that a truth set for the variants in a sample being tested would not be available). For the purposes of this study, the optimal threshold was naïvely defined as the tumor LOD score threshold which maximized the sum of sensitivity and specificity (Table 4.2.1).

In regards to sensitivity, a measure highly important for the detection of low frequency variants, the No Model, No BQSR aggregate condition demonstrated the best performance for Sets 2 and 3 (tLOD = 12, sensitivity = 0.9472 and tLOD = 13, sensitivity = 0.8690, respectively). For Set 1, the Bloocoo BQSR aggregate condition demonstrated the best performance (tLOD = 12,

sensitivity = 0.9448). In terms of specificity, the Bloocoo background error model demonstrated the best performance across all data sets. For Set 1 and Set 3, the highest specificity was achieved in combination with the No BQSR condition (tLOD = 12, specificity = 0.9532 and tLOD = 11, specificity = 0.9364, respectively). For Set 2, the aggregate Bloocoo BQSR condition exhibited the highest specificity (tLOD = 11, specificity = 0.9431). Accuracy was also reported for each aggregate condition, across the synthetic data sets, however, the measure is perhaps not appropriate for the problem of variant detection (i.e. given the ratio between true positives and true negatives will likely always be unbalanced, the accuracy statistic could precipitate an improper assessment of classification performance for detecting positive cases).

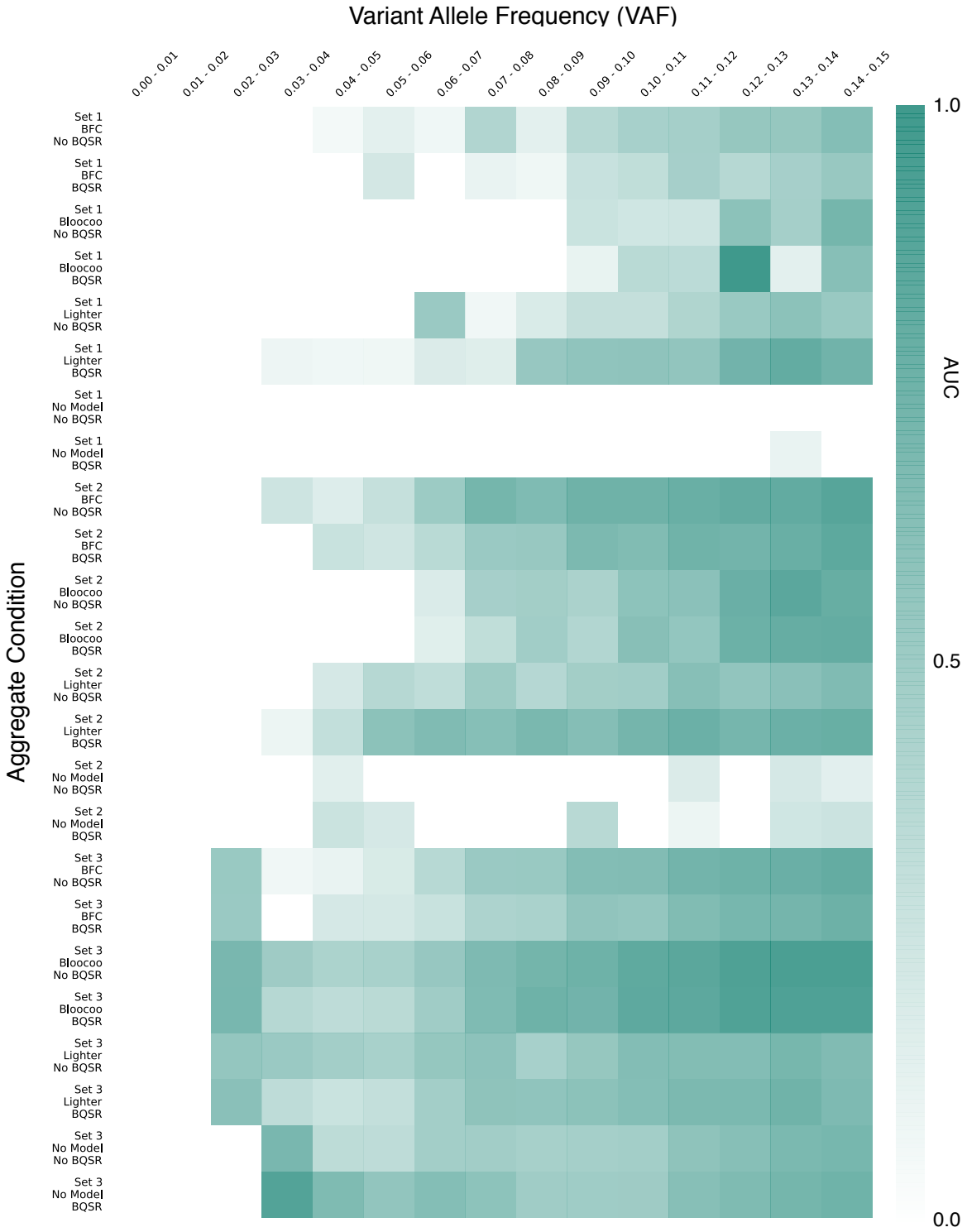
Importantly, while the No Model No BQSR condition for Set 3 demonstrated the best sensitivity, the combined sensitivity and specificity for the condition was much lower when compared to the Bloocoo BQSR condition (Sensitivity = 0.8690, Specificity = 0.8093; Sensitivity = 0.8353, Specificity = 0.9364, respectively). Thus, a worthy consideration here may be how sensitivity should be weighted compared to specificity, the ~3.5% loss in sensitivity may be worth the over ~13% improvement in specificity.

4.3 Performance Evaluation by Condition for Detection of Low Frequency Variants

A second heat map of sensitivity-specificity curve AUCs was generated using 0.01 width VAF ranges (e.g. 0.05 - 0.06 VAF) with the aim of understanding the performance of the aggregate conditions in the low frequency variant range (Figure 4.3.1). Unfortunately, computation of AUC were found to be compromised for allele frequency ranges below 0.15 VAF (Sets 1 and 2) and 0.1 VAF (Set 3), yielding AUCs with a zero value. While the low coverage (30x) of the data or poor performance by the aggregate conditions may have contributed to this, the likely culprit was the dearth of low frequency variants in the data. With no true positives in the lower ranges (i.e. VAF < 15%), we can see which conditions appear to generate true negative predictions in the lower ranges.

While the heat map does appear to reveal a trailing performance band across most aggregate conditions the only dataset worth considering is Set 3, which was the only set that exhibited true variants in the frequency range between 0.05 and 0.15 VAF (66 between 0.05 - 0.10 VAF; 10 between 0.10 - 0.15 VAF). Thus, Sets 1 and 2 are known to be entirely (false positive) noise

Figure 4.3.1 – Condition Sensitivity-Specificity AUC Across 0.01 Low VAF Ranges



Summary: A heat map of sensitivity-specificity curve AUC values for each aggregate condition within the VAF <= 0.15 range. Note low frequency variant range was previously defines as VAF <= 0.05. Each sensitivity-specificity curve was computed within a 0.01 VAF range interval.

below 0.15 VAF. If we consider this, it does appear that an increase in heat saturation exist in the 0.07 - 0.15 VAF range for Set 3, however, this is hardly conclusive. Perhaps the most striking features are the largely uniform bands presented by the No Model conditions for Sets 1 and 2. If we look to Figure 4.1.1, we see that the VAF ranges exhibiting the highest sensitivity specificity AUCs for these conditions appear to cluster together. This may indicate that their performance is largely attributable to a lack of noise contributed by a background error model.

5. Discussion

5.1 BQSR and Background Error Model Performance

For all synthetic datasets, the No Model No BQSR aggregate condition was found to be more sensitive than the No Model BQSR condition. Furthermore, for Sets 2 and 3, those exhibiting impurity and clonality, BQSR was found to also reduce specificity. This tentatively supports the alternate hypothesis for the first research question; that the BQSR software may diminish the ability of the GATK variant calling pipeline to detect low frequency variants. However, given that only Set 3 exhibited low frequency variants and that the bulk of true variants within that set were not low frequency variants, the extent to which low frequency variants contributed to these results is highly debatable. Importantly, this surfaces a bigger question: Is the BQSR software improving performance at all? Prior research has found this to be questionable, with an explanation being that the base calling algorithms on contemporary platforms are much improved over their predecessors, removing the need for BQSR software entirely [47-48].

Clear performance differences were seen between background error model conditions. While Bloocoo regularly exhibited the most balanced performance according to its sensitivity-specificity AUC, for Sets 2 and 3 the No Model condition exhibited superior sensitivity while maintaining good specificity (Table 4.2.1). For Set 3, considering the complete sensitivity specificity tradeoff alongside a preference toward higher sensitivity, the specificity exhibited by the Bloocoo No BQSR aggregate condition was far superior, while yielding only a mild reduction in sensitivity, in comparison to the No Model No BQSR condition (which exhibited the highest sensitivity). Lighter and BFC were found to perform relatively poorly by comparison, however, it's unlikely this can be put down to the software itself – the lack of parameter optimization may have handicapped these background error model conditions. Similarly, it's possible that for Bloocoo the default parameters were more appropriate for the synthetic data. This may be supported by the fact that each software implements a variation of the *k*-mer spectrum-based approach (Table 5.1.1).

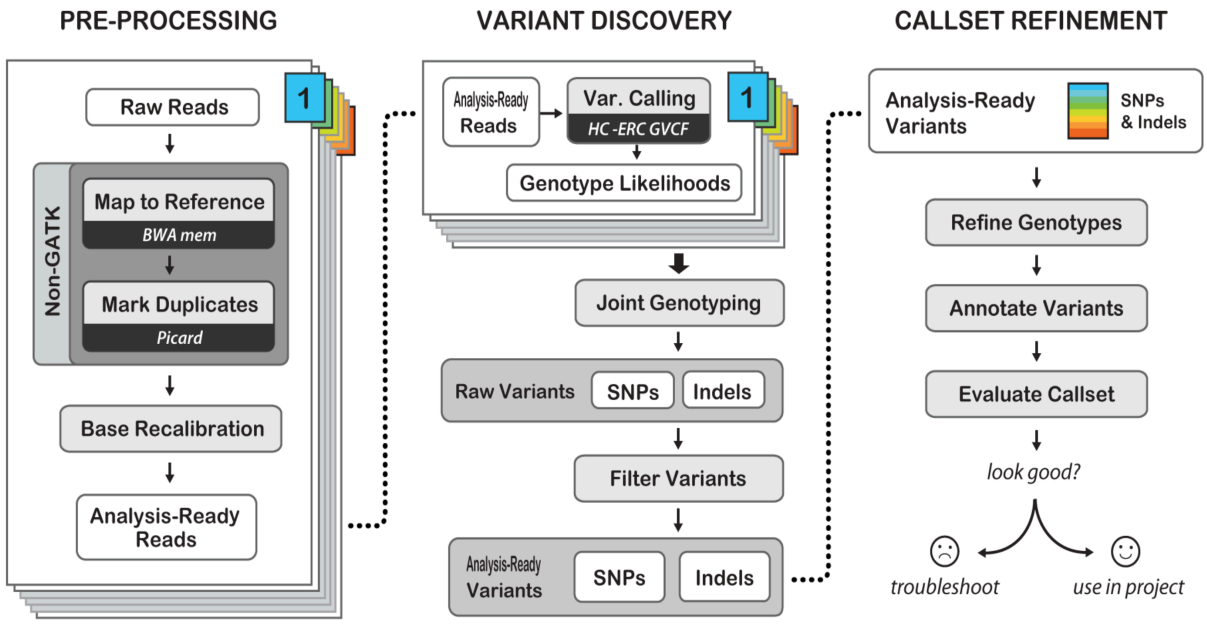
Table 5.1.1 – Overview of Error Model Characteristics

Background Error Model	Greedy Algorithm	Approach	Targeted Error Type	Quality Aware?	Basic Approach
BFC	No	*k*-mer spectrum	Substitutions	Yes	Classify *k*-mers as strong or weak by counting. For each read, find longest trusted substring by extension of solid *k*-mer on both ends. If no trusted *k*-mer matches the read, enumerate all trusted *k*-mers 1 mismatch away. If a trusted *k*-mer is found, undergo substring extension [49]. Reads marked uncorrectable if no eligible *k*-mer is found or correction requires multiple mismatching *k*-mers.
Bloocoo	Yes	*k*-mer spectrum	Substitutions	No	Classify *k*-mers as strong or weak by counting. Correct reads with an unambiguous error to their corresponding strong *k*-mers. Ambiguous reads are corrected through extension of the strong *k*-mer and a majority vote algorithm. False positives are avoided by verifying corrections are supported by multiple solid *k*-mers [50, 51].
Lighter	Yes	*k*-mer spectrum	Substitutions / Indels	No	Randomly subsample *k*-mers with replacement from the *k*-mer spectrum, filter reads with N bases, place acceptable *k*-mers into first Bloom filter. Classify *k*-mers and strong or weak with threshold and combine consecutive trusted *k*-mers and insert into second Bloom filter. For each read, find longest consecutive *k*-mer in second bloom filter that matches. Correct error reads through extension, leaving ambiguous corrections uncorrected [52].

Perhaps the most important takeaway here is that a background error model condition (Bloocoo) without BQSR outperformed its No Model control. This incentivizes further exploration of background error models for somatic variant calling – clearly, with further research and validation, they may be of value for the problem of variant detection. The more relevant question to this study, however, is whether a background error model can in fact enhance the detection of low-frequency variants. This question remains largely unanswered. It is notable, however, that for Set 3 the No Model condition exhibited the greatest sensitivity. Could it be that the error correction software evaluated stripped out the true variants in the 0.05 - 0.15 range? Unfortunately, given the complete lack of low frequency variants (i.e. < 0.05 VAF) in the data, this will remain an open question for now.

5.2 Best Practices & *in silico* Experimental Design

Figure 5.2.1: The Genome Analysis Toolkit Best Practices for Germline SNPs and Indels in Whole Genomes and Exomes – June 2016



Summary: Best practice documents for variant detection pipelines rarely articulate all computational steps required for reproducibility. For example, the best practices above indicate a single 'Base Recalibration' step, which in reality consists of applying the Base Quality Score Recalibration, Analyze Covariates, & Print Reads GATK modules to the sequencing data in four separate steps – the construction of two models of quality scores covariates, a visualization (intended to serve as a sanity check), and then the application writing of a BAM file with the recalibrated quality scores. Despite minor oversights in this figure, the documentation for this particular procedure, found elsewhere, is quite good. More revealing are the details for evaluating the callset output by this GATK pipeline – does the callset 'look good?', if yes, use it, if no, troubleshoot. Unfortunately, the 'look good' guidance embodies much of the challenge with the 'best practices' presented here – the lack of clarity requires significant expertise to reconcile exactly what dependencies and actions should be taken to execute the procedure. This gap precipitates ample opportunity for a researcher to introduce bias into an experiment and waste substantial time and resources. It is noteworthy that this documentation is far superior to the documentation available for other GATK use cases, one of which was pursued for this study.

Source: GATK Best Practices Documentation (URL: https://software.broadinstitute.org/gatk/best-practices/bp_3step.php?case=germshortwgs).

In an effort to manage the complexity of variant calling pipelines & experimental design, some pipelines developers have sought to author best practices documentation based on a set of specified use cases for their pipeline software. Unfortunately, no such 'best practices' appear to adequately cover the many considerations involved in an *in silico* analysis (let alone the *in vitro* considerations they are often dependent on) and thus may fail to precipitate their intended generalizability. Indeed, it is easy to read available best practice documents and imagine two separate researchers implementing two very different pipelines given the same set of instructions. Differences in sample acquisition, sample processing, library preparation, sequencing platform, computational resources, software availability, and parameterization are liable to lead to numerous inconsistencies due to differences. Much of these decisions may be traced back to the, often unaccounted for, granular decisions made throughout the course of the generation and analysis sequencing data – with a potentially catastrophic effect for reproducibility.

A good example of a 'best practice' documentation that may instill a poorly founded sense of experimental validity is the June 2016 best practice guidelines published for the GATK on the application of whole-genome sequencing (WGS) to the discovery of germline single nucleotide polymorphisms (SNPs) and indels [38-39, 67]. While the documentation is ostensibly well-defined and subject to regular revisions, an attentive reading reveals omissions and a lack of clarity that makes the use of the term 'best practices' optimistic. Although the visual guidelines reveal the basic procedures required, they lack a clear articulation of the actual steps in the analytics pipeline (Figure 5.2.1). Furthermore, these same GATK best practices state, under the heading 'What is *not* Best Practices?', that "the canonical Best Practices (as run in production at the Broad) are…optimized for the instrumentation (overwhelmingly Illumina) and needs of the Broad Institute sequencing facility" and that "they can be adapted…however, any workflow that has been significantly adapted or customized, whether for performance reasons or to fit a use case that (is) not explicitly covered, should not be called 'GATK Best Practices'" [55]. Thus, it must be asked, is it possible to reproduce the Broad Institutes' so-called 'best practices' at all? In this case, perhaps rather than claiming 'best practices', the Broad should consider renaming their 'best practice' workflows 'this is how we do it'[3]

Clearly stated, absent a full enumeration of the processes required to replicate any scientific best practice, the reproduction of any analysis or experiment:

- May lean inappropriately on a researchers' best judgment and expertise.

[3] Perhaps the Broad Institutes' naming decision was influenced by an aversion to copyright infringement from Montell Jordan on the grounds of his 90's hit single.

- May yield a compounding effect, whereby researchers lack a valuable tool for the quality documentation of their own work.

And, further, that:

- A failure in documenting best practices and/or a lack of transparency in best practices may expand the potential for researchers to unconsciously insert individual biases.
- Poor quality best practices may yield significant time and resource inefficiencies, resulting in a significant opportunity cost to researchers attempting to reproduce them.

That is not to say best practices and quality documentation are a silver bullet to many-a-researchers' woes, but rather that quality best practices provide an objective, widely available substrate upon which other researchers may construct experiments, pipelines, and software rationally and with full clarity.

5.3 Recommendations for Bioinformatics Software Development & Distribution

Throughout the course of this research numerous software limitations were encountered, many of which are common across scientific computing. Examples of these limitations included resource constraints, particularly relating to the memory usage of various algorithms and/or their implementations, to incompatible dependencies, poor documentation, a lack of graceful degradation, and poor adherence to standard error management protocols. These however represented relatively minor limitations however.

In contrast, much of the software downloaded and deployed for this study did not follow contemporary standards for software development. Comparative studies such as this are made challenging by heterogeneous software development and distribution practices. Fortunately, solutions for managing the distribution and deployment of software are widely available for free, mature, and offer significant utility for tracking the provenance of code and contributions by multiple authors. The following recommendations are designed to improve the reproducibility of analyses and assure proper tracking of software assets.

- Software should be maintained in a Git protocol based repository, such as Github, Gitlab, or Bitbucket
- Continuous integration and unit testing of builds should be managed via an automated service, such as Travis-CI, Bamboo, or Jenkins.

- If possible, software should also be made available via a Docker image. Alternatively, validated build scripts should be distributed to the appropriate package manager(s) (e.g. Homebrew, Linuxbrew, pip, Conda, npm). This significantly facilitates reproducibility.

- If possible, documentation should be managed through a language specific documentation framework and versioned with Read the Docs.

Executed correctly, these four points enable software to be widely distributed in a fashion that is compatible with modern systems. Software is less prone to bugs, and users of software are able to create, track, and resolve any bugs that do arise. Reproducibility can be significantly enhanced via the appropriate packaging of software and its dependencies. Documentation is prioritized and made widely available. Combined, these offer huge benefits to both the author and end-user of a software.

6. Limitations & Future Directions

6.1 Synthetic Data Intended Use & Limitations

The ICGC-TCGA-DREAM Somatic Mutation Calling Challenge data, used for the development of the pipeline module evaluation prototype and analysis, is subject to a number of limitations and caveats.

First and foremost, the data used for this study was selected for the pipeline development with quick accessibility being a primary consideration. The data is WGS with a coverage of 30x, which limits the ability of any variant calling method to confidently detect variants, particularly low frequency variants. If we consider a base call to be a hypothesis test (where h_0 assumes that a variant does not exist and the alternate h_1 that a variant does exist), we can consider depth of coverage at the base position to be analogous to sample size; depth of coverage is essential in determining the power of the test to detect a putative variant at the given position at or above a predefined minor allele frequency (MAF). Given the structure of the variant calling problem and the aforementioned challenges presented by somatic samples (tumor impurity, clonality, and CNVs), sequencing data collected with an ultra-deep NGS (defined as greater than 1000x (targeted) coverage [68]) approach is highly preferable for somatic variant calling in the clinical setting. There are several reasons for this: 1. When combined with databases of known variants, such as dbSNP [45], COSMIC [63-64], and ClinVar [69], the ultra deep sequencing approach is sensitive even without a normal pair, particularly for known variants 2. Ultra deep sequencing is also capable of revealing the complex genomic architecture of a cancer that may be particularly relevant in the clinical context [70] and 3. Variants are only

actionable if a precision therapy is available; this reduces the genomic search space making targeted panels more cost-effective, however, this can also limit the ability to discover and build evidence for novel variants. Thus, while ultra-deep NGS approaches are largely fit for purpose in the clinical setting, there are significant differences between somatic variant calling with ultra-deep NGS without a normal pair versus low coverage WGS with a normal pair, differences which may challenge the translation of the present iteration of this research into the clinical diagnostic setting.

A second limitation that is less apparent arises from the fact that the data used was synthetic, having been generated from the BAMSurgeon software [61]. BAMSurgeon outputs an aligned 'tumor' BAM file containing simulated mutations along with a truth VCF, using a provided BAM file, which represents the 'normal' condition. While BAMSurgeon does ostensibly reproduce genomic architectures and mutations similar to those observed in cancer, the degree to which these simulated architectures are truly representative of the complex biology of cancer samples, as seen in the clinical context, is unclear. For example, a documented limitation of the software is a tendency towards false-positive mutations in structural variant (SV) regions. A workaround suggested by the authors is to generate separate BAM files to test for different types of mutations [66] – an analytical scheme that may discount any covariation that may occur between SV regions and any *cis* single nucleotide variants (SNVs) or vice versa. In consideration of this, a larger question precipitates: Given a proposed analysis, what level of granularity is needed to computationally model sequencing data from a tumor sample such that it is practically representative of a truly biological counterpart? Further, how can we measure the degree to which simulated data represents a true biologically derived sample given a specific use case? These questions have yet to be answered and consequently precipitate a reduction in the confidence of any synthetic data analysis. While a counterpoint may be that even a true biological gold standard may be weakened by reductions in confidence, a biological gold standard can at least be made to closely mimic clinical data being generated at a specific site. For this reason, this research stands to benefit significantly from the analysis of background error models using such a biological gold standard.

6.2 Ultra Deep Sequencing Fixed Dilution Series

A future direction currently being pursued is the analysis of a biological gold standard sequenced with an ultra deep sequencing methodology. In fact, at the time of writing, the gold standard has already been generated via the serial dilution of DNA from an AML pre-treatment diagnostic assay sample into a 35-day post-treatment sample from the same patient. The dilutions were scaled at 10%, 3%, 1%, 0.3%, 0.1%, 0.01% and 0.0% MAF, resulting in the

experimental allele frequencies described by Table 6.2.1. This experiment will enable an improved examination of the research questions explored here by alleviating some of the limitations arising in the analysis of the WGS synthetic data.

Table 6.2.1 - Ultra Deep NGS Gold Standard Experimental VAFs by Variant

	IDH			NRAS			NPM1		
target % VAF of sample	var reads	ref reads	% VAF	var reads	ref reads	% VAF	var reads	ref reads	% VAF
10.00	159	1288	10.9882	155	994	13.4899	99	637	13.4510
3.00	86	2576	3.2306	84	2305	3.5161	76	1515	4.7768
1.00	26	1595	1.6039	15	1143	1.2953	19	849	2.1889
0.30	6	1138	0.5244	2	985	0.2026	8	613	1.2882
0.10	5	1056	0.4712	0	806	0	7	535	1.2915
0.01	2	1373	0.1454	0	871	0	4	692	0.5747
0.00	0 to 1	1108	0.0901	0	781	0	3	468	0.6369

7. Reference

1. Hourigan CS & Karp JE. Minimal residual disease in acute myeloid leukemia. *Nature Reviews Clinical Oncology*. 2013;10(8):460-471. doi:10.1038/nrclinonc.2013.100.

2. Short NJ & Jabbour E. Minimal residual disease in acute lymphoblastic leukemia: How to Recognize and Treat It. *Current Oncology Reports.* 2017;19:6. doi:10.1007/s11912-017-0565-x

3. Hills RK, Ivey A, Grimwade D, et al. Assessment of minimal residual disease in standard-risk AML. *New England Journal of Medicine*. 2016; 374:422-433. doi:10.1056/NEJMoa1507471

4. Lai C, Karp JE, & Hourigan CS. Precision medicine for acute myeloid leukemia. *Expert Review of Hematology.* 2016;(9):1-3.

5. Stock W & Estrov Z. Clinical use of minimal residual disease detection in acute lymphoblastic leukemia. In UpToDate, Larson RS (Ed.) [Internet]. UpToDate*,* Waltham, MA. (Accessed on February 2, 2016.)

6. Su X, Zhang L, Zhang J, Meric-Bernstam F, Weinstein JN. PurityEst: estimating purity of human tumor samples using next generation sequencing data. *Bioinformatics*. 2012;28(17): 2265-2266. doi:10.1093/bioinformatics/bts365.

7. Bergmann EA, Chen B-J, Arora K, et al. Conpair: concordance and contamination estimator for matched tumor–normal pairs. *Bioinformatics*. 2016;32(20):3196-3198. doi:10.1093/bioinformatics/btw389.

8. Kim S, Jeong K, Bhutani K, et al. Virmid: accurate detection of somatic mutations with sample impurity inference. *Genome Biology*. 2013;14(8):R90. doi:10.1186/gb-2013-14-8-r90.

9. Yu G, Zhang B, Bova GS, Xu J, Shih I, Wang Y. BACOM: *in silico* detection of genomic deletion types and correction of normal cell contamination in copy number data. *Bioinformatics*. 2011;27(11):1473-1480. doi:10.1093/bioinformatics/btr183.

10. Sadanandam A, Lal A, Benz SC, et al. Genomic aberrations in normal tissue adjacent to *HER2*-amplified breast cancers: field cancerization or contaminating tumor cells? *Breast Cancer Research and Treatment*. 2012;136(3):693-703. doi:10.1007/s10549-012-2290-3.

11. Muller E. et al. OutLyzer: software for extracting low-allele-frequency tumor mutations from sequencing background noise in clinical practice. *Oncotarget.* 2016;7(48):79485-79493. doi: 10.18632/oncotarget.13103

12. Aird D, Ross MG, Chen W-S, et al. Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome Biology*. 2011;12(2):R18. doi:10.1186/gb-2011-12-2-r18.

13. Benjamini Y, Speed TP. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Research*. 2012;40(10):e72. doi:10.1093/nar/gks001.

14. Ross MG, Russ C, Costello M, et al. Characterizing and measuring bias in sequence data. *Genome Biology*. 2013;14(5):R51. doi:10.1186/gb-2013-14-5-r51.

15. Laehnemann D, Borkhardt A, McHardy AC. Denoising DNA deep sequencing data—high-throughput sequencing errors and their correction. *Briefings in Bioinformatics*. 2016;17(1): 154-179. doi:10.1093/bib/bbv029.

16. Astrovskaya I, Tork B, Mangul S, et al. Inferring viral quasispecies spectra from 454 pyrosequencing reads. *BMC Bioinformatics*. 2011;12(Suppl 6):S1. doi: 10.1186/1471-2105-12-S6-S1.

17. Le H-S, Schulz MH, McCauley BM, Hinman VF, Bar-Joseph Z. Probabilistic error correction for RNA sequencing. *Nucleic Acids Research*. 2013;41(10):e109. doi:10.1093/nar/gkt215.

18. Bragg L, Stone G, Imelfort M, Hugenholtz P, Tyson GW. Fast, accurate error-correction of amplicon pyrosequences using Acacia. *Nature Methods*. 2012;9:425–426. doi: 10.1038/nmeth.1990.

19. Sameith K, Roscito JG, Hiller M. Iterative error correction of long sequencing reads maximizes accuracy and improves contig assembly. *Briefings in Bioinformatics*. 2017;18(1): 1-8. doi:10.1093/bib/bbw003.

20. Zagordi O, Bhattacharya A, Eriksson N, Beerenwinkel N. ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinformatics*. 2011;12:119. doi:10.1186/1471-2105-12-119.

21. Goodwin S, Gurtowski J, Ethe-Sayers S, Deshpande P, Schatz MC, McCombie WR. Oxford Nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome Research*. 2015;25(11):1750-1756. doi:10.1101/gr.191395.115.

22. Koren S, Schatz MC, Walenz BP, et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*. 2012;30(7):693-700. doi:10.1038/nbt.2280.

23. Hackl T, Hedrich R, Schultz J, Förster F. proovread: large-scale high-accuracy PacBio correction through iterative short read consensus. *Bioinformatics*. 2014;30(21):3004-3011. doi:10.1093/bioinformatics/btu392.

24. Salmela L, Rivals E. LoRDEC: accurate and efficient long read error correction. *Bioinformatics*. 2014;30(24):3506-3514. doi:10.1093/bioinformatics/btu538.

25. Lee H, Gurtowski J, Yoo S, Marcus S, McCombie WR, Schatz M. Error correction and assembly complexity of single molecule sequencing reads. *BioRxiv*. 2014. doi: 10.1101/006395

26. Medvedev P, Scott E, Kakaradov B, Pevzner P. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics*. 2011;27(13):i137-i141. doi: 10.1093/bioinformatics/btr208.

27. Nikolenko SI, Korobeynikov AI, Alekseyev MA. BayesHammer: Bayesian clustering for error correction in single-cell sequencing. *BMC Genomics*. 2013;14(Suppl 1):S7. doi: 10.1186/1471-2164-14-S1-S7.

28. Marçais G, Yorke JA, Zimin A. QuorUM: An error corrector for Illumina reads. Gibas C, ed. *PLoS ONE*. 2015;10(6):e0130821. doi:10.1371/journal.pone.0130821.

29. Li H. Exploring single-sample SNP and INDEL calling with whole-genome *de novo* assembly. *Bioinformatics* 2012;28:1838-44. doi: 10.1093/bioinformatics/bts280.

30. Butler J, MacCallum I, Kleber M, et al. ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Research*. 2008;18:810-20. doi: 10.1101/gr.7337908.

31. Yang X, Dorman KS, Aluru S. Reptile: representative tiling for short read error correction. *Bioinformatics*. 2010;26:2526-33. doi: 10.1093/bioinformatics/btq468.

32. Lim E-C, Mu¨ ller J, Hagmann J, et al. Trowel: a fast and accurate error correction module for Illumina sequencing reads. *Bioinformatics*. 2014;30:3264-5. doi: 10.1093/bioinformatics/btu513.

33. Wirawan A, Harris RS, Liu Y, Schmidt B, Schröder J. HECTOR: a parallel multistage homopolymer spectrum based error corrector for 454 sequencing data. *BMC Bioinformatics*. 2014;15:131. doi:10.1186/1471-2105-15-131.

34. Quince C, Lanzen A, Davenport RJ, Turnbaugh PJ. Removing noise from pyrosequenced amplicons. *BMC Bioinformatics*. 2011;12:38. doi:10.1186/1471-2105-12-38.

35. Reeder J, Knight R. Rapidly denoising pyrosequencing amplicon reads by exploiting rank-abundance distributions. *Nature Methods*. 2010;7:668-9. doi:10.1038/nmeth0910-668b

36. Quince C, Lanze´n A, Curtis TP, et al. Accurate determination of microbial diversity from 454 pyrosequencing data. Nature Methods 2009;6:639-41. doi:10.1038/nmeth.1361.

37. Mysara M, Leys N, Raes J, Monsieurs P. NoDe: a fast error-correction algorithm for pyrosequencing amplicon reads. *BMC Bioinformatics*. 2015;16(1):88. doi:10.1186/s12859-015-0520-5.

38. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, et al. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Research. 2010 Sep 1;20(9):1297–303.

39. DePristo MA, Banks E, Poplin RE, et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature genetics*. 2011;43(5):491-498. doi: 10.1038/ng.806.

40. The 1000 Genomes Project Consortium. A map of human genome variation from population scale sequencing. *Nature*. 2010;467(7319):1061-1073. doi:10.1038/nature09534.

41. The 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*. 2012;491(7422):56-65. doi:10.1038/nature11632.

42. The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*. 2015;526(7571):68-74. doi:10.1038/nature15393.

43. Cibulskis K, McKenna A, Fennell T, Banks E, DePristo M, Getz G. ContEst: estimating cross-contamination of human samples in next-generation sequencing data. *Bioinformatics*. 2011;27(18):2601-2602. doi:10.1093/bioinformatics/btr446.

44. Cibulskis K, Lawrence MS, Carter SL, et al. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature Biotechnology*. 2013;31(3):213-9.

45. Kitts A, Phan L, Ward M, et al. The database of short genetic variation (dbSNP) 2013 Jun 30 [Updated 2014 Apr 3]. In: The NCBI Handbook [Internet]. 2nd edition. Bethesda (MD): National Center for Biotechnology Information (US); 2013-.

46. Ni S, Stoneking M. Improvement in detection of minor alleles in next generation sequencing by base quality recalibration. *BMC Genomics*. 2016;17:139. doi:10.1186/s12864-016-2463-2.

47. Tian S, Yan H, Kalmbach M, Slager SL. Impact of post-alignment processing in variant discovery from whole exome data. *BMC Bioinformatics*. 2016;17:403. doi:10.1186/s12859-016-1279-z.

48. Cacho A, Smirnova E, Huzurbazar S, Cui X. A Comparison of base-calling algorithms for Illumina sequencing technology. *Briefings in Bioinformatics.* 2016;17(5):786-95. doi:10.1093/bib/bbv088.

49. Li H. BFC: correcting Illumina sequencing errors. *Bioinformatics*. 2015;31(17):2885-87. doi:10.1093/bioinformatics/btv290.

50.Drezen E, Rizk G, Chikhi R, et al. GATB: Genome Assembly & Analysis Tool Box. *Bioinformatics*. 2014;30(20):2959-61. doi:10.1093/bioinformatics/btu406.

51. Benoit G, Lavenier D, Lemaitre C, Rizk G. Bloocoo, a memory efficient read corrector. Poster session presented at: *European Conference on Computational Biology*. 2014 Sept 7-10; Strasbourg, France.

52. Song L, Florea L, Langmead B. Lighter: fast and memory-efficient sequencing error correction without counting. *Genome Biology*. 2014;15(11):509.

53. Space/time trade-offs in hash coding with allowable errors. Bloom BH & Computer Usage Company, Newton Upper Falls, MA. *Communications of the ACM*. 1970;13(7):422-26. doi:10.1145/362686.362692

54. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*. 2001;98(17):9748-53. doi:10.1073/pnas.171285098.

55. Broad Institute. Best practices for somatic SNV and indel discovery in whole genome and exome sequence. February 22, 2017. Available at: https://software.broadinstitute.org/gatk/best-practices/mutect2.php, Accessed February 22, 2017.

56. Broad Instititute. Picard Tools. Available at http://broadinstitute.github.io/picard/. Accessed March 1, 2017.

57. Li H, Handsaker B, Wysoker A, et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*. 2009;25(16):2078-2079. doi:10.1093/bioinformatics/btp352.

58. Wilks C, Cline MS, Weiler E, et al. The Cancer Genomics Hub (CGHub): overcoming cancer through the power of torrential data. *Database: The Journal of Biological Databases and Curation*. 2014;2014:bau093. doi:10.1093/database/bau093.

59. Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*. 2009;25(14):1754-1760. doi:10.1093/bioinformatics/btp324.

60. Cock PJA, Antao T, Chang JT, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*. 2009;25(11):1422-1423. doi:10.1093/bioinformatics/btp163.

61. Ewing AD, Houlahan KE, Hu Y, et al. Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection. *Nature methods*. 2015;12(7):623-630. doi:10.1038/nmeth.3407.

62. Kandoth C, McLellan MD, Vandin F, et al. Mutational landscape and significance across 12 major cancer types. *Nature.* 2013;(502)333-339. doi:10.1038/nature12634

63. Forbes SA, Bhamra G, Bamford S, et al. The Catalogue of Somatic Mutations in Cancer (COSMIC). *Current protocols in human genetics / editorial board, Jonathan L Haines . [et al]*. 2008;CHAPTER:Unit-10.11. doi:10.1002/0471142905.hg1011s57.

64. Bamford S, Dawson E, Forbes S, et al. The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website. *British Journal of Cancer.* 2004;91:355-358. doi:10.1038/sj.bjc.6601894.

65. Danecek P, Auton A, Abecasis G, et al. The variant call format and VCFtools. *Bioinformatics*. 2011;27(15):2156-2158. doi:10.1093/bioinformatics/btr330.

66. Ewing A. BAMSurgeon: Methods for spike-in mutations of BAM files. February 18, 2016. Available at: https://github.com/adamewing/bamsurgeon/blob/master/doc/Manual.pdf, Accessed February 22, 2017.

67. Broad Institute. Best practices for germline SNP and indel discovery in whole genome and exome sequence. February 22, 2017. Available at: https://software.broadinstitute.org/gatk/best-practices/bp_3step.php?case=GermShortWGS, Accessed February 22, 2017.

68. Sims D, Sudbery I, Ilott NE, et al. Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics*. 2014;15:121-132. doi:10.1038/nrg3642.

69. Landrum MJ, Lee JM, Riley GR, et al. ClinVar: public archive of relationships among sequence variation and human phenotype. *Nucleic Acids Research*. 2014;42(Database issue):D980-D985. doi:10.1093/nar/gkt1113.

70. Prandi D, Baca SC, Romanel A, et al. Unraveling the clonal hierarchy of somatic genomic aberrations. *Genome Biology*. 2014;15(8):439. doi:10.1186/s13059-014-0439-6.

8. Supplemental

8.1 Interleaved Paired-End Shuffling Algorithm

A simple script to shuffle interleaved paired-end FASTQ files was developed for use to realignment with BWA-mem. The shuffling algorithm first reads in chunks of *n* pairs, shuffling these chunks using the default random.shuffle() Python implementation, it then writes the chunks to a temporary directory. Each chunk is counted, with each read group FASTQ generating between 600 - 1300 chunk FASTQ files with 50,000 pairs (the default *n* parameter, effectively 100,000 reads), given the synthetic data. Next, the algorithm reassembles the FASTQ by randomly writing the shuffled FASTQ chunks to the new shuffled FASTQ file. The Python script is available via the source code repository (URL: https://github.com/greenstick/thesis-pipeline/blob/master/utils/shuffle-fastq.py). The current implementation runs in a single thread and lacks parallelization, however, this may be updated in the future.

8.2 Evaluated Background Error Models

See attached.

9. Addendum

Immediately prior to the original oral defense of this thesis it was found that some of the AUC values computed for each pipeline condition may have been artificially buoyed by the introduction of significant numbers of false positive corrections by the background error models that were subsequently classified as errors by the MuTect 2 variant caller (this is particularly noticeable in figures 4.1.4 and 4.3.1). While it is not believed that this affected the performance comparison of the conditions in relation to their ability to detect variants, this is nonetheless an area of of ongoing research for this projects. It is believed that the tracing of these false positive corrections along with the appropriate parameterization of the background error models will greatly mitigate or eliminate this effect and allow for better confidence in the quality of the comparisons made in this thesis.