DNA Coverage Prediction Using Aggregated Poisson Approximation

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Citable link</td>
<td><a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:42004048">http://nrs.harvard.edu/urn-3:HUL.InstRepos:42004048</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA">http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA</a></td>
</tr>
</tbody>
</table>
DNA Coverage Prediction Using Aggregated Poisson Approximation

Khalid Shakir

A Thesis in the Field of Biotechnology
for the Degree of Master of Liberal Arts in Extension Studies

Harvard University
November 2018
Abstract

DNA whole genome sequence analysis is an important process, but is timely and expensive. In past studies, experts have often theorized that once aligned to a reference genome, short reads of DNA sequence would cover a genome reference in a Poisson based distribution. Under this theory increasing the DNA sequencing depth across the genome should cover all bases with a minimum number of reads once the mean depth reaches a certain coverage threshold. However, since the completion of the human genome reference, there are clearly coverage disparities when examining the distributions of short reads. When a sample is mapped to the reference, the coverage distribution does not fit a Poisson, where among other things the variance in coverage should equal the mean.

Here a new tool is described that can better predict from a fraction of the eventual reads the coverage of a full sample. The degree of coverage for a full sample may be predicted based on the coverage analysis of a sequencing run with just a subset of the sample. The subset of read coverage is divided by metadata, for example read covariates, reference covariates, etc. Each division of read coverage is put into a bin, where each bin has an average coverage. The amount of desired coverage in the full sample is used to scale the average coverages using Poisson approximations of coverage for the desired full sample. Compared to a single poisson, when aggregated these approximations of stratified coverage are better able to predict the distribution of coverage for regions of the genome with high or low coverage.

The tool uses a number of stratification techniques. Disparate methods of stratification were evaluated during the development of the tool, some of which included: separating reads based on their characteristics, separating regions of the genome based on metrics generated from
the reference used for aligning the short reads, and the use of various metadata files containing summary information about the reference and prior human genomic analyses. The use of this tool in a processing pipeline, will enable data analysts and investigators to better estimate how additional sequencing would affect the coverage distributions in sequencing samples prepared with and without PCR amplification. This enhancement in workflow automation is expected to save both time and money for whole genome projects looking to reach a certain threshold of coverage for samples.
# Table Of Contents

List of Tables........................................................................................................vi

List Of Figures........................................................................................................vii

I. Introduction.............................................................................................................1

   History of Reference Based Sequencing ..............................................................1

   Existing Informatics for Sequence Data Analysis .............................................6

   Poisson Approximations of DNA Coverage.......................................................14

II. Materials and Methods.......................................................................................17

III. Results...............................................................................................................24

IV. Discussion..........................................................................................................38

Appendix..................................................................................................................42

   Tool Development Notes...................................................................................42

   Reference Panel..................................................................................................44

   Read Based Stratification Negative Results ......................................................45

   Additional Figures...............................................................................................48

References ...............................................................................................................52
List of Tables

Table 1. Reference Panel Low Coverage Predictions .................................................. 29
Table 2. GC Low Coverage Predictions ........................................................................ 31
Table 3. Combined Low Coverage Predictions .............................................................. 33
Table 4. PCR+ Low Coverage Predictions .................................................................... 35
Table 5. High Coverage Predictions ............................................................................ 36
Table 6. Reference Panel Samples ............................................................................... 44
List Of Figures

Figure 1. PCR- Poisson Prediction From a Single Mean.........................................................16
Figure 2. PCR- Mappability Stratification Dispersions and Predictions ...................................26
Figure 3. PCR- Reference Panel Stratification Dispersions and Predictions...............................28
Figure 4. PCR- GC Stratification Dispersions and Predictions................................................30
Figure 5. PCR- Combined Stratification Dispersions and Predictions........................................32
Figure 6. PCR+ Combined Stratification Dispersions and Predictions......................................34
Figure 7. Summary of Prediction Differences..............................................................................37
Figure 8. PCR- TLEN Stratification Dispersions and Predictions................................................47
Figure 9. PCR+ TLEN Stratification Dispersions and Predictions.................................................48
Figure 10. PCR+ Mappability Stratification Dispersions and Predictions....................................49
Figure 11. PCR+ Reference Panel Stratification Dispersions and Predictions..............................50
Figure 12. PCR+ GC Stratification Dispersions and Predictions................................................51
Chapter I
Introduction

History of Reference Based Sequencing

Sequencing technology furthers study of the central dogma, specifically allowing researchers to better understand the relationship between genotypes and phenotypes (G.-W. Li & Xie, 2011). Today, sequencing is used across many cases of human research, including investigations of somatic variation in common cancers (Ashworth & Hudson, 2013). For individual treatment sequencing is often used during clinical diagnosis (Feero, Guttmacher, & Collins, 2010). Widespread studies of genetic diversity in populations use sequencing as well, such as the recently completed 1000 Genomes Project (1000 Genomes Project Consortium et al., 2015).

Over the course of the 1990s a reference was created to provide a complete mapping of the human genome (Chial, 2008), with the understanding that genetic variation creates deviations in the coding and expression of proteins. The simplest cases of genetic variation from the reference are classified as single nucleotide polymorphisms (SNPs) with small insertions and deletions (INDELs) (Sherry et al., 2001). Larger structural variations may also appear within genomes, including copy number variation or translocations, where whole subsections of DNA may be rearranged (Lappalainen et al., 2013). However to complete an accurate analysis of samples, care must be taken to insure that sequencing depths across the genome are sufficient for investigative and variant sensitivity (Sims, Sudbery, Ilott, Heger, & Ponting, 2014).
The financial cost of sequencing has continued to decrease over the past couple of decades as new technologies have emerged (Check Hayden, 2014). Often genotyping individuals may be assayed using microarray technology, screening at known loci of common genetic variation. Arrays offer a significantly reduced cost relative to DNA sequencing. Simultaneously they introduce an important tradeoff by not necessarily detecting novel and low frequency variation (Ha, Freytag, & Bickeboeller, 2014). Similarly, projects such as HapMap (International HapMap Consortium, 2003) mapped common variations across human subpopulations. The HapMap project also established a panel of open consent samples that can be used freely in a wide variety of open research fields. At the time of the project’s completion the resulting catalog of genetic variation reduced the computational and financial burden for population studies. Instead of fully sequencing all of the individual members of a study, data scientists were able to extrapolate variant frequencies common to subsets of the human population.

More recently over the past decade, large-scale sequencing centers allow mapping genotypic variance on an individual allelic basis per sample using Next Generation Sequencing (NGS) or High Throughput Sequencing (HTS) technology, (Glenn, 2011). This technology uses short overlapping reads of DNA to provide a statistical review of variation in individual genomes, enabling novel discoveries and innovation. While the process has improved, so have the toolsets available for characterization and downstream analysis. As sequencing technology continues to evolve it is also possible to sequence and genotype data at smaller and smaller laboratories, without the liquid handling and computing requirements of a full sized sequencing center (Loman et al., 2012).

As in most industries, the race continues to obtain results faster, in some cases cheaper, while increasing widespread availability of best practice analysis. Clinical labs often diagnose
patients with sequencing. The clinicians attempt to determine potential variations of existing germline genetic disorders for individuals in a family, or treating somatic variations that may appear throughout one’s normal lifespan (Rizzo & Buck, 2012). Personal genomics and precision medicine provide the opportunity to characterize new targeted treatments. These customized therapies significantly benefit individuals with certain genetic variants, highlighting statistically significant results that may not be relevant when examining variations common to a wider cohort (Collins & Varmus, 2015).

Whole genome sequencing (WGS) enriches the potential discovery of variations within the full genome of an individual. However WGS involves reading significant amounts of DNA, and is computationally intensive to analyze relative to whole exome sequencing (WES) (Belkadi et al., 2015; Sun et al., 2015). With WES only the coding regions of DNA are selected for analysis using capture techniques that preferentially select exonic locations of DNA for further processing and sequencing. When the chromosomes of the human reference are laid out in series, the whole genome is approximately three billion bases long. In contrast whole exome sequencing frequently targets thirty to forty megabases (Asan et al., 2011). WES is significantly less expensive to sequence and analyze per sample, compared to whole genome sequencing, for both computation and storage when sequencing at similar coverage. Yet the practice of excluding a significant portion of the human genome from analysis must be considered before undertaking a WES versus WGS project.

Individual samples may be analyzed using several techniques including, shallow sequencing, targeting overlapping coverage of two to four reads per reference locus, or deep sequencing which targets coverage of twenty, thirty, or more reads per reference locus. Deeper sequencing provides higher evidence for each variant detected. However, because the whole
human whole genome is orders of magnitudes longer that the human whole exome, when aiming for a fixed cost per sample, studies will sometimes choose between protocols involving shallow whole genomes or deep whole exome sequencing.

To produce this increased depth one can start with additional DNA from the sample, or use more polymer chain reaction (PCR) cycles to generate more copies of each fragment for analysis. PCR amplification introduces other replication artifacts, due to the efficiency of the process, with the frequency of artifacts depending on the composition of the DNA being sequenced (Aird et al., 2011).

Sequencing of DNA reads may be commonly implemented as single-end reading of a fragment, or paired-end. In paired-end sequencing, a single fragment of DNA is sequenced from both ends, producing a pair of reads that may, or may not overlap depending on the selected size of the original fragment during library preparation (Zhang, Kobert, Flouri, & Stamatakis, 2014). By preparing the DNA libraries and shearing into larger DNA fragments, downstream analysis may be used to genotype larger insertions, deletions, or other structural variation. Larger structural information may also be inferred from longer reads, producing haplotypes important to human and other non-monoploid analyses (Sudmant et al., 2015).

After reading the fragments of DNA in a sequencer, the billions of reads must then be aligned to the genome. Lander and Waterman (1988) previously described a method of estimating random distributions in the DNA sequencing process, including Poisson approximations related to the start position of aligned reads to genome locations. Using Poisson estimates during genomic analysis has also been investigated by a number of groups over the years, some aiming to creating mathematical models to predict the amount of sequencing required to adequately fill in gaps of coverage (Roach, Thorsson, & Siegel, 2000). A Poisson
model is often used to estimate distributions of time based events. Because the aligner maps each read, single or paired, to a location on the genome, the process might have effectively been modeled as Poisson distribution. The chromosomes of the genome laid out in series essentially may be viewed as a timeline, with the read alignments overlapping some series of loci modeled as our events. Assuming then that the distribution of reads does follow a Poisson process, the probability that N reads overlap at the same loci may be calculated from our expected value of coverage. Here a tool is described that expands on the idea of modeling coverage using Poisson distributions. Instead of using a single Poisson it first produces multiple smaller distributions and then combines their individual predictions into an aggregated expectation of additional sequencing coverage.

When aligning short reads of DNA to the reference, variants such as SNPs and INDELs in the DNA will affect the exactness of the process, along with other covariates. Repeats such as those in known regions of short tandem repeats of bases are commonly known to make it difficult to uniquely identify where exactly a read should be mapped to DNA (Bornman et al., 2012). Additionally the centromeres are not well represented in recent versions of the human genome reference, although this continues to be improved with each revision (W. Li & Freudenberg, 2014). New sequencing approaches continue to come to market with optimizations such as longer reads. Many of the new techniques increase the ability to address fidelity issues in pericentromeric regions. But for the increased read length most of the technologies incur a tradeoff of increased sequence error rates or cost (Rhoads & Au, 2015; Sevim, Bashir, Chin, & Miga, 2016).

Another covariate during the sequencing process is the GC bias of the short DNA being read. Areas of the genome with skewed GC to AT ratios introduce various issues that affect the
sequencing and analysis process. Example artifacts in sequencing preparation include changes in frequency of PCR amplification and variable DNA fragment size selection (Benjamini & Speed, 2012; Chen, Liu, Yu, Chiang, & Hwang, 2013).

Existing Informatics for Sequence Data Analysis

To track and measure the various covariates with DNA samples there are many bioinformatics tools available. Several well developed libraries written in a number of programming languages enable developers to read and traverse DNA sequences stored in various file formats. When building a novel tool, a developer’s selection of a programming language restricts the libraries available for use. Assorted libraries have utility methods that may save time during development while increasing the runtime of the application. Meanwhile choosing an application that is extremely fast may require advanced knowledge not only to code but to diagnose error conditions. The ideal tool should enable a bioinformatician to quickly produce results in a reproducible and understandable manner, especially for the validation of results.

A common language used in bioinformatics is the statically typed Java programming language (Gosling, Joy, Steele, & Bracha, 2000). In a statically typed language, once a variable is defined it cannot change its type. This feature eliminates cases where two functions within a program may mistakenly use different types for the same variable. Java has the added advantage of being a compiled language. This means that programs written in Java will be turned into a more machine efficient intermediate form before being run by the computer. This compilation step enables the code to execute faster, and also serves to catch early programming errors such as wrong variable names, types, or other incorrect language syntax.
The computers running programs contain a limited amount of memory for storing values. The calculation of the DNA reads will need to read and analyze billions of bases, which is more than most computers can hold within RAM. Thus cleanup of unused variables will be important to ensure the program does not cause the computer to run out of shared memory. Another benefit of using Java is that the Java Virtual Machine (JVM) running the compiled Java code manages the necessary cleanup by keeping track of variables no longer in use, and releasing them from memory using an automatic garbage collector. By using Java for building the algorithm, the developed prediction tool is able to easily stay within normal bounds of memory provided by common computers, often on the order of four to eight gigabytes per process.

Java also has the benefit of having a number of tools to help with development. These programs include Eclipse (“Eclipse desktop & web IDEs | The Eclipse Foundation,” n.d.) and IntelliJ IDEA (“IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains,” n.d.). While writing Java programs, these integrated development environments provide various helpful utilities to developers, including colored syntax highlighting of the code, debugging tools for stepping through code line by line to look for errors, and often hints to make the program more readable for future developers.

A large advantage of Java is that once a program has been compiled into the machine friendly Java bytecode, anyone who wants to run the program will not need to recompile the original source code for their machine, assuming they can obtain the widely available JVM for their platform (“OpenJDK,” n.d.). Currently, compiled Java programs may be run on a wide number of platforms including Windows, Macintosh, and Linux. Thus the program may be developed on any common desktop environment, and later executed on any other supported environment without any expected compatibility issues.
There are several languages that a developer may author programs in, and then compile down into Java bytecode. Another popular language is Scala (Odersky et al., 2004), used by a number of newer tools that are part of DNA analysis toolkits (bioscala, n.d.). Scala has the advantage of using a terse language that allows the developer to express ideas in fewer characters, and the ability to have variables be implicitly passed between methods instead of requiring the developer to specify them on each call. However this code density, and non-explicitness, may make the code harder to understand for inexperienced developers, or developers who are unfamiliar with Scala. Other JVM based languages such as Groovy (Koenig, Glover, King, Laforge, & Skeet, 2007) or Kotlin (“Kotlin Programming Language,” n.d.) are described as easier for Java developers to understand but have not been widely used within my current work environment for DNA processing applications.

A language that is widely used for DNA analysis, and programming in general, is Python. Python is considered very easy to read and write, and used in wide array of bioinformatics programs and libraries including Biopython, pandas, and Jupyter (“Biopython · Biopython,” n.d.; McKinney, 2015; Perez, n.d.). With a relatively liberal software license compared to Java, Python is easily installed and is even installed by default on a number of platforms. However, Python interprets scripts at runtime, and uses dynamic typing. It generally runs slower than the same algorithms written in compiled languages. While Python is widely used, my software engineering cohort within the Broad Institute generally selects Java or Scala over Python for larger CPU and memory intensive programs.

Of all the languages C has the best runtime performance (Ritchie, 1993) or the expanded C++ (Stroustrup, 2000). As C/C++ do not contain automatic memory management, it is possible to write very fast and efficient programs that reuse data structures. While a C/C++ program may
create and reuses data structures, it is also the programmer's responsibility to ensure that the objects are removed from memory. Over a longer running program, the failure to release objects may exhaust the memory available on the computer, causing the tool to exit prematurely. Without careful programming, accessing the wrong memory location can lead to errors as well.

Compiled C/C++ code is very machine efficient, but the lack of an intermediary bytecode format, like Java uses, means that to run on multiple platforms such as Windows and Linux the source code must be recompiled for each operating system. This also applies to bioinformatics libraries such as htslib (samtools, n.d.), which must be distributed as platform specific libraries, or as source code that must be compiled before use.

As the C/C++ languages are highly efficient, and the programming language used to develop the operating systems such as Windows, Linux and macOS, C/C++ are widely available. However, cross-platform development environments for C/C++ are currently not as feature rich as Java development environments, Eclipse and IntelliJ IDEA. Without these rich environments, debuggers are not as integrated into development, making unexpected errors much harder to diagnose. By avoiding a C/C++ based development environment, other developers are able to more easily evaluate and enhance the coverage predictor in the future. The development environment used for writing and debugging can greatly assist the stability of the program, and reduce the amount of time needed to produce a working and accurate program. It also helps if others are able to easily build the software from source code, such that other developers may contribute to or extend the functionality in the future.

When writing source code for a compiled application, or any program that uses third party libraries, a build tool that organizes dependencies, compiles source code, and even runs
automated tests will help the wider developer community. One may write software for use without a build tool, but using one makes development easily reproducible and collaborative.

Options for Java build tools include Apache Maven, Gradle, and SBT (“Gradle Build Tool,” n.d., “sbt - The interactive build tool,” n.d.; Porter, van Zyl, & Lamy, n.d.). Apache Maven is one of the earliest build tools for Java, has a very wide distribution, and supports a library format that includes the ability to pull third party dependencies from online repositories. However, the biggest drawback to using Apache Maven as a build tool is that the project definition is written in XML (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 1997), which can be verbose and hard to read for developers. Gradle is another alternative, with build definitions written in the Groovy programming language instead of XML. Like Maven, Gradle build files are also importable into integrated development environments such as Eclipse and IntelliJ IDEA, allowing the environments to setup development and debug sessions with their enhanced features.

While Java libraries are distributed as binaries, the source code for programs is often shared by developers. A source control management system (SCM) provides revision control, storing a history of changes to the program. Git SCM (“Git,” n.d.) supports a distributed model of development, where multiple developers may work on various parts of software at the same time, and then optionally merge their results together. A popular place for storing source code and collaborative reviewing is the GitHub online source code repository (Build software better, together, n.d.). GitHub allows future developers to view Git source code history via a web browser instead of on the command line, and the web based software provides features that make it easy for larger developer teams to review submitted patches and changes. Git and GitHub do
have a learning curve for new developers, but they are widely used across many programming languages and software teams.

For the programs parsing and analyzing sequencing data the standard for storing the reference genome is the FASTA format (“BLAST - NCBI,” n.d., hts-specs, n.d.). This reference template contains an outline of human genome chromosomes as contiguous strings of DNA. Short reads of DNA from sequenced samples are mapped and aligned to the reference template by programs such as BWA (H. Li, 2013). The alignment tools store the short reads in one of two industry standard file formats. When the reads are stored as text the format is SAM, or Sequence Alignment/Map, and when the text is compressed into a binary file, the format is BAM, or Binary Alignment/Map (H. Li et al., 2009). Due to the number of reads, the uncompressed SAM format is not an effective file representation for whole genomes or exomes. Additionally, because an analysis may require investigating subsets of the compressed reads within a BAM file, the specification allows for an optional index to accompany the BAM file, or to be generated from the contents of the original file. This index then allows libraries to translate reference coordinates to file offsets for accessing reads at specific genomic locations.

The BAM file specification also enables files to store information on groups of reads. A single sample may be sequenced at different times, possibly sequenced even on different machines. In an aggregated BAM the Read Group (RG) tag is used to label reads that were sequenced at the same time from the same sample. This RG tag may be used by analysis tools to filter out or select only reads that belong to a certain read group.

Often for bioinformatic analyses the BAM and FASTA will be accompanied by other metadata stored in a variety of genomic formats. Common formats for this metadata are VCF and BED (Danecek et al., 2011; Quinlan & Hall, 2010). BED files can store values for each genomic
location, often in a compressed format called bedGraph. VCF allows storing any arbitrary information about genomic locations. VCFs are often much larger than BED files but enable developers to store information in headers that may be applicable for provenance and future analyses.

There are many Java toolkits for bioinformatics developers that are able to open, traverse, and write common file formats. Two popular choices are the Genome Analysis Toolkit (GATK) and the Picard suite of tools (DePristo et al., 2011; “Picard Tools - By Broad Institute,” n.d.). The GATK team has recently released a fourth major revision, often referred to a GATK4. It promises faster analyses of genomic data particularly in compute environments where the files may not be stored on the machine the program is executing on. The GATK4 also provides an ecosystem with a simplified programing interface for developers that uses in-memory multiprocessing APIs such as the Spark platform. Additionally, the GATK4 provides hooks into the R software package, often used for statistical analysis and visualization of datasets. The development of this tool was started on a prerelease of the GATK4 that was in early alpha status.

An alternative to the GATK4 are the Picard tools. Like the GATK4, Picard also uses the HTSJDK as a library for traversing reads and reference data, while providing utility methods for parsing command line arguments and opening files. The advantages of Picard are that the code is stable, and has been actively used and developed for several years, analyzing petabytes of genomic data for a wide number of developers in hundreds of customized pipelines around the world. Picard even provides existing metrics calculators including tools that measure GC bias across the genome.

Bioinformatics tools are often used to analyze relative large datasets or the order of petabytes, running pipelines for hours or days at a time. Job schedulers can submit the programs
to high-performance computing (HPC) or to cloud based servers (Hwang, Dongarra, & Fox, 2013), allowing the programs to parallelize the processing of the genomic data. Job workflows running on schedulers can divide tabulation and computation by genomic location, or other filtering techniques. Ultimately the workflows run a gathering or reduction step to consolidate the intermediate outputs into a final result. This division of computation provides the ability to produce a final calculation in a much shorter timespan by using more compute nodes and memory at the same time. And if any parallel run of the program fails, possibly due to a transient network issue, or a problematic compute node, the same job can be restarted on a new HPC node or cloud machine, only requiring waiting for that one portion of the analysis to run before continuing. HPC and cloud computing resources are the best environments for running these analyses, versus running on a personal computer. The large size of the data files, plus the amount of compute resources needed for the analysis mean that most organizations use dedicated environments for processing terabytes or petabytes of genomic data.

Even though a variety of compute environments may be used, in research it is important to document methods for provenance, and empower other scientists to reproduce the results. As part of maintaining provenance, there are tools available that encapsulate the environments required to execute a program, specifying the underlying operating system and specific versions of programs used to run the analysis. While full virtual machine images have been used in the past, in recent years Linux containers have become more a popular way of bundling the specific operating system, executables, and scripts (Chamberlain & Schommer, 2014).

While there are many options for managing workflows (Di Tommaso, n.d.) the Workflow Definition Language (WDL) is quickly becoming an option from one-off analyses to large cohort datasets (“gnomAD browser,” n.d., WDL, n.d.; Lek et al., 2016). WDL is co-developed by the
OpenWDL community and supported by a number of institutes, business, and organizations (WDL, n.d.). The program Cromwell (Broad Institute Data Sciences Platform, n.d.), to which the author has contributed development, is able to run a WDL script on a local laptop during early tool development, before moving to HPC and cloud environments for scaling purposes.

Poisson Approximations of DNA Coverage

Lander and Waterman et al. have theorized, studied, and evaluated how the alignment of short reads of DNA should follow a Poisson process (Lander & Waterman, 1988; Roach et al., 2000; Wendl & Wilson, 2008). The coverage of the reference by reads should approximate a Poisson distribution, where the mean depth of read coverage per location should also approximate the variance in coverage. This model suggests that coverage will increase in all regions at the rate of a poisson.

With a naive Poisson model for coverage, one could in theory sequence a small portion of the full sample and measure its coverage across the entire reference. If the coverage distribution was truly Poisson, then even if large parts of the genome were not covered, as number of reads increases more and more of the genome would be covered. A scale factor for approximating coverage distributions can be produced by taking the desired coverage and dividing it by the measured coverage of a partial sequencing experiment. Multiplying the average coverage ($\lambda$) by the scale factor will then give a new average coverage, that could then be used as a predictive average for the purposes of modeling a Poisson distribution of coverage for a fully sequenced whole genome sample. When looking at the results of the single Poisson prediction versus the actual coverage distribution of a deeply covered sample, it becomes clear that on average, there
exist regions of the genome that continue to lack adequate coverage for sensitive and accurate downstream analysis.

In the Poisson model for coverage at large enough depth the expectation is that all bases will be covered by some reads. Additionally the coverage distribution is expected to peak only around the average coverage, and that there should not be any locations measuring extremely high coverage. Examining the coverage distribution of a whole genome the average coverage peak may partially approximate a Poisson, but it is not an exact fit (figure 1). The Poisson approximation overestimates the number of genomic locations that will be covered at the average coverage amount. Just below and just above the average coverage the approximation underestimates the number of covered genomic locations. Additionally in zero to low coverage areas the approximation does not capture the counts at all. While not as extreme as the low coverage approximation difference, there are also small amounts of high coverage locations not counted either.
Figure 1. PCR- Poisson Prediction From a Single Mean.
By taking the average depth of coverage from a read group and scaling it by some desired coverage factor the Poisson approximation begins to model the actual coverage. However there are bases in the fully sequenced sample that are still uncovered. Also loci with average coverage are overrepresented by this prediction model. Not shown are a small number of loci that have extremely high coverage in the actual sample, with read depths $\geq 250$. 
Chapter II

Materials and Methods

A new GATK4 tool, or walker, was created initially based on the Picard tool CollectWgsMetrics. The GATKWGSMetrics was first designed to produce exact coverage counts using GATK4 methods of loci traversal instead of using the Picard API. From there modifications were made to the code to account for minor traversal differences in the GATK4 toolkit. The original port of code base produced exact counts except for the fact that by default the GATK4 filters a very small number of paired end reads from the traversal engine. This discrepancy is due to the usage of a method “ReadUtils.isBaseInsideAdaptor”. The function examines the locations of a pair of aligned reads relative to the reference locus being traversed. If the locus is estimated to overlap an adaptor sequence on either read the GATK4 filters the read. This is distinct from the traversal behavior of the CollectWgsMetrics tool thus it produces a number of small coverage differences between the two tools.

The GATKWGSMetrics was then outfitted with the ability to generate tables using the GATKReports API. The walker also gained the ability to divide counts of coverage distributions by various metrics. Covariates used for stratification included attributes within the reads, attributes from an external file resource about each genomic loci, or using the bases of the reference genome. For the purposes of analysis the walker skipped any location with an N base, meaning that regions of the reference genome that have not been well characterized by the Genome Reference Consortium in the human reference version 38 (GRCh38 or hg38) were not included within the analysis. Similarly, the analysis was limited to the autosomes such that the coverage analysis could in theory be applied indeterminate of the number of sex chromosomes
across samples. Mitochondrial sequences were a not included due to the repetitiveness of the DNA and the vastly different average coverage.

In each case of generated coverage distributions based on various stratifications, a predictive model was formed using the relative contribution of each bin’s stratified coverage. A new GATK4 tool PredictGATKWGSMetrics looks at metrics generated by GATKWGSMetrics. The metrics produced contain multiple measurements, including the average coverage for each bin as well as the proportional coverage contribution of a bin. Each average was each scaled by the proportion of coverage contribution, then multiplied by the scale of desired increased coverage. Each scaled average was used to produce a new Poisson distribution that was then aggregated to create a composite prediction model (equations 1-3). In each case the prediction was compared to actual coverage distribution by measuring the sum of absolute differences between actual and expected coverage values.

A new resource was created and used to stratify genome locations by their prior coverage in a panel of twenty-four HapMap and 1000 Genomes samples (table 6). Each of the samples was previously sequenced and aligned for internal research purposes by other groups within the Broad Institute, however the samples are also available for other researchers to use from the Coriell biobank (“Coriell Institute,” n.d.). The samples formed a reference panel of coverage across a diverse background of anonymous human donors. Using a panel of BAM files processed by the same sequencing center that generated the PCR+ and PCR- samples also reduced potential biases introduced by alternate alignment workflows.
Equation 1. Poisson Single Prediction.
A probability distribution ($P(x)$) based on single predictor of coverages ($P_o$). This estimate uses a single average coverage over all loci ($\lambda$) scaled by a factor ($\pi$). As shown in figure 1 this model underpredicts lower coverage regions.

$$P(x) \propto P_o(x|\pi\lambda)$$

Equation 2. Poisson Per-Locus Prediction.
On the other extreme from a single poisson prediction, a probability distribution ($P(x)$) may be produced by aggregating predictions created per-locus. Across the whole genome reference ($R$) each locus ($l$) produces an individual coverage distribution ($P(x|L i l)$), scaled by the proportion of the reference covered by each locus ($P(l)$), which is a constant ($1/|R|$). The coverage of the locus ($\lambda_l$) scaled by a single factor ($\pi$) produces a per-locus prediction ($P_o$). The sum of these per-locus predictions produces an aggregate distribution used for predictive purposes.

$$P(x) \propto \sum_{l \in R} P(x|L i l)P(l)$$

$$P(x) \propto \sum_{l \in R} P_o(x|\pi\lambda_l)P(l)$$

Equation 3. Poisson Mixture Prediction.
An alternative between creating a single mean coverage and the other extreme of producing a per-locus distribution. Coverage distributions are first separated into groups ($g'$) based on covariates that assign loci to different bins. Each group produces an independent distribution of coverage ($P_o$) based on the mean coverage calculated per that group ($\lambda(g')$). Each distribution of coverage is scaled by the contribution of coverage for the group ($P(g')$). The results are combined into an aggregated prediction of coverage using the scaling factor ($\pi$).

$$P(g') \propto \sum_{g' = g(l)} P(l)$$

$$P(x) \propto \sum_{g'} (P_o(x|\pi\lambda(g')))P(g')$$
A new GATK4 walker called the GATK-WGSCoverageGenerator traversed portions of the genome producing the depth of the HapMap panel location and storing the results in a sites-only VCF.\(^1\) Using the created HapMap panel of genomic depth, a coverage stratifier was able to divide up the genomic locations into bins. Each bin in the panel was based on the sum of prior coverage across the multiple samples. Bins of coverage depth of 25, 50, and 100 were analyzed for their predictiveness.

Another reference track was also tested for predictions based on information relative to genomic locations. Karimzadeh et al. (2017) created bedGraph estimating the mappability of each location in the reference based on K-mer lengths of 24, 36, 50, or 100 bases long. The mappability track measured the uniqueness of each location, with 1.0 being completely unique to 0.0 being not unique at all. For compatibility with the GATK4 that only supports BED and not bedGraph, the 100 base K-mers file was converted to the BED format in a preprocessing WDL script.

The mappability track was provided as a new stratification for each genomic location with the score multiplied by 100. Bins were then selected to capture mappability windows of 10, 5, and 20. Due to the non-overlapping of the bins, completely unique locations with mappability equal to 100 were stratified into their own bin. If a genomic location was not present within the track it was assigned to a bin that tracked all unknown mappability regions.

Lastly a reference track of genomic variability was extracted for stratification purposes directly from the reference data. For each genomic location, the base of the genomic location was combined with 75 bases upstream and 75 bases downstream of the location. This produced a window of 151 bases where then a proportional ratio was calculated comparing the count of G or

\(^1\) The VCF format was chosen as it can store multiple annotation values per loci. VCF is also well supported by the GATK4 for reading, writing, and indexing.
C bases to the sum of A, T, G, or C bases. A symmetrical window was selected for analysis to account for the fact that reads overlapping any location could be forward or reverse reads, either increasing or decreasing in base quality as the walker continued to move downstream.

Each of the stratifications was tested individually to measure independent performance. To evaluate the predictive results of combining stratification counts there was also an analysis that stratified genomic locations by mappability and the HapMap coverage. The mappability and HapMap coverage bins were also stratified into much smaller bins by GC of 0.5%.

As many steps were computationally intensive custom gatherer utilities were created to summarize the reference and read counts from the various GATKReport objects. The gathering steps performed by GatherGATKWGSMetrics include not only adding up the previous counts but also a recalculation of the measured and estimated distribution statistics including variance, mean, and dispersion. To efficiently scan through the tables within the GATKReport, custom indexing code was developed and inserted into a copy of the GATK4 engine to allow fast retrieval of counts even when parsing through thousands of lines of whitespace separated text.

The GATK4 includes several default filters for removing reads that may not be ideal for analysis or that may be artifacts of bad sequencing or alignment. Additional filters used for GATKWGSCoverageGenerator included removing reads that: were listed as secondary alignments, were flagged as not passing vendor quality thresholds, had a mapping quality less than 20, were paired-end reads where mates aligned in unexpected orientations, or were reads that did not have a mapped paired end at all.

One filter often used for traversal of BAM files is the duplicate read filter. This filter eliminates reads flagged as being either duplicates of DNA from the PCR process, or duplicates due to upstream artifacts during sequencing. As the original DNA library preparation procedure
often includes cycles of PCR amplification, it is possible that those duplicate strands of DNA could end up in disjoint read groups. When combined into a full genome BAM file the Picard MarkDuplicates would tag reads independent of their read groups. For the purposes of this coverage analysis there was interest in predicting coverage even within the areas of higher duplication due to the PCR processes. To observe the differences in predictability for libraries prepared with and without PCR a sample using each process was analyzed and labeled either PCR+ or PCR-, and the GATKWGSCoverageGenerator did not filter out any reads marked as duplicates.

The sample NA12878 used for analysis as this DNA is commonly used as a baseline for method analysis and is freely available for a wide variety of research purposes. Samples prepared with and without PCR amplification have previously been sequenced and aligned for purposes of internal methods development within the Broad Institute and elsewhere. For the prediction analysis, the first read group from each BAM was selected and then a read group blacklist was created to exclude all other read groups. The blacklist was then fed to the GATK4 engine to remove the reads from traversal. This produced counts of coverage for only the selected read group. Separately, the unfiltered full sample was also run without a blacklist and with stratifications through the GATKWGSMetrics suite to produce the actual distribution of coverage, plus a total sum coverage of genomic bases. In each case this total sum coverage was fed to the PredictGATKWGSMetrics as the target whole genome coverage.

Numbers in the realm of human whole genome calculations usually require 64 bits of memory for signed fixed point values. For example the human reference genome is $3.2 \times 10^9$ bases long. However the maximum value a fixed-point number in WDL Version 1.0 may represent is $2.7 \times 10^9$ as the type only stores 32 bit integers, unlike the 64 bits available for WDL.
floating-point numbers. Thus for the purposes of passing exact coverage amounts collected by GATK4WGSMetrics to the predictor PredictGATK4WGSMetrics the workflow used a string variable for the genome wide counts.

In addition to the GATK4 and its associated libraries including the HTSJKD, Apache Commons Math was used for producing Poisson distributions for each of the averages (“Math – Commons Math: The Apache Commons Mathematics Library,” n.d.). For the purposes of visually analyzing the data, R and RStudio were used (R Core Team, 2018; RStudio Team, 2015). The GATKReports were able to be read directly into R by an existing gsalib library (Garimella, 2014). Data manipulation was done using tidyverse and magrittr (Bache & Wickham, 2014; Wickham, 2017). The tidyverse library ggplot2 and the cowplot extension were used for plot generation along with extrafont for formatting (Chang, 2014; Wickham, 2009; Wilke, 2017).
Chapter III

Results

The hypothesis of this thesis proposes that stratifying coverage improves Poisson based predictions of sequencing coverage. The analyses began with the reads extracted from a single read group in a full sample BAM. A prediction of full sample was generated from the subset of reads. The prediction was then compared versus the actual distribution of reads in the full sample.

Reads from the full PCR- sample measured an average (µ) depth of 37.778 across the genome with a variance variance (σ²) 117.12 producing a dispersion (D = σ²/µ) of 3.10. The dispersion for a Poisson must be 1.0, as the mean would be equal to the variance. For predictive purposes the dispersion should be as close to 1.0 as possible to approximate a Poisson distribution. The subset of reads from the PCR- read group measured an average coverage 1.75 per loci with a variance of 3.01 for a dispersion of 1.72.

The total number of bases covered by the traversal of the GRCh38 autosome measured $2.7 \times 10^9$ loci. The sum coverage of all the bases analyzed was $4.8 \times 10^9$ for the read group and $1.0 \times 10^{11}$ for the full sample produced without PCR. To produce the predicted distribution of the full sample the read group average coverage was therefore scaled by 21.64 to approximate the same coverage. When comparing the distribution of a prediction from a single read group to the actual coverage, the differences were summarized using a metric calculated by taking the absolute difference between the two distributions at each point (equation 4). The best case divergence is 0.0 and a worst case difference would be 2.0. In the case of the single
approximation the difference between predicting using a single average and the actual coverage was 0.27 for the NA12878 PCR-sample.

\[
\Delta = \sum_{i=0}^{250} |(\text{actual}_i - \text{predicted}_i)|
\]

Equation 4. Prediction Difference Metric. The prediction difference for any given model is the sum of the absolute differences between the actual and predicted coverage distributions across coverage values from zero to 250 inclusively.

Compared to using a single Poisson for prediction, using the mappability track to stratify locations in the genome produced closer predictions to the actual sequencing depth around regions of average coverage. The stratifications based on scaled mappability scores of 5, 10, 20 produced 22, 12, and 7 bins. Overall for the PCR-sample the bulk of the dispersions were closer to 1.0 than the single approximation (figure 2A). Some bins had dispersions higher than 10.0, however they contributed to a relatively small amount of read coverage, thus they did not contribute greatly to the prediction model.

Each mappability bin predicted similar distributions of coverage, each differing from the actual coverage distribution by 0.19 for the sample prepared without PCR amplification (figure 2B). Near the average peak the mappability prediction curves were closer to the actual coverage than the single predictor. The single Poisson predictor inconsistently estimated coverage, overestimating in regions of average coverage, while underestimating for below average regions particularly regions of low coverage. Importantly there are regions of the genome sparsely covered by reads even after multiple read groups were aggregated. The mappability stratifications did not show a strong increase in prediction in these lower coverage sections. The
actual coverage shows that there is a peak of genomic loci with zero coverage that was still not predicted by using the mappability stratification.

Figure 2. PCR- Mappability Stratification Dispersions and Predictions. The bulk of dispersions (A) for the mappability stratifiers, based on the Umap bedGraphs, are closer to 1.0 than the dispersion from a single mean and variance calculated for the read group data. At the actual average coverage depth, the predictions based on mappability show a lower coverage prediction compared to the single Poisson (B). However both the single Poisson and the mappability stratifier underestimate the number of reads covering reference bases with greater depth.

One stratification that did increase coverage predictions for genomic loci with relatively low coverage was the HapMap reference panel stratification. The stratifications of the reference by prior analysis produced coverage depth ranges of 25, 50, and 100 reads broken into bin counts...
of 401, 201, and 101 respectively. In each case the last bin in each stratification contained many regions of the genome measuring panel coverage depths over 10,000 reads, in aggregate, across the reference panel samples.

The number of bins was also large enough to study the mean to variance ratio of relatively small aggregate coverage distributions. The bulk of the dispersion density for the reference panel bins was clustered closer to 1.0 than the single Poisson predictor. There were a number of bins with larger Poisson values over 10.0 yet the largest bin coverage measured dispersions closer to 1.0 than the single dispersion taken from the PCR-read group coverage (figure 3A).

The HapMap reference panel stratification helped with predicting the number of regions that would ultimately receive zero or little coverage. For example, when using 101 bins to stratify the autosome loci by prior HapMap depth, the model predicted from a single read group that the full sample coverage would have 1.07% of loci, or $2.9 \times 10^7$ bases, lacking coverage. The actual percentage of uncovered loci was 1.91% in the full sample analyzed without PCR (table 1). Stratifying by more bins increased the predictions of uncovered locations, as breaking the prior coverage stratification into coverage bins of 25 depth estimated that 1.33% of the full sample would be completely uncovered.
Figure 3. PCR- Reference Panel Stratification Dispersions and Predictions. The reference panel produces a large number of bins however most of the dispersion is closer to 1.0 than the single bin estimation (A). Many of the bins with much higher variance-to-mean ratios actually represent very few reads. For the purposes of predicting low coverage reads the reference panel starts to predict low coverage rates, but does not perform as well at prediction average coverage as the prior mappability stratifier.

In each case of different bin sizes the HapMap reference coverage helped not only for zero coverage but also for low coverage regions as well. Only 17.77% of the autosome was covered by four or more reads by the original read group. Using a single Poisson, prediction estimated that, when the average coverage amounts increased to the levels of a full sample, 100% of bases would be covered by four or more reads. However the actual number for the full sample was 97.57% covered. Using the HapMap samples to stratify the prior read depth per loci.
into 100 bins predicted that for the full BAM 97.99% of the genome would be covered by four or more reads.

Table 1. Reference Panel Low Coverage Predictions.

<table>
<thead>
<tr>
<th>coverage</th>
<th>pcr- rg</th>
<th>pcr- single</th>
<th>pcr- map_20</th>
<th>pcr- refcov_100</th>
<th>pcr- actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>22.02%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>1.07%</td>
<td>1.91%</td>
</tr>
<tr>
<td>1</td>
<td>28.12%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.65%</td>
<td>0.25%</td>
</tr>
<tr>
<td>2</td>
<td>23.43%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.21%</td>
<td>0.15%</td>
</tr>
<tr>
<td>3</td>
<td>14.38%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.08%</td>
<td>0.11%</td>
</tr>
<tr>
<td>&gt;= 4</td>
<td>12.06%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>97.99%</td>
<td>97.57%</td>
</tr>
<tr>
<td># uncov bases</td>
<td>2,414,219,081</td>
<td>0</td>
<td>0</td>
<td>55,267,003</td>
<td>66,652,719</td>
</tr>
</tbody>
</table>

Percentages of bases covered in the analyzed read group (rg) and the whole sample (actual). The prediction models from a single Poisson (single), and using mappability for stratification (map_20) both estimated that 100% of bases would be covered when the read group coverage was scaled to the amount of coverage in the full sample. However using the reference panel more accurately predicted that while the majority of bases would receive coverage, some subset of loci would not have coverage of at least four reads (# uncov bases).

A stratification using the GC ratio was computationally very fast for the tool compared to looking up data values from an external file such as the mappability track or the coverage amounts stored by the reference panel. Using a GC calculation window of 151 bases meant that even by stratifying at 0.5%, intervals produced 184 bins. Some of the bulk of dispersion density clustered closer to the ideal 1.0, but the density also spread out to be greater and also less than the dispersion measured from a single average and variance of coverage (figure 4 A). For a small amount of coverage there were stratification bins that measured even less than 1.0 for their dispersion, yet this was only for a very small amount of coverage contributing very little overall to the predictions.
The GC stratifications did not produce as much change in prediction model as the mappability track nor the HapMap prior coverage. The lower coverage regions predicted only 82 autosomal loci would be sparsely covered instead of the 66,652,719 bases that were covered by three reads or less in the full PCR-sample (table 2). In the average coverage regions the prediction only slightly improved as compared to both the mappability and the prior reference coverage predictions (figure 4 B).

Figure 4. PCR- GC Stratification Dispersions and Predictions. Stratifying the genome using the ratio of GC bases within a window of the loci produced a number of dispersion bins, some of which were closer to 1.0 and others that were further from 1.0 relative to the calculated single dispersion across the read group data (A). A small number of reads even landed in bins that had dispersions less than 1.0. The number of reads in these dispersion bins was very small relative to the overall coverage of the read group. The predictions based on the GC stratifications did slightly improve, however they grossly underestimated the uncovered bases and the number of loci covered at or near mean depth in the full sample.
Table 2. GC Low Coverage Predictions.

<table>
<thead>
<tr>
<th>coverage</th>
<th>pcr- gc_0.5</th>
<th>pcr- actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000000%</td>
<td>1.912689%</td>
</tr>
<tr>
<td>1</td>
<td>0.000001%</td>
<td>0.253671%</td>
</tr>
<tr>
<td>2</td>
<td>0.000000%</td>
<td>0.152000%</td>
</tr>
<tr>
<td>3</td>
<td>0.000002%</td>
<td>0.109625%</td>
</tr>
<tr>
<td>&gt;= 4</td>
<td>99.999997%</td>
<td>97.572015%</td>
</tr>
<tr>
<td># uncov bases</td>
<td>82</td>
<td>66,652,719</td>
</tr>
</tbody>
</table>

A coverage prediction model based on GC stratifications predicts that even after additional sequencing that some locations will still have 0-3 read depth. However this estimation is much lower that the actual amount of uncovered reference bases.

The best improvement on the naive Poisson prediction model was ultimately a combination of reference-based stratifications. A mixture of a mappability stratification using bins of size 20, crossed with reference coverage stratification bins of size 100, showed an increase in the zero to lower coverage and average coverage predictions for regions across the genome. Using both stratifications created 707 bins of predictive read coverage. Stratifying the bins even further into by GC bins of 0.5% produced 57,568 bins. A number of bins contained dispersions around 1.0 (figure 5 A) but like the GC-only stratification this was a small amount. However even the bulk of the dispersions were clustered closer to 1.0 than single approximation, for both analyses with and without additional GC stratification.

The increase in number of bins ultimately improved predictions according to the difference metric. The difference for the stratification combination without GC was 0.16 while adding GC stratification decreased the difference to 0.15. Between the two predictions, an additional 1,286,065 bases in the autosome were predicted to be uncovered by four or more reads in the full PCR- sample.
Figure 5. PCR- Combined Stratification Dispersions and Predictions. Adding GC to the stratification by mappability and prior coverage analysis produces a number of bins with dispersion coefficients near 1.0 (A). The contribution is still relatively small compared to the majority of coefficients. However, the dispersions for the bins produced from the combined stratifications showed highest density nearer to 1.0 than a single predictor for a Poisson approximation. In the predictions (B) the distribution of lower coverage and average coverage are better predicted than the simple approximation.
Table 3. Combined Low Coverage Predictions.

<table>
<thead>
<tr>
<th>coverage</th>
<th>pcr- map_20 refcov_100</th>
<th>pcr- map_20 refcov_100 gc_0.5</th>
<th>pcr- actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.21%</td>
<td>1.24%</td>
<td>1.91%</td>
</tr>
<tr>
<td>1</td>
<td>0.54%</td>
<td>0.54%</td>
<td>0.25%</td>
</tr>
<tr>
<td>2</td>
<td>0.19%</td>
<td>0.20%</td>
<td>0.15%</td>
</tr>
<tr>
<td>3</td>
<td>0.11%</td>
<td>0.12%</td>
<td>0.11%</td>
</tr>
<tr>
<td>&gt;= 4</td>
<td>97.95%</td>
<td>97.90%</td>
<td>97.57%</td>
</tr>
<tr>
<td># uncov bases</td>
<td>56,335,237</td>
<td>57,621,302</td>
<td>66,652,719</td>
</tr>
</tbody>
</table>

Using combined stratifications predicts a non-zero number of bases that will not be covered by four or more reads. The mappability and reference stratifiers improve predictions when the GC stratifier is added into the modeling.

For comparison of the effects on the prediction model based on PCR amplification, a PCR+ sample was also used for measuring differences between the single and various aggregate prediction models. The particular read group prepared with PCR amplification started with less coverage than the read group prepared without. The PCR+ read group used for prediction measured an aggregate coverage depth of 2,922,313,641 (μ = 1.06, σ² = 1.93, D = 1.82) whereas the full sample had a coverage depth of 107,496,284,524 (μ = 39.16, σ² = 128.89, D = 3.29) producing a scaling factor of 36.78.

As compared to the PCR- predictions, the PCR+ predictions also improved using the same multiple stratifications (figure 6). Again, adding GC stratification to the existing mappability and HapMap depth stratifications produced a small number of dispersions closer to 1.0 than using the average and variance of depth from the single PCR+ read group. For the actual predictions similarly the PCR+ sample was able to estimate a percentage of uncovered bases in the full sample. Based on a single Poisson from the average of the read group’s coverage estimates, a predicted zero reads would be uncovered, while the combined model, including GC
stratification, estimated that 56,030,741 positions would still have three or less reads covering them in a full sample (table 4).

Figure 6. PCR+ Combined Stratification Dispersions and Predictions. PCR+ plus reads show a similar dispersion density (A) to the earlier PCR- density analysis. Similarly, the predictions of coverage (B) improve in the low coverage and average coverage regions.
Table 4. PCR+ Low Coverage Predictions.

<table>
<thead>
<tr>
<th>coverage</th>
<th>pcr+ rg</th>
<th>pcr+ map_20 refcov_100</th>
<th>pcr+ map_20 refcov_100 gc_0.5</th>
<th>pcr+ actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>38.30%</td>
<td>1.05%</td>
<td>1.09%</td>
<td>1.90%</td>
</tr>
<tr>
<td>1</td>
<td>33.05%</td>
<td>0.60%</td>
<td>0.60%</td>
<td>0.23%</td>
</tr>
<tr>
<td>2</td>
<td>17.92%</td>
<td>0.22%</td>
<td>0.23%</td>
<td>0.15%</td>
</tr>
<tr>
<td>3</td>
<td>7.24%</td>
<td>0.10%</td>
<td>0.12%</td>
<td>0.11%</td>
</tr>
<tr>
<td>&gt;= 4</td>
<td>3.48%</td>
<td>98.02%</td>
<td>97.96%</td>
<td>97.61%</td>
</tr>
<tr>
<td># uncov bases</td>
<td>2,649,656,057</td>
<td>54,278,406</td>
<td>56,030,741</td>
<td>65,645,757</td>
</tr>
</tbody>
</table>

The original PCR+ read group only covers 3.48% of the autosome with four or more reads. After using the supplemental coverage prediction based on aggregate Poisson approximation the 97.96% of the genome is predicted to be covered. Like in the PCR- analysis, the mappability and reference stratifiers improve predictions when the GC stratifier is added into the modeling.

Another area of interest in coverage analysis are areas where there will be extremely high coverage. The original read group in the PCR- sample had 20,095 reference bases with at least 250 reads covering their locations. However using a single Poisson approximation of the coverage means, even after scaling the average coverage for the full sample, the model would predict that zero locations would be covered at higher coverage. Instead by dividing the genome into regions using the combination of stratifiers the new estimation of uncovered higher depth regions increases measures to 413,123 in the PCR- sample (table 5).
Table 5. High Coverage Predictions.

<table>
<thead>
<tr>
<th>coverage</th>
<th>pcr-rg</th>
<th>pcr-map_20 refcov_100 gc_0.5</th>
<th>pcr-actual</th>
<th>pcr+rg</th>
<th>pcr+ map_20 refcov_100 gc_0.5</th>
<th>pcr+ actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= 250</td>
<td>0.001%</td>
<td>0.015%</td>
<td>0.014%</td>
<td>0.001%</td>
<td>0.018%</td>
<td>0.018%</td>
</tr>
<tr>
<td># high cov bases</td>
<td>20,095</td>
<td>413,123</td>
<td>395,609</td>
<td>16,581</td>
<td>486,475</td>
<td>497,593</td>
</tr>
</tbody>
</table>

A mixture of samples from a wide cohort of ethnicities were previously sequenced by the Broad Institute. A walker traversed all 24 samples together to produce a sum coverage per location. The results were stored in a file for the purpose of characterizing areas of depth during subsequent coverage predictions when analyzing new read groups.

In summary, this analysis evaluated several ways of stratifying the reads overlapping a genome from a single read group. The predictions when compared to the measurements from a fully sequenced sample differed for a number of the analyses (figure 7). Compared to using a single average for predicting the coverage distributions when read depth approached full sample levels, the stratifications for insert length did not affect the predictions at all. Stratifying the genome by a mappability score decreased the difference between the prediction and the actual distribution, but an increase in stratification bins did not affect the prediction. Predictions using a reference panel were improved by increasing the number of bins of stratification. The predictions of reference panel stratification were also improved by leaving the Hapmap bin sizes at their widest thresholds, and instead increasing the number of bins by dividing the genome by mappability as well. These predictions were further improved by significantly stratifying the genome by thousands of additional bins achieved by adding a GC based stratification. The GC bins by themselves did not improve differences in prediction as much as the mappability and coverage panel. Each of the above prediction similarities was seen in samples produced both
with and without PCR amplification during the sample library preparation process, including for the independent stratifications (appendix figures 9-12).

Figure 7. Summary of Prediction Differences. Each of the shown stratifiers boosts prediction performance over a single poisson model. Predictions do not improve by changing the bin size of the mappability or GC stratifiers alone. Adjusting the windows of reference panel coverage, from 100 down to 25, does improve the estimates. Alternatively, instead of shrinking the reference panel coverage bins, mixtures models also better approximate the actual coverage. This is seen when first combining the reference panel stratifications with mappability, and then is reinforced by adding a GC stratification.
Chapter IV
Discussion

Using the coverage priors from the HapMap reference panel increased predictions in the lower coverage areas. This is possibly caused by some portions of the reference consistently being unable to collect aligned reads, even across multiple samples. Using bins to separate out these locations for predictive purposes meant that we could measure the coverage within those isolated regions and then scale up the contribution of coverage using the aggregated Poisson model. The underestimation of the percentage of zero covered reads is most likely due to the overestimation of bases covered by average coverage. If the estimates of average coverage can be better predicted, with dispersions closer to 1.0, then the predictions of minimal covered regions should benefit from the more accurate aggregate Poisson approximations.

A related investigation of factors could focus on improving the initial dispersions of the stratified bins. The goal would be for the majority of bins to contain reads with coverage measurements where the variance approaches the mean coverage, approximating a true Poisson distribution. In this search for moving the dispersion coefficients closer to 1.0, other read covariates to consider include stratifying the genome based on the average mappability score assigned to the reads. Using read-based mappability scores in a reference panel could complement the reference-based K-mer mappability, or uniqueness scores. The average read mappability across a panel of samples would infer the prior ability of the aligner to uniquely map reads to each loci. In that case the stratification would be performed by populating additional annotations in the VCF generated by the tool that generates the coverage track, currently titled GATKWGSCoverageGenerator.
Stratifying the genome based on GC content produced a number of bins with dispersions closer to one, and improved predictions of full sample coverage. A follow-up investigation could look into whether expanding the GC windows sizes would provide additional context for locating problematic areas of the genome with lower coverage. This could enhance the performance of the prediction analysis tool as the GC ratio calculations may be computed using an efficient sliding window, even for large window sizes. This will be faster than retrieving values from an external VCF resource which requires more resources to decompress, decode, and lookup. However by itself GC binning was unable to account for areas of the genome that remained uncovered after additional sequencing. A mixture of covariates will still likely bestow improved results.

Mixing GC stratification into the mappability did improve results, even though it increased the number of overall bins well into the tens of thousands. These small bins with varying amounts of coverage could have affected the coverage predictions, but the majority of dispersions in the combined prediction model were grouped near 1.0. These large numbers of bins also increased size of the files generated for metrics tabulation, increasing from 3.5MB to 73.5MB when adding GC to the combination stratifier. Finding the optimal set of stratification combinations that are performant for disk, memory, and compute will be a consideration for subsequent study.

Using a reference panel approach also involves selecting appropriate background samples to reduce potential bias in results due to ethnicity (Kessler et al., 2016). The HapMap reference panel sequenced by the Broad was a good choice due to its diversity, and the production similarity to the NA12878 PCR+ and PCR- samples. However those who are looking to use a similar predictive analysis on their own sequence data should ideally have a wide selection of
samples available for creating their own reference panel. Even as the community advances in
gold standard sequencing, biases remain due to variations in the pipelines selected for data
production (Kessler et al., 2016; Zook et al., 2014). While the samples are open access there may
be additional preparation and versions of the panel required as pipelines undergo upgrades and
modifications. Thus the ideal predictive model does not require a panel. A future in-depth study
of the lower coverage areas in the panel could help identify characteristics that a prediction tool
could scan for independently.

It may be determined that a reference panel of coverage is the best way of estimating
future coverage. In this situation a large cohort analysis similar to gnomAD (Lek et al., 2016)
could generate average coverage depths. The results could be filtered in a way such that the data
could remain open access. Already allele fractions from larger cohort studies are available for
population and clinical study. A similar average coverage depths analysis may prove equally
valuable for sequencing depth prediction.

Another sample to consider for evaluation beyond NA12878 would be a synthetic diploid
created from a pair of homozygous samples (H. Li et al., 2017). This new benchmarking sample
will be important in evaluating variant analyses. The novel diploid may also become a popular
open access sample tool used for other purposes such as the coverage predictions done in this
study.

For this analysis the evaluation of the confidence of Poisson approximations was
examined using the dispersion coefficients. As a Pearson’s chi-squared test does not work well
where the expected number of results is greater than five, a Fisher exact test was considered
(West & Hankin, 2008). However the large numbers of reads in the genomic analysis quickly
overflowed the calculations available via Apache Commons and R. It is possible that converting
the discrete measurements of coverage may be measured with other test statistics; however, this
would require additional study or collaboration outside of the author’s current study and
experience. One common suggestion worth looking into would be the Kolmorogov-Smirnov
(KS) test (Lilliefors, 1967).

Exomes are frequently used to save money on analysis but add additional variables to
consider for prediction analysis. The process by which only exonic regions are extracted from
libraries for sequencing will introduce its own biases in the coverage (Samorodnitsky et al.,
2015). The borders of targeted sequencing areas will also show an increase in depth up to the
desired depth, and even then the coverage may not be uniform across exonic regions. At a
minimum one should evaluate stratifications based on the distance from the edge of targets, and
possibly more covariates would need to be specially designed for exomes prediction.

The technique of aggregating Poisson approximations has improved predictions of
coverage, even while starting with a low coverage of reads. Studies may now estimate additional
coverage for each sample with better accuracy using the stratification of a single read group.
These improved assessments should save time, resources, and money for sequencing laboratories
and data scientists.
During active development of new tools there often needs to be a way to evaluate the tools within the context of a larger pipeline. As new GATK4 tools were being tested at whole genome scale the pipeline was also designed to run on two megabases of chromosome twenty. This enabled the pipeline to quickly test the various utilities assisting with early identification of issues before having the entire whole genome results finished. If an error was detected Cromwell was configured to abort the existing workflow run. Due to Cromwell’s call caching functionality in many cases reruns with a modified pipeline were able to copy the outputs of previous workflows that had not changed.

The gatherer code was initially very slow. JProfiler is a tool for measuring hotspots of high usage either of CPU, memory, or other resources such as idle compute-threads (“Java Profiler - JProfiler,” n.d.). Profiling the tools helped isolate locations that could be optimized such that the program could run in an efficient amount of time. In one case the composite keys used in the GATK4 report tables were made up entirely of Java strings. The profiler showed that there were thousands of string-formatting operations occurring as the report was being generated, significantly slowing down the execution report tables. The number of string formatting operations was reduced by storing the rows as a heterogeneous list of elements. In the list elements, instead of always using Java strings, values were stored as the most appropriate type, often as Java longs for counting. Upon the final writing of the GATKReport the table generation was sped up by adjusting the formatting of the floating and fixed point values. Floating point
numbers required truncating them to eight digits after the decimal. For all other string conversions that did not need to customize the length of the numbers, using a string formatter was significantly slower than just calling the built-in Java method for converting any object to a string.

There were scaling issues that popped up due to the way this genomic analysis was performed relative to the size of the sequencing depth. For example, removal of the duplicate filtering required refinement of the memory parameters for the Java process running the GATK4 metrics tool. On the on-premise shared clusters reserving and using more memory per job reduces for the number of jobs that can run simultaneously on the limited number of compute hosts. Meanwhile on the cloud environments reserving and using more memory per job requires an additional cost-per-time of use of the machine. When duplicates were being filtered for the purposes of counting read coverage the overall coverage was relatively low across the genome. However after removal of the GATK4’s read duplicates filter, there were several genomic regions where there were significantly higher rates of duplication compared to the average genome location. To empower the GATK4 to produce pileups of reads in these locations for further stratification required significantly more memory. Because there was no prior expectation of coverage for these regions instead all jobs across the genome were given a relatively high amount of memory for processing.

One way that could significantly reduce time and cost of the analysis would be to optimize the amount of memory required for a region of the genome. This could be accomplished by using a file similar to the reference coverage track, preprocessed and adapted into a WDL friendly file format such as TSV. Then as interval subsections were traversed more memory could be allocated to extremely deep regions of the genome while the rest of the tasks
could run with a minimal set of memory. An analysis like this could have performance improvements in other pipelines as well where downsampling of reads to a normal coverage may not be desired in an effort to avoid false negative results, for example searching for certain types of variations in complex cancer analyses (Ye et al., 2016).

Reference Panel

Table 6. Reference Panel Samples.

<table>
<thead>
<tr>
<th>Sample Id</th>
<th>Ethnicity</th>
<th>Gender</th>
<th>Sample Id</th>
<th>Ethnicity</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>HG00096</td>
<td>British</td>
<td>Male</td>
<td>HG02922</td>
<td>Esan</td>
<td>Female</td>
</tr>
<tr>
<td>HG00268</td>
<td>Finnish</td>
<td>Female</td>
<td>HG03006</td>
<td>Bengali</td>
<td>Male</td>
</tr>
<tr>
<td>HG00419</td>
<td>Southern Han Chinese</td>
<td>Female</td>
<td>HG03052</td>
<td>Mende</td>
<td>Female</td>
</tr>
<tr>
<td>HG00759</td>
<td>Dai Chinese</td>
<td>Female</td>
<td>HG03642</td>
<td>Sri Lankan Tamil</td>
<td>Female</td>
</tr>
<tr>
<td>HG01051</td>
<td>Puerto Rican</td>
<td>Male</td>
<td>HG03742</td>
<td>Indian Telugu</td>
<td>Male</td>
</tr>
<tr>
<td>HG01112</td>
<td>Colombian</td>
<td>Male</td>
<td>NA18525</td>
<td>Han Chinese</td>
<td>Female</td>
</tr>
<tr>
<td>HG01500</td>
<td>Iberian</td>
<td>Male</td>
<td>NA18939</td>
<td>Japanese</td>
<td>Female</td>
</tr>
<tr>
<td>HG01565</td>
<td>Peruvian</td>
<td>Male</td>
<td>NA19017</td>
<td>Luhya</td>
<td>Female</td>
</tr>
<tr>
<td>HG01583</td>
<td>Pakistani</td>
<td>Male</td>
<td>NA19625</td>
<td>African Ancestry</td>
<td>Female</td>
</tr>
<tr>
<td>HG01595</td>
<td>Vietnamese</td>
<td>Female</td>
<td>NA19648</td>
<td>Mexican Ancestry</td>
<td>Female</td>
</tr>
<tr>
<td>HG01879</td>
<td>African-Caribbean</td>
<td>Male</td>
<td>NA20502</td>
<td>Toscani (Tuscan)</td>
<td>Female</td>
</tr>
<tr>
<td>HG02568</td>
<td>Gambia</td>
<td>Female</td>
<td>NA20845</td>
<td>Gujarati Indian</td>
<td>Male</td>
</tr>
</tbody>
</table>

A mixture of HapMap and 1000 Genomes samples of various ethnicities were previously sequenced at the Broad Institute. A new GATKWGSCoverageGenerator traversed all 24 samples together to produce a sum coverage per location. The results were then used with by GATKWGSMetrics to characterize the genome by areas of depth while predicting coverage for the NA12878 PCR- and PCR+ samples.
Read Based Stratification Negative Results

One experiment that did not produce results was stratifying coverage by read based covariates. Early in the design of the experiment, several possible covariates were combined into a filter. The preliminary results showed that a subset of reads appeared to fit a poisson distribution as compared to the whole genome. While ultimately the project showed that a using a mixture of poissons does enhance predictions, the research also showed that using TLEN did not contribute to the improvements.

To evaluate stratifications based on information from the reads, a stratifier was created to extract the insert length attribute as an integer from each read and then place value into a bin, collecting insert lengths of one hundred bases each. Insert sizes are characterized by a TLEN field stored per aligned read within BAM files, originally defined as Template LENgth. The TLEN field has not been well defined by the specification and is often stored instead as the insert size between two paired-end reads (“picard / Wiki / Differences_between_Picard_and_SAM,” n.d.). The TLEN distributions may therefore differ between samples processed with competing bioinformatics tools. However the insert size values are still expected to conform across read groups produced by the same read alignment pipeline.

Three stratification sizes were analyzed based on the read insert size. The TLEN field was extracted from the reads, converted to an absolute value, and then separated into bins 50 bases, 100 bases, and 200 bases wide. The minimum bin started at zero and extend until one base before the next bin. A bin was also created to collect large insert sizes of 600 bases or above.

As measured by various metrics, bins of stratification did not generated better or worse predictive results for the insert length stratifier. The stratifier produced 4, 7, and 13 bins for
TLEN values of 200, 100, and 50 bases wide respectively. The predictions from every binning stratification each produced a difference from the distribution of the full sample of 0.27. As the stratification sizes decreased in the number of insert size bases per bin, the dispersions generated for the bins grew both more toward and away from the dispersion of the single read group (figure 8 A). A small number of bins had dispersions in coverage close to 10.0. However the predictions for each set of bins did not vary more than a single bin prediction from the actual coverage distribution (figure 8 B).

The stratifier using the insert lengths of a subset of reads did not produce changes in full sample coverage approximations compared to the single Poisson model. Insert length is likely not a good variable to use for stratification purposes, as the sample preparation processes that affect insert size selection may also affect coverage depth (Schbath, Bossard, & Tavaré, 2000). There may still be other factors captured within read data that could be used for building better coverage predictions. These factors may best be read from a reference panel or a track. The track for reference mappability improved predictions, stratifying bins by repetitiveness and uniqueness of regions. This helped with average coverage estimates but still produced poorly performing estimates in low coverage regions.
Figure 8. PCR- TLEN Stratification Dispersions and Predictions.
A number of reads fell into bins of stratification based on insert length that measured dispersions less than and greater than the measured dispersion for a single Poisson (A). The predictions generated from the average coverage of each bin did not produce aggregated predictions that deviated from the actual coverage more or less than using just one mean coverage estimator (B).
Figure 9. PCR+ TLEN Stratification Dispersions and Predictions.
The PCR+ dispersions and predictions for the insert size stratifications show similar results to the PCR- sample (figure 8). Despite changes in dispersion density (A) the aggregated Poisson predictions over the single Poisson approximation predictions did not improve.
Figure 10. PCR+ Mappability Stratification Dispersions and Predictions. The majority of coverage measures an increased dispersion (A), while the prediction improves especially at and before the average coverage (B). These changes match observations for the PCR- sample (figure 2).
Figure 11. PCR+ Reference Panel Stratification Dispersions and Predictions. Most dispersion is located closer to 1.0 while many bins have small amounts of coverage at higher dispersions (A). The prediction improves similar to mappability near the average but less so just before the average (B), again as was seen in the PCR- sample (figure 3). The zero to low coverage regions show better prediction improvements compared to other stratifications.
Figure 12. PCR+ GC Stratification Dispersions and Predictions.
As in the PCR- sample (figure 4) there are a small number of reads in bins close to dispersion 1.0 (A). Overall the stratifications by GC content only produce minor improvements in the predictions (B).


BLAST - NCBI. (n.d.). Retrieved July 13, 2016, from


Lek, M., Karczewski, K. J., Minikel, E. V., Samocha, K. E., Banks, E., Fennell, T., … Exome


Retrieved from http://pandas.pydata.org/

Odersky, M., Altherr, P., Cremet, V., Emir, B., Micheloud, S., Mihaylov, N., … Zenger, M.


Sun, Y., Ruivenkamp, C. A. L., Hoffer, M. J. V., Vrijenhoek, T., Kriek, M., van Asperen, C. J.,


