A Large Scale, Cloud-Based, Low Cost and Reproducible Mutation Calling Pipeline Using Docker Containers.

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

Citation

Citable link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37365103

Terms of Use
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 http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA
A large scale, cloud-based, low cost and reproducible mutation calling pipeline using Docker containers.

Noel Namai.

A Thesis in the Field of Mathematics and Computations for the Degree of Master of Liberal Arts in Extension Studies.

Harvard University.

May 2019.
Abstract.

The identification of variants that occur in the human genome remains a critical step in the analysis of Next Generation Sequencing (NGS) data. Accurate and timely variant calling is essential to precision medicine which seeks to relate particular genomic variations in a patient’s genome to targetable genes, drugs, and treatments tailored to each patient.

However, the analysis of these low-frequency variants requires sensitive algorithms within computational intensive pipelines. With high costs and lack of technology, most institutions resort to running bioinformatics pipelines as bash scripts on local clusters which are not only slow and cumbersome but also challenging to implement. This has in part led to reducing reproducibility and exhaustion of local data storage at such institutions.

Therefore, I demonstrate how to implement bioinformatics pipelines in the cloud using open source tools like Github, Docker and Broad Institute Genome Analysis Toolkit (GATK). This fast and low-cost implementation leverages parallel execution and auto-scaling within the cloud to handle the computationally intensive pipeline. I show that such pipelines produce accurate results by analyzing data from the 1000 Genomes Project. This would allow researchers to use more of their time doing research and less time configuring workflows.
Acknowledgments.

*If I have seen further, it is by standing on the shoulders of giants.*

~ Sir. Isaac Newton.

*Letter to Robert Hooke (15 February 1676)*

I would like to acknowledge my thesis director, Dr. Zoran Djordjevic. His courses in Big Data Analytics and Cloud Computing played a significant role in the conception and writing of my thesis. Equally, together with Dr. Jeff Parker, I would like to thank Dr. Sylvain Jaume the Director & Research Advisor of Technology Programs for the patience and help in the early stages of the thesis process. Finally, I am forever indebted to my mother, Consolata Namai, who has given me unwavering support throughout the years during my studies.
Table of Contents.

Table of Contents ................................................................. v
List of Figures ................................................................. viii
Preface ................................................................. ix

1. Introduction ................................................................. 1

2. Genomic Variant Discovery .................................................. 6
   2.1. The 1000 Genomes Project ................................................ 7
   2.2. Genomic Variant Discovery ............................................... 10
      2.2.1. Genetic variation ........................................... 10
      2.2.2. Next Generation Sequencing .................................. 11
      2.2.3. Variant identification and analysis ................................. 12
   2.3. The Genome Alignment Toolkit ........................................... 15
      2.3.1. Data preprocessing ........................................... 16
      2.3.2. Variant Discovery ........................................... 19
      2.3.3. Callset Refinement ........................................... 19

3. The Computing Framework ................................................. 21
   3.1. Cloud Computing ...................................................... 21
      3.1.1. Types of cloud deployments ................................... 22
      3.1.2. Cloud Computing Models ....................................... 24
      3.1.3. Advantages of Cloud Computing ................................ 25
   3.2. Amazon Web Services ................................................. 26
      3.2.1. Global Infrastructure ........................................... 26
# List of Figures.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Growth of DNA sequencing</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Genetic variation in the 1000 Genomes Project</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>The distribution of individuals’ genomes sequenced in the 1000 Genomes Project</td>
<td>9</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Types of Genetic variation</td>
<td>13</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>The AWS Global Infrastructure</td>
<td>28</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Containers and virtual machines</td>
<td>34</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>The Docker architecture</td>
<td>36</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>The AWS infrastructure</td>
<td>42</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>The core processing steps in the variant calling pipeline using GATK</td>
<td>46</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Runtime comparison for the local cloud deployment of the pipeline</td>
<td>61</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>Cost comparison for the cloud deployment of the pipeline</td>
<td>63</td>
</tr>
</tbody>
</table>
Preface.

Until recently, clinicians were only able to take Genomics and Genetics into consideration for the treatment and diagnosis of congenital disabilities and a limited set of diseases which had straightforward, predictable inheritance patterns which were caused by a change in a single gene.

The recent increase in data generated by genomic research has given scientists and clinicians more insight in the study of the role of multiple genetic factors in much more complex diseases such as cancer, diabetes, and cardiovascular diseases. However, this exponential growth has resulted in high sparsity data sets which require significant storage space and long processing time for computationally intensive algorithms that are used to analyze these data.

As such, the idea of Cloud Computing has gained much traction in genomics research. However, a large gap still exists between biological researchers and computer scientists in leveraging cloud computing in genomics. Therefore, this document seeks to describe how cloud computing can be used in genomics for research and try to bridge this gap.
Beginning in Chapter 1, I introduce and offer a glimpse into the world of Big Data. Genomics. I discuss the current and future use of high throughput genomics data and explore bottlenecks that might hinder the promise of Precision Medicine due to the exponential growth of DNA sequencing data.

Chapter 2 proceeds by horning into one of the most common areas of genomics. I introduce Genomic Variant Discovery by reviewing the 1000 Genomes Project and recent efforts to catalog human genetic variants. I also discuss various processes that the Genome Alignment Tool Kit algorithms provide solutions for. Most of these algorithms are computationally intensive of which are suitable for deployment in a cloud infrastructure.

After a general problem background, I introduce the computational framework that will be used in the project in Chapter 3. I discuss the concepts of cloud computing while emphasizing on the elasticity, reproducibility and privacy features that make it ideal for large-scale analysis and storage of genomics data. I also briefly discuss the concept of portable containers and virtualization that is taking the computing world by storm.

In the final chapter, I will discuss the various aspects of the project and show how to integrate bioinformatics pipelines with AWS Batch and seamlessly deploy and 

x.
execute the pipeline using data from the 1000 Genome Project. All the steps to
create the computing environment will be described in detail in the
Supplementary Material section.
Since the completion, the Human Genome Project (HGP) (Lander et al., 2001) and (Venter et al., 2001), the growth of the total number of genomes sequenced worldwide has increased rapidly. Improvements in NGS techniques have led to reduced costs and time required for genome sequencing, and it has been estimated that almost two billion human genomes could be sequenced by 2025 (Stephens et al., 2015).

Following the promise of genomic medicine to revolutionize the diagnosis and treatment of disease, many efforts are currently in place to enable the sequencing of large populations. Genome-Wide Association Studies (GWAS) have been instrumental in the discovery of genetic variations as a cause of both common and complex diseases like asthma, cancer, diabetes, heart disease, and mental illnesses.

For instance, England and Saudi Arabia through the 100,000 Genomes Project and the Saudi Genome Project respectively have announced plans to sequence 100,000 genomes each, one-third of Icelandic population has donated blood samples for genetic testing (Sulem et al., 2015), while the United States Precision Medicine Initiative (Collins & Varmus, 2015) and China Genome Project both aim to sequence one million genomes each in the next few years.
Figure 1: Growth of DNA sequencing (Stephens et al., 2015). The plot shows the growth of DNA sequencing both in the total number of human genomes sequenced (left axis) as well as the worldwide annual sequencing capacity (right axis: Tera-basepairs (Tbp), Peta-basepairs (PBP), Exa-basepairs (Ebp), Zetta-basepairs (Zbp)).

It’s easy to see that huge progress has been made since the initial sequencing of the first human genome. Unfortunately, a lot remains to be done in the development of large scale methods to process and analyze sequencing data produced in large population studies. It could, therefore, be said that cancer genomic analysis now hides behind terabytes of sequenced data and thus there 2.
is an urgent need for reliable data mining and classification methods that can bring NGS into routine clinical practice.

The wealth of genomic data created by NGS will, therefore, require efficient and accurate bioinformatic tools for the data analysis. With the advent of Precision Medicine, the detection of somatic mutations in cancer samples by Whole Genome Sequencing (WGS) and Whole Exome Sequencing (WXS) will become routine in cancer research and in the clinical setting where identification of somatic mutations forms the basis for personalized medicine. Therefore, highly accurate bioinformatic analysis pipelines are essential. (Krøigård, Thomassen, Lænkhølm, Kruse, & Larsen, 2016)

Ensemble approaches which integrate multiple variant callers have recently shown improved results as compared to other variant callers. For instance, in the recent ICGC-TCGA Dream Mutation Calling challenge, ensemble approaches showed improved accuracy in the identification of somatic variants. For instance, (Kim, Jacob, & Speed, 2014) developed a statistical model that combines multiple callers, (Fang et al., 2015) used an Adaptive Boosting model and (Rashid, Robles-Espinoza, Rust, & Adams, 2013) used majority voting to classify a variant as somatic.

It’s no doubt that the Human Genome Project will go on record as one of the
greatest scientific achievements of our time, a lot has to be done to maximize its impact on society. For instance, much more research is needed to detect biomarkers for disease susceptibility and response to drugs.

The initial sequencing of the human genome, set forth the second phase of human genomics. Genome-Wide Association Studies (GWAS) focuses on identifying genomic variations responsible for hereditary diseases and other medically relevant traits. GWAS archives this by examining the genomes of thousands of individuals for correlations between genomic variants and the trait of interest like human disease.

As such, Cloud Computing is becoming a popular venue for hosting and processing data from large genomic studies. It offers the ability to store data securely in a single location and near the computational infrastructure that will be used to analyze it. Also, Cloud Computing provides the ability to rent and pay for the exact resources needed, and thus the users do not have to deal with the additional cost of owning or maintaining the computational resources (Langmead & Nellore, 2018).

The genome has certain essential features e.g. association of genetic traits and diseases, identification capability especially in forensics and revelation of family relationships. This has led to increased concerns about personal privacy.
Likewise, direct-to-consumer DNA testing has led to increased availability of genomic data which in turn has increased the likelihood that genome data will be made available in less regulated environments (Naveed et al., 2015). Since Cloud clusters can be configured with security measures needed to adhere to privacy standards, this could be very useful since most genomic and medical data are subject to privacy laws.

In summary, with the exponential increase in genomic data, the cloud will be a major player to realize the benefits of precision medicine since it provides a one-stop shop for storage, computation, and security.
2. Genomic Variant Discovery.

With recent advances in genome sequencing technologies, it is now possible to use GWAS on the human genome to analyze genetic variation and further understanding of common human diseases. However, much more work is still required to accurately establish the frequencies and haplotype backgrounds for all variants.

The first part of this chapter briefly introduces the 1000 Genomes Project (1000 Genomes Project Consortium et al., 2015), which ran between 2008 and 2015. Here, I explore the data from this project which is considered the largest public catalog of human variation and genotype data. The data from this project will be used in the thesis for developing, testing, benchmarking and analysis of the pipeline developed.

The second and third part of this chapter discusses the idea of genomic variant discovery and genome analysis pipelines respectively. I explore the biology behind the algorithms and methods of variant discovery. I introduce the Broad Institute’s Genome Alignment Toolkit (GATK) (Van der Auwera et al., 2013), which remains the main component of the genome analysis pipelines.
2.1. The 1000 Genomes Project.

The central goal in biology and medicine is mapping the relationship between genotype and phenotype. The Reference Human Genome Sequence (International Human Genome Sequencing Consortium, 2004) published in 2004 provided a foundation for the study of human genetics, however fully achieve the promise of precision medicine, complete knowledge and understanding of genomic sequence variation across the entire genome are required.

Therefore, in a workshop to plan a “Deep Catalog of Human Genetic Variation” held on September 17, 2007, to September 18, 2007, in Cambridge, England, the following was laid out as the specific goals of the 1000 Genomes Project:

a) The discovery of SNPs that appear more than 1% or higher in diverse populations.

b) More comprehensive discovery (variants down to frequencies of 0.1 - 0.5%) in functional gene regions.

c) Discovery of Structural Variants, such as Copy Number Variants (CNVs), other Insertions and Deletions (Indels), and Inversions, including sequence level understanding of breakpoints.

The complete report is available in the following link:
When initiated in 2008, The 1000 Genomes Project was tasked for providing a comprehensive description of common human genetic variation by applying WGS and WXS to a diverse set of individuals from multiple populations (1000 Genomes Project Consortium et al., 2010). The foremost goal of the project was to find most of the genetic variants that occurred at least 1% in the populations studied.
By 2015, when the project finally ended, scientists had created the largest public catalog of human variation and genotype data. Using the genomes of 2,504 individuals from 26 populations, over 88 million variants (84.7 million SNPs, 3.6 million Indels, and 60,000 Structural Variants) were cataloged onto high-quality haplotypes (1000 Genomes Project Consortium et al., 2015).

Figure 3: The distribution of the 2,504 individuals from 26 populations whose genomes were sequenced in the 1000 genomes project.
2.2. Genomic Variant Discovery.

To achieve the promise of precision medicine and develop a better understanding of cancers, researchers are usually faced with the task of analyzing many cancer genomes. This requires the discovery and development of additional tools. The NGS data analysis process generally includes three main steps: primary, secondary, and tertiary data analysis. Genome sequencing instruments automatically perform an initial analysis, but the bulk of analysis steps occur after sequencing is completed.

2.2.1. Genetic variation.

In genetics, the term variant is used to refer to a specific region of the genome which differs between two genomes. Such an alteration may be benign, disease-causing, or of unknown significance. These could be termed as “variants” or “mutation”. Genetic variation is divided into three main forms:

a) Single nucleotide polymorphism (SNP).

Any nucleic acid substitution, whether a transition (interchange of the purine (A/G) or pyrimidine (C/T) nucleic acids); or a transversion (interchange of a purine and pyrimidine nucleic acid) (Figure 4 (a)).
b) Insertion or deletion (INDEL).

An Insertion (IN) or deletion (DEL) refers to a single stretch of DNA sequence ranging from two to hundred base-pairs in length (Figure 4 (b)).

c) Structural variation (SV).

Structural Variation refers to genetic variation that occurs over a larger DNA sequence. This includes both Copy Number Variation (CNV) and chromosomal rearrangement events. The five common types of structural variants are shown in Figure 4 (c).

2.2.2. Next Generation Sequencing data analysis process.

a) Primary data analysis.

This analysis is carried out during genome sequencing from tissue samples. This provides base calls and associated quality scores representing the primary structure of DNA or RNA strands. This is done by built-in software in sequencing systems automatically.

b) Secondary data analysis.

Secondary data analysis consists of the alignment and assembly of DNA or RNA fragments for a sample along a reference genome, from which genetic variants can be determined. This could be classified as either:
DNA sequencing data analysis: includes DNA sequence alignment, variant calling, and data visualization.

RNA sequencing data analysis: includes data analysis for a broad range of RNA sequencing experiments including gene expression analysis and total RNA expression profiling.

c) Tertiary data analysis.

This final step involves the interpretation of genetic variation to provide knowledge and insights into biology and the causes of disease and how to treat or prevent them.

Further information can be found at

2.2.3. Variant identification and analysis.

Genetic variation of an individual determines the characteristics of an organism and is influenced by both genetic makeup and the environmental an individual is in. This is the main cause of the differences between members of the same species and is also a cause of multiple diseases. Although germline mutations easily identifiable and follow distinct lines of heredity, some mutations (somatic) follow much weaker relationships and are difficult to identify.
<table>
<thead>
<tr>
<th>Reference</th>
<th>ACTGACGCATGCATCATGCATGC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SNP</strong></td>
<td>ACTGACGCATGCATCAT*TCATGC</td>
</tr>
</tbody>
</table>

a) SNPs are a result of the substitution of a single base-pair. Here we have a transversion event substituting a T in place of a G.

<table>
<thead>
<tr>
<th>Reference</th>
<th>ACTGACGCATGCATCATGCATGC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insertion</strong></td>
<td>ACTGACGCATG GTA CATCATGCATGC</td>
</tr>
<tr>
<td><strong>Deletion</strong></td>
<td>ACTGACG--TGCATCATGCATGC</td>
</tr>
</tbody>
</table>

b) Indels affect a string of base-pairs. This example shows an insertion of string GTA, and a deletion of CA string.

### Structural Variation

- **Deletion**
- **Insertion**
- **Inversion**
- **Duplication**
- **Copy Number Variation**

c) Graphic showing the five types of structural variants.
Figure 4: a) SNPs result from the substitution of a single base-pair. Here we have a transversion event substituting a T in place of a G. b) Indels affect a string of base-pairs. In this example, the insertion string shows that GTA has been inserted, and the deletion string shows a deletion of CA. c) The five most common types of structural variants. Source: European Molecular Biology Laboratory.

a) Germline mutations.

A gene variation in a body's reproductive cell (egg or sperm) that becomes incorporated into the DNA of every cell in the body of the offspring. Germline mutations are passed on from parents to offspring. Also called hereditary mutation.

b) Somatic mutations.

Unlike germline variants, somatic variants occur within the genome but are not inherited. Identification of somatic variants enables the identification of variant hotspots. These hotspots can be used to study significant genes and pathways that can then be used in predictive, prognostic, remission and metastatic analysis of cancer. These somatic variant hotspots can also be used as therapeutic targets (Vijayan, Yiu, & Zhang, 2017).
c) Mutations calling.

Both germline and somatic Single Nucleotide Variants (SNVs) and small Insertions and Deletions (Indels) are common drivers of cancer growth and development. Therefore, accurately detecting these mutations is paramount to the secondary analysis in cancer research. However, the challenge and complexity of NGS data analysis lie in the heterogeneous nature of tumor samples, in addition to the cross-contamination between tumor and matched normal samples (Fang et al., 2015).

Furthermore, since variants are much rarer on the genomic scale, their detection requires very sensitive algorithms. The accurate detection of mutations is further complicated by errors due to amplification biases introduced during sample libraries preparations, machine errors during sequencing, software errors and mapping artifacts introduced during alignment.

2.3. The Genome Alignment Toolkit.

Genome analysis pipeline identifies germline and somatic variants within both WXS and WGS data by comparing allele frequencies in normal and tumor sample alignments with the reference genome, annotating each mutation, and aggregating mutations from variant callers into one Variant Call Format (VCF) file.
These pipelines start with a reference alignment step followed by co-cleaning to increase the alignment quality.

The Genome Analysis Toolkit (GATK) Best Practices provide step by step recommendations for performing variant discovery analysis in NGS data. There are several different GATK Best Practices workflows tailored to particular applications depending on the type of variation of interest and the technology employed. The Best Practices documentation attempts to describe in detail the key principles of the processing and analysis steps required to go from raw reads coming off the sequencing machine, all the way to an appropriately filtered VCFs that can be used in downstream analyses.

2.3.1 Data preprocessing.

The first phase in all WGS secondary data analysis involves pre-processing the raw sequence data (provided mostly in FASTQ) to produce analysis ready BAM files. This involves alignment to a reference genome as well as some data cleanup operations to correct for technical biases and make the data suitable for analysis. The main steps include:
a) Mapping to the Reference Genome.

This first processing step is performed per-read group and consists of mapping each individual read pair to the reference genome, which is a synthetic single-stranded representation of common genome sequence that is intended to provide a common coordinate framework for all genomic analysis.

The mapping algorithm processes each read pair in isolation, thus this could be parallelized to increase throughput. The tools involved include the Burrows Wheeler Aligner (BWA) (Li & Durbin, 2009) and Samtools (Li et al., 2009).

b) Marking Duplicates.

This second processing step is performed per-sample. It identifies read pairs that are likely to have originated from duplicates of the same original DNA fragments through some artifactual processes. These are considered to be non-independent observations, so the program tags all but of the read pairs within each set of duplicates, causing them to be ignored by default during the variant discovery process.

This step constitutes a major bottleneck since it involves making a large number of comparisons between all the read pairs belonging to the sample, across all of its read groups. Another bottleneck occurs in the sorting operation that operates across all reads belonging to the sample.
The algorithms involved in both steps continue to be the target of optimization efforts to reduce their impact on latency. The tools involved include Picard Tools (McKenna et al., 2010).

c) Base (Quality Score) Recalibration.

This third processing step is performed per-sample and consists of applying machine learning to detect and correct for patterns of systematic errors in the base quality scores, which are confidence scores emitted by the sequencer for each base. Base quality scores play an important role in weighing the evidence for or against possible variant alleles during the variant discovery process, so it's important to correct any systematic bias observed in the data.

The recalibration procedure involves collecting covariate statistics from all base calls in the dataset, building a model from those statistics, and applying base quality adjustments to the dataset based on the resulting model.

Finally, the recalibration rules derived from the model are applied to the original dataset to produce a recalibrated dataset. This is parallelized in the same way as the initial statistics collection, over genomic regions, then followed by a final file merge operation to produce a single analysis-ready file per sample. The tools involved include GATK (McKenna et al., 2010).
2.3.2. Variant Discovery.

Here analysis-ready BAM files are processed for variant calls. This involves identifying genomic variation in one or more individuals and applying filtering methods appropriate to the experimental design. The output is typically in VCF format although some classes of variants (such as CNVs) are difficult to represent in VCF and may, therefore, be represented in other structured text-based formats.

This is parallelized over genomic regions, then followed by a final file merge operation to produce a single VCF file per sample. The tools involved include MuTect2 (McKenna et al., 2010).

2.3.3. Call set Refinement.

Depending on the application, additional steps such as filtering and annotation may be required to produce a call set ready for downstream genetic analysis. This typically involves using resources of known variation, truth sets and other metadata to assess and improve the accuracy of the results as well as attach additional information.

The tools involved include SnpEff (Cingolani et al., 2012) and ANNOVAR (Wang, Li, & Hakonarson, 2010) annotate variants based on their genomic locations.
It also predicts coding effects such as synonymous or nonsynonymous amino acid replacement, starts codon gains or losses, stops codon gains or losses, or frameshifts. Predicted effects are with respect to protein-coding genes. Variants affecting non-coding genes are annotated and the corresponding bio-type is identified.

Recently published method for analyzing genomics data using genome graphs was shown to processes one whole-genome sequencing sample in 6.5 hours using a system with 36 CPU cores (Rakocevic et al., 2019). Since this method iteratively augments genome graphs of thousands of individuals to yield incremental gains in variant calling accuracy, the size of such data grows exponentially and could only be suitable for a framework with elastic and computational storage.

Several genomic analysis platforms like SevenBridges, FireCloud, Cancer Genomics Cloud (CGC) (Lau et al., 2017) are all cloud-native with SevenBridges offering a multi-cloud environment where a user could choose between Amazon Web Services, Google Cloud Platform or Alibaba Cloud.

For instance, the CGC enables researchers to rapidly access and collaborate on massive public cancer genomic datasets, including The Cancer Genome Atlas (Cancer Genome Atlas Research Network et al., 2013). It provides secure on-demand access to data, analysis tools, and computing resources. Researchers from diverse backgrounds can easily visualize, query, and explore cancer genomic datasets visually or programmatically.
In this chapter, I discuss the computational framework that is mostly behind such platforms and how they have been able to provide one-stop-shops for genomic analysis.


Cloud computing is the on-demand delivery of computing power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay as you go pricing.

Cloud computing provides a simple way to access servers, storage, databases and a broad set of application services over the internet. A cloud services platform such as AWS owns and maintains the network-connected hardware required for these application services, while you provision and use what you need via a web application.

3.1.1. Types of cloud deployments.

Determining the cloud computing architecture that developers will deploy applications is usually the first and most important step in cloud computing. Currently, there are three different cloud deployment services i.e. on a public cloud, private cloud and hybrid cloud.
a) Public Cloud.

Public clouds are owned and operated by a third party cloud service providers, which deliver their computing resources like servers and storage over the internet. The cloud provider manages all hardware, software, and other supporting infrastructure. Examples include Amazon Web Services, Google Cloud Platform, IBM Bluemix, and Microsoft Azure.

b) Private Cloud.

A private cloud refers to cloud computing resources used exclusively by a single business or organization. A private cloud can be physically located on the company's on-site datacenter. Some companies also pay third-party service providers to host their private cloud. A private cloud is one in which the services and infrastructure are maintained on a private network.

c) Hybrid Cloud.

Hybrid clouds combine public and private clouds, bound together by technology that allows data and applications to be shared between them. By allowing data and applications to move between private and public clouds, a hybrid cloud gives your business greater flexibility, more deployment options, and helps optimize your existing infrastructure, security, and compliance.

As cloud computing has grown in popularity, several different models and deployment strategies have emerged to help meet the specific needs of different users. Each type of cloud service and deployment method provides different levels of control, flexibility, and management.

a) Infrastructure as a Service (IaaS).

Contains the basic building blocks for cloud IT and typically provide access to networking, computers (virtual or on dedicated hardware), and data storage space with the highest level of flexibility and management control.

b) Platform as a Service (PaaS).

Removes the need for the management of underlying infrastructure (usually hardware and operating systems) and allows users to focus on the deployment and management of applications.

c) Software as a Service (SaaS).

Provides a completed product that is run and managed by the service provider and users do not have to think about how the service is maintained or how the underlying infrastructure is managed.

- Trade capital expense for variable expense: users don’t have to invest heavily in data centers and servers beforehand. Users pay only when they consume computing resources.

- Economies of scale: usage from hundreds of thousands of customers are aggregated in the cloud which translates into lower pay-as-you-go prices.

- Eliminates guessing about capacity: users can access as much or as little capacity as needed, and scale up and down as required with almost no notice.

- Increase speed and agility: new IT resources are only a click away, which means that users can reduce the time to make resources available from weeks to just minutes.

- Cheap to set up and maintain: cloud computing lets users focus on production rather than on running and maintaining data centers.

- Go global in minutes: users can deploy applications in multiple regions around the world with just a few clicks. This means that they can provide lower latency and a better experience at minimal cost.
3.2. Amazon Web Services.

Since 2016, Amazon Web Services (AWS) has been offering Information Technology (IT) infrastructure services to businesses in the form of web services - now commonly known as cloud computing. When using AWS, developers are provided an opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with demand.

Today, AWS provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. With a broad set of infrastructure services, currently over 140, I discuss services that will be used in the thesis in this section.

3.2.1. Global Infrastructure.

a) Regions.

A Region is an isolated geographical location in the world where we have multiple Availability Zones. Each Region is designed to be completely isolated from the other Regions. The AWS Cloud consists of 20 geographic Regions around the world, which announced four more Regions.
b) Availability Zones.

An Availability Zone consists of one or more isolated discrete data centers. Availability Zone is designed as an independent failure zone and are physically separated within a typical metropolitan region and are located in lower-risk flood plains.

With redundant power supply, networking, and connectivity Availability Zones offer the ability to operate applications that are highly available, fault tolerant, and scalable than would be possible from a single data center. The AWS operates from 60 Availability Zones around the world, with announced plans for 12 more Availability Zones.

3.2.2. Elastic Compute Cloud (EC2).

Elastic Compute Cloud (EC2) is a web service that provides resizable compute capacity in the cloud that makes web-scale computing easier for developers. EC2 reduces the time required to obtain and boot new server instances (called EC2 instances) to minutes, allowing you to quickly scale capacity, both up and down, as computing requirements change.

EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.
EC2 provides developers and system administrators the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Figure 5: The AWS Global Infrastructure. The Cloud spans 60 Availability Zones within 20 geographic regions around the world, with announced plans for 12 more Availability Zones and four more Regions in Bahrain, Cape Town, Hong Kong, and Milan. Source: Overview of Amazon Web Services, AWS Whitepaper.

AWS provides developers the ability to increase or decrease compute capacity depending on the demands of an application and hence frees a developer from the costs and complexities of planning, purchasing, and maintaining hardware.
This, in turn, transforms what are commonly large fixed costs into much smaller variable costs. There are three types of EC2 instances:

a) **On-Demand Instances.**

On-Demand instances provide users with the option to pay for computing capacity by the hour with no long term commitments.

b) **Reserved Instances.**

Unlike On-Demand Instances, Reserved Instances provide developers a capacity reservation and offers up to 75% discount compared to On-Demand instance pricing.

c) **Spot Instances.**

Spot Instances allow developers to bid on spare EC2 computing capacity. Since Spot instances are available at a discount compared to On-Demand pricing, they could help significantly reduce the cost of running and growing an application’s compute capacity while maintaining a low budget.
3.2.3. EC2 Auto Scaling.

EC2 Auto Scaling is a service that enables a developer to maintain an application’s availability and allows automatic addition or removal of EC2 instances according to predefined conditions. The fleet management features of EC2 Auto Scaling to maintain the health and availability of EC2 instances fleet will be used in this thesis.

EC2 Auto Scaling provides two autoscaling mechanisms; dynamic scaling which response to changing demand and predictive scaling automatically schedules the right number of EC2 instances based on predicted demand. These two mechanisms could be applied together to scale faster.

3.2.4. Elastic Container Registry (ECR).

Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images. ECR is integrated with Elastic Container Service (ECS), simplifying development to production workflow.

ECR eliminates the need to operate separate container repositories or worry about scaling the underlying infrastructure.
ECR hosts images in a highly available and scalable architecture, allowing users to reliably deploy containers for applications.

3.2.5. Elastic Container Service (ECS).

Elastic Container Service (ECS) is a highly scalable, high-performance container orchestration service that supports Docker containers and allows developers to easily run and scale containerized applications on AWS. ECS eliminates the need for developers to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.

3.2.6. Batch.

Batch enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. Batch dynamically provisions the optimal quantity and type of computer resources (e.g., CPU or memory-optimized instances) based on the volume and specific resource requirements of the batch jobs submitted.

Batch eliminates the need to install and manage batch computing software or server clusters, allowing developers to focus on analyzing results and solving problems. Batch orchestrates batch computing jobs across the full range of compute services and features, such as EC2 and Spot Instances.
3.2.7. Simple Storage Service (S3).

Simple Storage Service (S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases.

S3 provides easy to use management features so developers can organize data and configure finely tuned access controls to meet specific business, organizational, and compliance requirements. S3 is designed for 99.999999999% of durability, and stores data for millions of applications for companies all around the world.

3.3. Virtualization and Containers.

Virtualization is the process of creating a software-based or virtual representation of virtual applications, servers, storage, and networks. It is the single most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses.

Some benefits of virtualization include increased IT agility, flexibility, and scalability with significant cost savings. Greater workload mobility, increased
performance and availability of resources, automated operations make it simpler to manage and less costly to own and operate.

Since virtualization technology has serious drawbacks, such as performance degradation due to the heavyweight nature of Virtual Machines (VM), the lack of application portability, slowness in the provisioning of IT resources e.t.c. The IT industry has been steadily embarking on a Docker inspired containerization journey since the Docker initiative was specifically designed to make the containerization paradigm easier to grasp and use.

3.2.1. Containers.

A container is a software bucket comprising everything necessary to run the software independently. There can be multiple containers in a single machine that are completely isolated from one another as well as from the host machine.

Docker enables the containerization process to be accomplished in a risk-free and accelerated fashion. A Docker container includes a software component along with all of its dependencies: binaries, libraries, configuration files, scripts, jars e.t.c. Containers eliminate the friction that comes with shipping code to distant locations. Docker also lets us test the code and then deploy it in production as fast as possible.
Figure 6: Containers and virtual machines. A container runs natively on Linux and shares the kernel of the host machine with other containers while a virtual machine (VM) runs a full-blown operating system.

<table>
<thead>
<tr>
<th>Virtual Machines (VMs)</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Represents hardware-level virtualization.</td>
<td>Represents operating system virtualization.</td>
</tr>
<tr>
<td>Heavyweight.</td>
<td>Lightweight.</td>
</tr>
<tr>
<td>Slow provisioning.</td>
<td>Real-time provisioning and scalability.</td>
</tr>
<tr>
<td>Fully isolated and hence more secure.</td>
<td>Process-level isolation and hence less secure.</td>
</tr>
</tbody>
</table>

Table 1: Comparison between virtual machines and containers. (Learning Docker: Optimize the power of Docker to run your applications quickly and easily by Pethuru Raj, Jeeva Chelladurai and Vinod Singh.)
3.2.2. Advantages of Containers.

- Conflicting dependencies: with containers, users can run applications with conflicting dependencies easily by running each application in a separate container.
- Missing dependencies: installing applications in a new environment is fast with containers since all dependencies are packaged along with the application in a container.
- Platform differences: moving from one operating system to another is no longer a problem. If both systems run Docker, the same container will execute without issues.

3.2.3. Docker.

Docker (Merkel & Dirk, 2014) is an open source containerization engine, which automates the packaging, shipping, and deployment of any software applications that are presented as lightweight, portable, and self-sufficient containers, that will run virtually anywhere. The Docker lets users quickly assemble composite, enterprise scale, and business-critical applications.
Docker's client-server architecture allows the client to talk to the daemon, which is the workhorse for building, running, and distributing Docker containers. The Docker client and daemon usually run on the same system, but a client could be connected to a remote daemon. The Docker client and daemon interact with each other using a REST API over UNIX sockets or a network interface.

a) The Docker daemon.

The Docker daemon listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.
b) The Docker client.

The Docker client is the primary way that users interact with Docker. When a developer uses commands such as `docker run`, the client sends these commands to the Docker daemon, which carries them out. The `docker` command uses the Docker API. The Docker client can communicate with more than one daemon.

c) Docker registries.

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry. When a developer pulls or runs docker images, the required images are pulled from your configured registry. When a developer publishes the docker image, the `docker push` command pushes the image to the configured registry.
Raw FASTQ files from sequencing machines are not meaningful in research. Therefore, secondary analysis multi-step workflows should be designed to take raw reads in FASTQ format and process them to identify the variation in a biological sample compared to a standard genome reference. The pipeline follows the following steps:

a) Mapping of raw reads onto a standard genome reference using an alignment algorithm e.g. Smith-Waterman Algorithm (Waterman, Smith, & Beyer, 1976), Burrows-Wheeler Transform (Li & Durbin, 2009) or Genomic Graphs (Rakocevic et al., 2019).

b) Perform variant calling i.e. identify the differences between the mapped reads and the reference genome. The most common tools include MuTect2 (Cibulskis et al., 2013), Varscan2 (Koboldt et al., 2012) and Strelka (Saunders et al., 2012).

c) Annotate the variants by combining with other information and databases to identify genomic variants highly correlated with a disease or drug response. Examples include Annovar (Wang et al., 2010) and Snpeff (Cingolani et al., 2012).
Although it might be cheap to run such pipelines in local clusters, the cloud provides flexibility, scalability and fine-grained documented details for running computational jobs e.g. real costs and computational metrics. Likewise, in the long run, the use of Batch and spot instances could lead to further reduced costs than using a High-Performance Cluster (HPC). I, therefore, set up the project to test whether the use of Batch and compared various run time of our pipeline on different EC2 instance configurations. Likewise, I compared the costs of running the pipeline under various EC2 instance configurations.

In this section, I will discuss the various aspects of the project and show how to integrate bioinformatics pipelines with Batch and seamlessly deploy and execute the pipeline using data from the 1000 Genome Project.

4.1. Project Description.

4.1.1. Docker Containers.

Docker containers have recently gained much traction in bioinformatics. This is due to the fact that most of the bioinformatics software is Linux based and have multiple dependencies, therefore packaging such tools with all the dependencies in a single Docker container saves a lot of time in deployment, update and reuse of such tools.
The example below shows a Dockerfile that was used to develop the image

noelnamai/bwa:0.7.12. The file contains all of the commands that you use to
package BWA into a Docker container. For this project, all Dockerfiles are

```dockerfile
# specifying the base image
FROM ubuntu:16.04

# name and email of the person who maintains the file
MAINTAINER Noel Namai noelnamai@yahoo.com

# updating ubuntu and installing other necessary software
RUN apt-get clean all \
    && apt-get -y update \
    && apt-get install -y \n        bwa \n    && apt-get clean

# chmod of /opt/ and change working directory
RUN chmod -R 777 /opt/
WORKDIR /opt/

# specify the command executed when the container is started
CMD ["/bin/bash"]
```

The following Docker containers were built and used in this project, and are
publicly available from Docker hub:
<table>
<thead>
<tr>
<th>Container</th>
<th>Dockerfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>noelnamai/fastqc:0.11.3</td>
<td><a href="https://github.com/noelnamai/thesis/blob/master/docker/fastqc/Dockerfile">https://github.com/noelnamai/thesis/blob/master/docker/fastqc/Dockerfile</a></td>
</tr>
</tbody>
</table>

Table 3: Docker containers were built and used in this project, and are publicly available from Docker hub

### 4.1.2. The Cloud infrastructure.

The pipelines are built of multiple steps (jobs) that take a set of input files and process them with computer-intensive algorithms and then passes the outputs into subsequent steps for analysis. These steps are linked to building a flexible and robust genomics processing workflow.
In this project, the bioinformatics jobs are orchestrated as a workflow using Nextflow (Di Tommaso et al., 2017). By packing these jobs in Docker containers (Merkel & Dirk, 2014), we easily use the power of Batch to scale. The further ability to use EC2 Spot Instances leads to optimization on cost since we could bid on spare EC2 compute capacity, which can save up to 90% off of traditional On-Demand prices.

Figure 8: The AWS infrastructure that the pipeline employs to run jobs in Batch using ECS, Spot Instances and Docker containers.
The various steps to set up a working environment on Batch are defined in the Supplementary materials section, however, a complete working environment depends on various resources and permissions:

a) IAM roles that provide service permissions.
b) An environment that launches and terminates compute resources for jobs.
c) A custom Amazon Machine Image (AMI).
d) A job queue to submit jobs and to schedule the appropriate resources within the computing environment.
e) Job definitions that define how to execute an application.

Batch dynamically scales the compute resources in response to the number of runnable jobs in the job queue and also manages job placement. Therefore, the state machine will be composed of smaller batches that submit a job to Batch, and then poll the Batch API for the job status. If the job status equals SUCCESS, it proceeds to log the job status and continues to the next step. If the job status equals FAILED, it will end the execution of the state machine.

4.1.3. Data sources.

Apart from the raw FASTQ files, other files are provided as a resource bundle from The Broad Institute site located at ftp://ftp.broadinstitute.org/bundle/hg19.
a) The GATK Resource bundle.

The GATK resource bundle is a standardized collection of files for working with human sequencing data with the GATK. The Broad Institute provides several versions of the bundle corresponding to the various reference builds. Currently, the Broad Institute is transitioning to support the Grch38/hg38 reference build, but all files necessary have not yet generated e.g. the hg38 version of the Broad's exome intervals. This led to the choice of complete hg19 reference build.

b) CEU samples.

The samples used are from 61 reference families that were contributed by The Centre d'Etude du Polymorphisme Humain (CEPH), Foundation towards the 1000 genome project. The CEPH collection includes a total of 809 individuals in the reference families living in Utah with Northern and Western European ancestry. This panel is mostly used when a common reference data set is of value e.g. when characterizing DNA polymorphisms and in constructing the human genetic map (Dausset et al., 1990).

4.1.4. Software requirements.

Transforming FASTQ files to BAM files is computationally intensive and requires a lot of computing memory and power. A minimum of 2 GB of memory is required, but 4 GB to 8 GB are recommended when working with the human genome (Van der Auwera et al., 2013).
<table>
<thead>
<tr>
<th>Sample</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRR764692</td>
<td>1 GB</td>
</tr>
<tr>
<td>SRR742200</td>
<td>2 GB</td>
</tr>
<tr>
<td>SRR701475</td>
<td>3 GB</td>
</tr>
<tr>
<td>ERR034518</td>
<td>4 GB</td>
</tr>
<tr>
<td>ERR034524</td>
<td>5 GB</td>
</tr>
<tr>
<td>ERR034548</td>
<td>6 GB</td>
</tr>
<tr>
<td>SRR099452</td>
<td>7 GB</td>
</tr>
<tr>
<td>SRR107050</td>
<td>8 GB</td>
</tr>
<tr>
<td>SRR107047</td>
<td>9 GB</td>
</tr>
<tr>
<td>SRR107048</td>
<td>10 GB</td>
</tr>
</tbody>
</table>

Table 4: FASTQs from 10 individuals from the Utah residents (CEPH) with Northern and Western European ancestry.

The following packages are required to run the pipeline:

- Trimmomatic (Bolger, Lohse, & Usadel, 2014).
- Burrows-Wheeler Aligner (BWA) (Li & Durbin, 2009).
- SAMtools (Li et al., 2009).
- Picard Tools (McKenna et al., 2010).
- Genome Analysis Toolkit (GATK) (McKenna et al., 2010).
- Annovar (Wang et al., 2010).
Figure 9: The core processing steps that are necessary to transform raw sequencing data into VCF using GATK.
1. Trimmomatic.

Perform a variety of useful trimming tasks for Illumina data using Trimmomatic.

Input: Raw FASTQ files.

Output: Trimmed raw FASTQ files.

```
$ java -jar trimmomatic-0.38.jar PE \
    -phred33 \ 
    read_1.fq.gz read_2.fq.gz \ 
    trimmed.paired.1.fq.gz trimmed.unpaired.1.fq.gz \ 
    trimmed.paired.2.fq.gz trimmed.unpaired.2.fq.gz \ 
    ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 \ 
    LEADING:3 \ 
    TRAILING:3 \ 
    SLIDINGWINDOW:4:15 \ 
    MINLEN:36
```

This step removes adapters (ILLUMINACLIP:TruSeq3-PE.fa:2:30:10), leading low quality or N bases below quality 3 (LEADING:3), trailing low quality or N bases below quality 3 (TRAILING:3); scans the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15 (SLIDINGWINDOW:4:15) and drops reads below the 36 bases long (MINLEN:36).

2. BWA-MEM.

Generate a SAM file containing aligned reads using BWA.

Input: Trimmed raw FASTQ files.

Output: SAM file.
This step creates an aligned SAM file which contains reads from all input FASTQ files combined and aligned to the same reference. The Burrows-Wheeler Aligner is based on a backward search with the Burrows-Wheeler Transform and efficiently aligns short sequencing reads against a large reference sequence while allowing mismatches and gaps (Li & Durbin, 2009).

3. Samtools View.

Converts the SAM output from BWA to BAM format using Samtools.

Input: SAM file.

Output: BAM file.

```
$ samtools view -bS reads.sam > aligned.bam
```

Although SAM files contain aligned reads from BWA, it is text-based and thus the user must first convert the SAM to binary format i.e. BAM which is much easier for computer programs to work with but not human readable.

4. Picard AddOrReplaceReadGroups.

This step uses Picard tools to add a group to the raw BAM file. Many tools in the GATK require the presence of at least one read group (RG) tag to which each read can be assigned.
Input: BAM file.

Output: BAM file with the reading group added.

```
$ java -jar picard.jar AddOrReplaceReadGroups \\
  I=aligned.bam \\
  O=aligned.grouped.sorted.bam \\
  SO=coordinate \\
  RGLB=sample \\
  RGPL=illumina \\
  RGPU=sample \\
  RGSM=sample
```

This step AddOrReplaceReadGroups, creates a new BAM file with the same information as the input BAM file, except that the new BAM file will have read group information attached.

5. Picard MarkDuplicates.

This step uses Picard tools to mark duplicate reads and creates a binary index file i.e. BAI from the BAM file.

Input: BAM file.

Output: BAM file with duplicate reads marked as duplicates.

```
$ java -jar picard.jar MarkDuplicates \\
  I=aligned.grouped.sorted.bam \\
  O=aligned.grouped.sorted.deduplicated.bam \\
  METRICS_FILE=output.metrics \\
  VALIDATION_STRINGENCY=SILENT \\
  CREATE_INDEX=true
```

49.
During sequencing, the same DNA molecules could be sequenced multiple times thus resulting in duplicate reads which should not be counted as additional evidence for or against a putative variant. This step identifies these reads as duplicates so that other GATK tools could ignore them.

This creates a new BAM file with the same content as the input BAM file, except that any duplicate reads are marked as duplicates.

6. GATK RealignerTargetCreator.

This step creates a target list of intervals to be realigned. Many alignment artifacts result in many bases mismatching the reference near the misalignment that is easily mistaken as SNPs. This step determines suspicious intervals which are likely in need of realignment.

Input: BAM file.

Output: A list of known indel sites that will be used as targets for realignment.

```
$ java -jar GenomeAnalysisTK.jar \
  -T RealignerTargetCreator \ 
  -nt cpus \ 
  -R genome.fa \ 
  -I aligned.grouped.sorted.deduplicated.bam \ 
  -known mills.indel.sites.vcf \ 
  -known known.indel.sites.vcf \ 
  -o output.intervals.list
```

The file created, output.intervals.list contains the list of intervals that the program identified as needing realignment within the genome.
The list of known indel sites is used as targets for realignment.

7. GATK IndelRealigner.

This step uses GATK to perform local realignment of reads around indels over intervals determined by the RealignerTargetCreator.

Input: BAM file.

Output: BAM file with better local alignments in the regions that were realigned.

```
$ java -jar /usr/GenomeAnalysisTK.jar \
  -T IndelRealigner \ 
  -R genome.fa \ 
  -I aligned.grouped.sorted.deduplicated.bam \ 
  -known mills.indel.sites.vcf \ 
  -known known.indel.sites.vcf \ 
  -targetIntervals output.intervals.list \ 
  -o aligned.grouped.sorted.deduplicated.realigned.bam
```

This creates a new BAM file containing all the original reads, but with better local alignments in the regions that were realigned.

8. GATK BaseRecalibrator.

This step uses GATK to detect systematic errors in base quality scores.

Inputs: BAM file.

Output: The recalibration table by base quality score.
Variant calling algorithms rely heavily on the quality scores assigned to the individual base calls in each sequence read. The per-base estimates of error emitted by the sequencing machines are subject to various sources of systematic technical error which could lead to over or underestimated base quality scores in the data.

Base Quality Score Recalibration (BQSR) is a process in which we apply machine learning to model these errors empirically and adjust the quality scores accordingly. This allows us to get more accurate base qualities, which in turn improves the accuracy of the variant calls.

9. GATK PrintReads.

This step uses GATK to write out sequence read data (for filtering, merging, subsetting, etc).
Input: BAM file.

Output: A processed BAM file.

```
java -jar /usr/GenomeAnalysisTK.jar \
    -T PrintReads \
    -nct cpus \
    -R genome.fa \
    -I aligned.grouped.sorted.deduplicated.realigned.bam \
    -BQSR output.data.table \
    -o processed.bam
```

This is a generic utility for manipulating sequencing data in BAM format and it can dynamically merge the contents of multiple input BAM files, resulting in merged output sorted by coordinates. This creates a new BAM file containing all the original reads, but with more accurate base substitution, insertion and deletion quality scores. The original quality scores are usually discarded in order to keep the file size down.

10. GATK HaplotypeCaller.

GATK HaplotypeCaller calls Germline SNPs and Indels via local re-assembly of haplotypes.

Input: BAM file.

Output: VCF file.
The GATK HaplotypeCaller performs the following steps:

a) It determines which regions of the genome it needs to operate on.

b) The program then builds a De Bruijn graph to reassemble the active regions and identifies what are the possible haplotypes present in the data. The program then realigns each haplotype against the reference haplotype using the Smith-Waterman algorithm in order to identify potentially variant sites.

c) Then for each active region, the HaplotypeCaller performs a pairwise alignment of each read against each haplotype using the PairHMM algorithm. This produces a matrix of likelihoods of haplotypes given the read data. These likelihoods are then marginalized to obtain the likelihoods of alleles for each potentially variant site given the read data.

d) In the last step, the HaplotypeCaller applies Bayes' rule on each potentially variant site, using the likelihoods of alleles given the read data to calculate the likelihoods of each genotype per sample given the read data observed for that sample. The most likely genotype is then assigned to the sample.
Whenever HaplotypeCaller encounters a region with variation, it discards the existing mapping information and completely reassembles the reads in that region. This increases the accuracy of the HaplotypeCaller when calling regions that are traditionally difficult to call.

11. GATK VariantFiltration.

This step filters variant calls based on INFO and FORMAT annotations.

Input: VCF file.

Output: VCF file with filtered variants.

```
$ java -jar /usr/GenomeAnalysisTK.jar \
   -T VariantFiltration \
   -R genome.fa \
   -V sample.sorted.vcf \
   -window 35 \
   -cluster 3 \
   -filterName FS \
   -filter "FS > 30.0" \
   -filterName QD \
   -filter "QD < 2.0" \
   -o sample.sorted.filtered.vcf
```

The GATK VariantFiltration performs hard-filtering on variant calls based on the criteria provided by the user.

The final step functionally annotates genetic variants detected from diverse genomes using ANNOVAR. Using a desktop computer, ANNOVAR requires approximately 4 min to perform gene-based annotation and approximately 15 min to perform variants reduction on 4.7 million variants (Wang et al., 2010).

Input: VCF file.

Output: VCF file with annotations.

```
$ perl /annovar/table_annovar.pl sample.sorted.filtered.vcf 
    annovar/humandb \
    -buildver hg19 \
    -out sample.annovar \
    -protocol refGene \
    -operation g \ 
    -nastring . \ 
    -vcfinput \ 
    --thread cpus \ 
    --maxgenethread cpus \ 
    -polish \ 
    -otherinfo
```

ANNOVAR performs gene-based annotation by identifying whether the variant causes protein-coding changes and the amino acids that are affected.
4.2. Pipeline Implementation.

The pipeline was implemented using Nextflow (Di Tommaso et al., 2017). This pipeline could be in different ways but was implemented especially to run parallel jobs as distributed tasks in multiple instances using the AWS Batch queuing engine. This workflow makes use of POSIX compatibility shared file system. We put together the tools and software discussed in section 4.1.4 as a reliable file system that tasks share with each other the input and output files as they are produced in a distributed cluster of computers.

The cloud parameters e.g. the VM image ID, VPC subnet ID, Security Groups and the instance type that the pipeline will use is specified in the configuration file (Supplementary Material 11).

```
cloud {
    userName = "ec2-user"
    instanceType = "t2.micro"
    keyName = "bioinformatics"
    subnetId = "subnet-4ea1cf2a"
    securityGroup = "sg-0e611108c17231471"
    imageId = "ami-0e93d03da78fa14a4"
}
```

It is important to note that Nextflow requires a Linux image that provides support for Cloud-init mechanism and has a Java runtime (version 8) and a Docker engine (version 1.11 or later) installed.
Credentials should be defined in the working environment using the standard variables. The credentials should not be specified in the Nextflow configuration file due to security reason.

AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_DEFAULT_REGION

Each process specifies the following:

a) Atleast one Docker container by using the `container` directive.

b) All the container images to be used must be published in a Docker registry such as Docker Hub or a Container Registry that can be reached by ECS Batch.

c) The batch executor must be specified in the pipeline configuration.

d) Atleast one batch queues for pipeline execution must be specified using the `queue` directive. Batch queues allow you to bind the execution of a process to a specific computing environment that is; number of CPUs, type of instances (On-demand or Spot), scaling ability, etc.

e) All container image must include the Command Line Interface tool or it could be installed in a custom AMI image.

The complete pipeline is available at

4.3. Results and Discussion.

To validate the pipeline, I used the developed Germline Mutation Calling (Figure 9) as suggested by the GATK Best Practices (Van der Auwera et al., 2013) and as implemented on (Figure 8) to generate germline variant calls from raw FASTQs from 10 individuals from the Utah residents (CEPH) with Northern and Western European ancestry.

Samples were selected with varying sample sizes ranging from 1 GB to 10 GB. Three General Purpose instances were selected since they provide a balance of computing, memory, and network resources. These instances use a custom Intel Xeon E5-2676 v3 Haswell processor optimized specifically for EC2. They run at a base clock rate of 2.4 GHz and can go as high as 3.0 GHz with Intel Turbo Boost. The specs as provided by Amazon are shown below:

<table>
<thead>
<tr>
<th>Instance</th>
<th>vCPUs</th>
<th>RAM</th>
<th>Network Performance</th>
<th>EBS Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>m4.large</td>
<td>2</td>
<td>8 GiB</td>
<td>Moderate</td>
<td>450 Mbps</td>
</tr>
<tr>
<td>m4.xlarge</td>
<td>4</td>
<td>16 GiB</td>
<td>High</td>
<td>750 Mbps</td>
</tr>
<tr>
<td>m4.2xlarge</td>
<td>8</td>
<td>32 GiB</td>
<td>High</td>
<td>1000 Mbps</td>
</tr>
</tbody>
</table>

Table 5: Three General Purpose instances were selected since they provide a balance of computing, memory, and network resources.
With Enhanced Networking, these instances deliver up to 4 times the packet rate of instances without Enhanced Networking, while ensuring consistent latency, even when under high network I/O. While using placement groups, Enhanced Networking could reduce average latencies between instances by 50%. The M4 instances are EBS Optimized by default, with additional, dedicated network capacity for I/O operations. The instances support 64-bit HVM AMIs launched within a VPC.

4.3.1. Reduced run time.

The use of 8 vCPUs and 32 GiB of memory provided the fastest computational time as compared to instances with 2 vCPUs 8 GiB of memory, 4 vCPUs 16 GiB and a local deployment. All the pipeline processes used all the vCPUs in the instances and the memory requirement was set to 95% of the available memory.

This suggests that faster computational times could be achieved using larger EC2 instances with more vCPUs and large memory as shown in Figure 10.
Figure 10: Runtime comparison for the local cloud deployment of the pipeline.

Instances with 2 vCPUs 8 GiB of memory, 4 vCPUs 16 GiB of memory and 8 vCPUs and 32 GiB were used.

4.3.2. Reduced computational costs.

A total of 26 spot instances with 2 vCPUs 8 GiB of memory, 24 spot instances with 4 vCPUs 16 GiB of memory and 26 spot instances with 8 vCPUs and 32 GiB of memory were used respectively to run the pipeline in AWS Batch for all the samples. The use of instances with 8 vCPUs and 32 GiB resulted in the lowest costs overall for both On-demand and Spot Instances.
This could be attributed to short run times of the pipeline using while using 8 vCPUs or the ability to bid for much lower price percentage as compared to the other instances.

A hybrid solution could surface since some individual tools like Trimmomatic and Fastqc do not require much computational power and could easily be run on smaller instances with 2 vCPUs, while computationally intensive processes like BWA and the Haplotypecaller could be processed on larger instances. Overall, Spot Instances provided a cheaper solution compared to on-demand Instances.
Figure 11: Cost comparison for the cloud deployment of the pipeline. A total of 26 spot instances with 8 vCPUs and 32 GiB of memory provided an optimal price as compared to the other instances.
4.4. Future Work.

4.4.1. Kubernetes.

Kubernetes is an open source platform for automating deployments, scaling, and operations of application containers across clusters of hosts, providing a platform for container-centric infrastructure. Kubernetes ensures that container orchestration can scale properly, mask failures and maximizes system resource.

When scaling containers in a large scale production environment, developers are usually tasked to develop containers scheduling systems, monitoring of containers during run time, designing auto-scaling and load balancing systems for the container instances and determining if a zero downtime is required for deployment.

Therefore, the automated coordination of these systems and processes is very important but could be very complicated. However, automation tools like Kubernetes could help developers achieve these goals.

Kubernetes started as an open source software project in 2014 by Google and was build upon Google’s internal cluster management system. Kubernetes key features include:
- Automated deployment and replication of containers.
- Online scale-in and scale-out of container clusters.
- Load balancing over groups of containers.
- Rolling upgrades of application containers.
- Resiliency, with the automated rescheduling of failed containers i.e. self-healing of container instance.
- Controlled exposure of network ports to systems outside of the cluster.

Unlike centralized orchestration systems, Kubernetes uses a combination of microservices and small control loops by combining the effects of separate, autonomous entities that collaborate. Kubernetes helps manage the high complexity of systems that express, analyze, and use system dependencies.

When deploying the Kubernetes, large scale bioinformatics workloads could benefit as follows:

a) The Kubernetes API uses similar information to all objects and thus makes writing generic tools that work across all objects much simpler, which in turn enables the development of consistent user experience.

b) The Kubernetes actions are based on observation rather than the state diagram and thus container jobs are robust to failures and perturbations. Therefore, when a controller fails or restarts it simply picks up where it left off. This is important when implementing job retries after failure.
4.4.2. GATK 4.0.

Earlier versions of GATK were focused on germline short variant discovery, which played a central role in NGS data analysis. GATK 4.0 includes both well established pipelines and new tools that take advantage of the latest developments in machine learning and neural networks algorithms.

GATK 4.0 adds somatic short variant calling with Mutect2 (Cibulskis et al., 2013), which combines somatic modeling algorithm Mutect with the haplotype-centric logic of the germline caller, HaplotypeCaller (Poplin et al., 2017). GATK 4.0 adds full discovery pipeline capabilities for both germline and somatic CNV and adds somatic allelic CNV based estimation of tumor heterogeneity. Also, GATK 4.0 is engineered to scale seamlessly from gene panels and exomes to WGS datasets.

GATK 4.0 has been extensively optimized for performance, flexibility, speed, and scalability, and includes end-to-end pipeline scripts that can be run on any local or cloud computing infrastructure. Furthermore, several features will be beneficial for both the development and analysis of NGS pipelines and data. For instance, consolidated pre-processing for all variant discovery, detection of Germline SNPs & Indels and the use of a neural network approach to germline short variant filtering will improve the analysis of NGS data.
The performance and of NGS pipelines will be greatly improve due to the use of engineering features like parallelization by Apache Spark (Zaharia et al., 2016), cloud deployment support, hardware-optimized versions of bottleneck algorithms and the use of GenomicsDB datastore for germline joint-calling.

Lastly, the availability of official Docker images, having Picard tools bundled in GATK 4.0 will improve the convenience of the use of the GATK Best Practices pipelines out of the box.

4.5. Conclusions.

I have a great belief that computational biology remains one of the main areas of science that will benefit greatly from cloud computing. Several features such as the renting on a pay-as-you-go basis of computational resources, storage and bandwidth; the convenience of renting and paying for the exact resources needed; and convenience of not owning or maintaining the resources will remain attractive.

Since there has been an exponential increase in the amount of sequencing data, a sustained increase in both volume and variety of genomics data is imminent. Ideas like personalized medicine will generate massive amounts of data in terms of Single Cell Sequencing, Digital Imaging, Transcriptomics, Proteomics, and Metabolomics. The study of the Epigenome and the Microbiome will generate
over four-fold amounts of data as compared to NGS sequencing (Topol, 2014).

Cloud computing will, therefore, become an important part of future efforts to analyze and host large archives of sequencing data that could be used in multiple collaborations. Such efforts will benefit from the ability to hold data in a single location, near to the computational infrastructure that will be used to analyze it.

Additionally, the ability to easily configure security measures needed to adhere to privacy standards will be of much benefit to personalized medicine and scientific research. Research institutions are beginning to allocate funds and migrate computational workloads to the cloud due to the increasing popularity of the cloud and its advantages in the scientific community.

Likewise, the cloud provides a playground for the scientific community to develop and standards for evaluating the accuracy of new analytical technologies, such as variant callers. The value of NGS data is wholly dependent not only upon accurate analysis and identification of mutations but also upon low cost and reproducible methods of interpretation and the translation of the resulting data-driven insights into the clinic.
To achieve this, scientists must develop, explore, and compare alternative computational and evaluation methods in order to formulate standards of the best possible quality.
Supplementary Material.

Step 1: Sign up for a new AWS account.

A user can sign up at https://portal.aws.amazon.com/billing/signup#/start for an AWS account.

Step 2: Create a Virtual Private Cloud.

a) From the Amazon VPC console at https://console.aws.amazon.com/vpc/, click on Launch VPC Wizard.

b) Select VPC with Public and Private Subnets, and then choose Select.

c) On the VPC with Public and Private Subnets page, give the VPC a name, keep the default options, and then choosing to Create VPC.

Step 3: Create a Security Group.

a) In the navigation pane of the Amazon VPC Management console, select Security Groups and then click on Create Security Group.

b) Type a name and a description for your security group in the corresponding boxes. In the VPC box, choose the identifier for the Amazon VPC created before.
Step 4: Creating an Elastic File System (EFS).

a) From the Amazon EFS Management Console at https://console.aws.amazon.com/efs/, select Create File System.

b) Select the VPC from the VPC list. Under Create mount targets.

c) Select the checkboxes for all of the Availability Zones with the default subnets, automatic IP addresses, and the default security groups chosen.

d) Name the file system, keeping General Purpose and Bursting selected as the default performance and throughput modes.

e) Choose Create File System.

Step 5: Creating an Amazon Machine Image (AMI).


b) From the Community AMIs, search Amazon Linux AMI 2.0. The Amazon ECS-optimized Amazon Linux 2 AMI is the recommended AMI to use for launching your Amazon ECS container instances. Click Select.

c) Select a t2.micro instance for the instance type since the instance type does not impact the AMI. Select Next: Configure Instance Details.

d) Use the default configurations of the Instance Details. Select Next: Add Storage.
e) Docker images and file sizes in Bioinformatics could vary a lot, therefore the developer should choose a larger EBS storage in the storage configuration that would fit the data expected. Select Next: Add Tags.

f) Tags are optional since the Compute Environment will create the tags when launching instances. Select Next: Configure Security Group.

g) Select the Security Group that was created before. Make sure that the security group accepts SSH connection at port 22 and NFS connection at port 2049, then select Review and Launch.

h) The user can review all the settings in the next window and then select Launch.

i) Select an existing key pair or create a new one and then select Launch Instances.

j) Once the instance is launched and running, the user can connect to the instance by typing the following in the console:

```
$ ssh -i /home/noelnamai/bioinformatics.pem
e2-user@ec2-35-168-23-14.compute-1.amazonaws.com
```

k) The user should update the kernel and install awscli, docker, ecs, java and nextflow.
$ sudo yum update -y
$ sudo yum install -y python python-pip awscli
$ sudo pip install --upgrade awscli
$ sudo amazon-linux-extras install docker
$ sudo service docker start
$ sudo usermod -a -G docker ec2-user
$ sudo amazon-linux-extras disable docker
$ sudo amazon-linux-extras install -y ecs
$ sudo systemctl enable --now ecs
$ sudo yum install -y java-1.8.0
$ curl -s https://get.nextflow.io | bash
$ sudo mv nextflow /usr/bin/

l) Once complete, from the EC2 Dashboard, select **Actions → Image then Create Image.**
m) Give the image a name and description and then select **Create Image.**

**Step 6: Creating an IAM Group.**

a) Navigate to IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/) and under **Groups** select **Create New Group.**

b) Type the name of the group in the **Group Name** box, and then click **Next Step.**

c) Attach the following permissions to the new group:

- **AmazonEC2FullAccess, AmazonS3FullAccess,** and **AWSBatchFullAccess.** This will give users under this group permission to run jobs on AWS Batch.
d) The user can review the permissions and group settings and then select
   **Create Group.**

**Step 7: Creating an Identity and Access Management User.**

a) Navigate to the IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/) and select **Users** and then choose **Add user.**

b) Give the user name for the new user. Give the user **Programmatic access** to allow access to the API, AWS CLI, or Tools for Windows PowerShell. This also creates an **Access Key** for each new user. Select **Next: Permissions.**

c) Select **Add user to group** and select the group created above.

d) In the next screens, select **Next: Tags** and then **Create User.**

**Step 8: Creating Identity and Access Management (IAM) Service Roles.**

The AWS Batch needs permissions to run/stop EC2 instances. This can be done by creating the required roles in IAM. These are managed as separate roles in IAM under **Roles** and some of these are created automatically when a Batch job is set up for the first time, but we can make sure that these are already present and configured properly beforehand.
The following service roles are required:

- ecsInstanceRole.
- AWSBatchServiceRole.
- AWSServiceRoleForEC2SpotFleet.
- AmazonEC2SpotFleetTaggingRole.

Below I show how to create the AWSBatchServiceRole. The other roles follow the same procedure to be created.

a) Navigate to the IAM console at https://console.aws.amazon.com/iam/ and select **Create Roles** under **Roles**.

b) Under the type of trusted entity, choose **Batch**. Choose the service that will use this role, select **Batch**.

c) Choose Next: Permissions, Next: Review. For Role Name, type **AWSBatchServiceRole** and choose to **Create Role**.

d) The summary should be as shown below.

**Step 9: Setting up a Compute Environment.**

a) From the AWS Batch console at https://console.aws.amazon.com/batch/, select the region to use and then from **Compute environments** select **Create environment**.

76.
b) For the **Compute environment type** select **Managed**. Assign a unique name for your to compute environment under **Compute environment name**. Select AWSBatchServiceRole and ecsInstanceRole for **Service role** and **Instance role** respectively.

c) Select an existing Amazon EC2 key pair to associate with the instance at launch. Select **Spot** under Provisioning model and set the maximum price at 40%. Select AmazonEC2SpotFleetTaggingRole under **Spot fleet role**.

d) Select **Minimum vCPUs**: 0, **Desired vCPUs**: 2 and **Maximum vCPUs**: 256. This will allow all EC2 instances to be shut down if no job is available.

Check **Enable user-specified Ami ID** and then select the AMI image ID previously created.

e) Select the VPC previously created under **VPC id** and then select which subnets in the selected VPC should host your instances. Add a key and value under **EC2 tags** and then click on **Create**.

**Step 10: Setting up a Job Queue.**

a) Navigate to the AWS Batch console at

[https://console.aws.amazon.com/batch/](https://console.aws.amazon.com/batch/) and select **Job queues** and then click on **Create queue**.
b) Give the queue a name under **Queue name** check **Enable job queue** so that the job queue can accept job submissions. Set **Priority** to 1 and then select the previously created compute environment.

**Step 11: Nextflow Configuration.**

Next, the user has to specify the following in the configuration file:

```yaml
params {
    results          = "s3://noelnamai/results"
    genome_fasta     = "s3://noelnamai/hg19/ucsc.hg19.fasta"
    genome_fasta_sa  = "s3://noelnamai/hg19/ucsc.hg19.fasta.sa"
    genome_fasta_fai = "s3://noelnamai/hg19/ucsc.hg19.fasta.fai"
    genome_fasta_bwt = "s3://noelnamai/hg19/ucsc.hg19.fasta.bwt"
    genome_fasta_ann = "s3://noelnamai/hg19/ucsc.hg19.fasta.ann"
    genome_fasta_amb = "s3://noelnamai/hg19/ucsc.hg19.fasta.amb"
    genome_fasta_pac = "s3://noelnamai/hg19/ucsc.hg19.fasta.pac"
    Genome_fasta_dict = "s3://noelnamai/hg19/ucsc.hg19.dict"
    known_dbsnps_file = "s3://noelnamai/hg19/dbsnp_138.hg19.vcf"
    known_indel_file  = "s3://noelnamai/hg19/1000G_phase1.indels.hg19.sites.vcf"
    known_dbsnps_1000_file = "s3://noelnamai/hg19/1000G_phase1.snps.high_confidence.hg19.sites.vcf"
    mills_indel_file = "s3://noelnamai/hg19/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf"
}
docker {
    enabled = true
}
remove = true
runOptions = '-u $(id -u):$(id -g)'
}

process {
    executor = "awsbatch"
    queue = "job-queue"
}

executor {
    awscli = "/home/ec2-user/miniconda/bin/aws"
}

aws {
    region = "us-east-1"
}

cloud {
    instanceType = "t2.micro"
    userName = "ec2-user"
    subnetId = "subnet-4ea1cf2a"
    securityGroup = "sg-0e611108c17231471"
    imageId = "ami-0e93d03da78fa14a4"
}

report {
    enabled = true
    file = "$params.results/report.html"
}

timeline {
    enabled = true
    file = "$params.results/timeline.html"
}

workflow.onComplete = {
    println "Completed at : $workflow.complete"
    println "Duration : $workflow.duration"
    println "Execution status : "${workflow.success ? 'OK': 'failed'}"
}
Step 12: Running jobs with AWS Batch.

To run the pipeline, type the following in the console. Make sure Nextflow is installed in the local computer and that the AWS credentials have been set up.

```
$ nextflow run main.nf -w s3://noelnamai/work/
```

```
N E X T F L O W  ~  version 18.10.1
Launching `main.nf` [loving_lovelace] - revision: 163a81e822

GERMLINE VARIANT DISCOVERY (SNVs + INDELS)
============================================
Start       : Wed Feb 06 01:50:33 GMT 2019
BWA         : BWA 0.7.15
Annovar     : Annovar 4.18
Samtools    : Samtools 1.9
Picard      : Picard 2.18.15
Trimmomatic : Trimmomatic 0.38
GATK        : GenomeAnalysisTK 3.8-0

[warm up] executor > awsbatch
[75/266d90] Submitted process > trimmomatic (ERR034518)
[ac/f64413] Submitted process > fastqc (ERR034518)
[cf/bcbe97] Submitted process > bwa_mem (ERR034518)
[13/9f4401] Submitted process > samtools_view (ERR034518)
[18/d61a41] Submitted process > picard_add_or_replace_read_groups (ERR034518)
[1d/d003ae] Submitted process > samtools_flagstat (ERR034518)
[7f/70edaf] Submitted process > picard_mark_duplicates (ERR034518)
[b8/c594f7] Submitted process > gatk_realigner_target_creator (ERR034518)
[2d/ff9dd8] Submitted process > gatk_indel_realignment (ERR034518)
```
Submitted process > gatk_base_recalibrator (ERR034518)
Submitted process > gatk_print_reads (ERR034518)
Submitted process > haplotype_caller (ERR034518)
Submitted process > picard_sort_vcf (ERR034518)
Submitted process > gatk_variant_filtration (ERR034518)
Submitted process > annovar (ERR034518)
Completed at : Wed Feb 06 10:36:12 GMT 2019
Duration : 8h 45m 39s
Execution status : OK
Definition of Terms.

Amazon Elastic Compute Cloud (EC2): a web service offered by AWS that provides scalable computing capacity in the cloud.

Amazon Simple Storage Service (S3): a web service offered by AWS that provides storage through web services interfaces.

Amazon Web Services (AWS): a collection of remote computing services that together make up a cloud computing platform offered over the Internet by Amazon.com.

Amazon Elastic Container Registry (ECR): a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.

Amazon Elastic Container Service (ECS): a highly scalable, high-performance container orchestration service that supports Docker containers and allows developers to easily run and scale containerized applications on AWS.

Copy Number Variation (CNV): when the number of copies of a particular gene varies from one individual to the next. It has long been recognized that some cancers are associated with elevated copy numbers of particular genes.
Deoxyribonucleic Acid (DNA): a molecule called deoxyribonucleic acid (DNA), which contains the biological instructions that make each species unique. The four types of nitrogen bases found in nucleotides are adenine (A), thymine (T), guanine (G) and cytosine (C). The order, or sequence, of these bases, determines what biological instructions are contained in a strand of DNA.

DNA Sequencing: the process of determining the order of the four chemical building blocks (bases) that make up the DNA molecule. The sequence tells scientists the kind of genetic information that is carried in a particular DNA segment.

Genome Analysis Toolkit (GATK): a structured programming framework designed to ease the development of efficient and robust analysis tools for NGS data using the functional programming philosophy of MapReduce.

Genome-Wide Association Studies (GWAS): an approach that involves rapidly scanning markers across the complete sets of genomes of many people to find genetic variations associated with a particular disease.
Human Genome Project (HGP): an international research effort to sequence and map all of the genes (together known as the genome) of members of our species, Homo sapiens that was completed in April 2003.

Next Generation Sequencing (NGS): also known as High-Throughput Sequencing, is the catch-all term used to describe a number of different modern sequencing technologies that allow us to sequence DNA and RNA much more quickly and cheaply than the previously used Sanger sequencing.

The Variant Call Format (VCF): a text file format (most likely stored in a compressed manner) that contains meta-information lines, a header line, and the data lines each containing information about a position in the genome. The format also has the ability to contain genotype information on samples for each position.

Whole Exome Sequencing (WXS): a laboratory process that is used to determine the nucleotide sequence primarily of the exonic (or protein-coding) regions of an individual’s genome and related sequences, representing approximately 1% of the complete DNA sequence.
Whole Genome Sequencing (WGS): a laboratory process that is used to determine nearly all of the approximately 3 billion nucleotides of an individual’s complete DNA sequence, including non-coding sequence.

AWS Identity and Access Management (IAM): a web service that helps administrators to securely control access to AWS resources. IAM is used to control who is authenticated (signed in) and authorized (has permissions) to use resources.

Amazon Machine Image (AMI): provides the information required to launch an instance, which is a virtual server in the cloud. Multiple instances can be launched from a single AMI when multiple instances with the same configuration are needed.

The Human Genome Project (HGP): an initiative which was led at the National Institutes of Health (NIH) which produced a very high-quality version of the human genome sequence that is freely available in public databases. That international project was successfully completed in April 2003, under budget and more than two years ahead of schedule.
The Centre d'Etude du Polymorphisme Humain (CEPH): a foundation in Jean Dausset, Paris, France that contributed its collection of 61 reference families to the 1000 genomes project. The CEPH collection includes families collected by R. White (Utah), J. Dausset (French), J. Gusella (Venezuelan), and J. Egeland (Amish). There are a total of 809 individuals accounting for 832 pedigree positions in the reference families.
References.


Sulem, P., Helgason, H., Oddson, A., Stefansson, H., Gudjonsson, S. A., Zink,


