System for Interpretation of Personal Genomes

by

Marc Fiume

A thesis submitted in conformity with the requirements for the degree of Doctorate of Philosophy

Department of Computer Science
University of Toronto

Copyright © 2015 by Marc Fiume
Abstract

System for Interpretation of Personal Genomes

Marc Fiume
Doctorate of Philosophy
Graduate Department of Computer Science
University of Toronto
2015

Genomics is undergoing a revolution sparked by higher throughput and cost effective DNA sequencing technologies. Sequencing has become a ubiquitous tool with varied inputs, scopes, techniques, technologies, and purposes.

There is potential for the development of software systems that assist in translating raw sequence data into actionable information that helps improve disease assessment, detection, and treatment so that individuals who are or may be affected by genetic conditions are treated with an unprecedented level of precision and predictiveness. This new kind of medicine informed by personal genomic interpretation promises to have immense medical and economic benefits.

Despite the capacity of new sequencing technologies to generate huge volumes of raw sequence data, it remains a substantial informatics challenge to efficiently analyze it. HTS technologies produce data at a rate that exceeds Moore's Law, creating enormous technical and usability issues.

Freely available tools that are both powerful enough to be efficient and user-friendly enough to be used by genomic researchers without informatics expertise are scarce. Notwithstanding a few exceptions, users are often forced to choose between powerful, specialized software that needs to be run on the command-line or alternatives that are less specialized but are graphical and user-friendly.
This thesis presents two software platforms that combine techniques from various domains of computer science, most notably data structures, databases, algorithm design, data visualization, user interface design, and user experience design, that together form a highly integrated system for interpretation of personal genomes that is both powerful and easy to use.
Acknowledgements

I am grateful to many people who have helped me complete this work. Much gratitude is owed to my supervisor, Michael Brudno, who has mentored me for many years: in the last year of my undergraduate degree and throughout my graduate work. His friendship, enthusiasm, and scientific rigour are infectious qualities that create a very enjoyable environment to work in. I am especially thankful for his support of my passions, without which this would not have been possible.

Gary Bader and Ravin Balakrishnan, two additional members of my committee, have also had tremendous effect. They provided me with expert, insightful, and pragmatic advice throughout the process that undoubtedly improved the direction and outcomes of these projects.

I have had the pleasure of working with many bright colleagues within the department. I am grateful for collaborations that evolved into friendships with Misko Dzamba, Orion Buske, James Vlasblom, Eric Smith, Andrew Brook, Marta Girdea, Sergiu Dumitriu and many others that I have worked with on the Savant and MedSavant projects.

I am honoured to be a student of the Department of Computer Science at University of Toronto. It sets a high bar for professionalism, integrity, excellence, and collegiality that makes for a dynamic and inspiring atmosphere to work in. Relationships cultivated in the lecture rooms of the Bahen Centre and on the ice at McCormick Arena have enriched my life as a student at this school.

I am very appreciative of my uncle, Eugene Fiume, who served as a compass in navigating my career as a student at the university. He initially guided me into the Bioinformatics program, which fostered my interests in computer science and genetics, and has continued giving wise advise in the years since.

I’m indebted most of all to my immediate family. My father Mike and mother Mary have gone to extraordinary lengths to support me throughout my graduate studies, and have always been amazingly understanding of sacrifices of time and energy needed to complete the
graduate program. Finally, my wife Andrea, who shares wholeheartedly with me on this journey, receives my utmost gratitude, admiration, and love.
for Daniel Pfaff
# Contents

1  Background  
   1.1  Genetics  
      1.1.1  Coding DNA  
      1.1.2  Noncoding DNA  
   1.2  Genome Evolution  
   1.3  Genetic Variation  
      1.3.1  Single Nucleotide Variants  
      1.3.2  Indels  
      1.3.3  Structural Variants  
   1.4  The Human Genome  
   1.5  Genetic Disease  
      1.5.1  Classes of Genetic Diseases  
      1.5.2  Inheritance Patterns  
      1.5.3  Heritability of Mutations  
   1.6  DNA Sequencing  
   1.7  High Throughput Sequencing Data & Analysis  
      1.7.1  Sequencing Scope  
      1.7.2  Basecalling  
      1.7.3  Paired Sequencing  
      1.7.4  Alignment  
      1.7.5  Genotyping  
      1.7.6  Applications of Genotyping  
   1.8  Genomic Medicine  
      1.8.1  Challenges in Enabling Genomic Medicine  
   1.9  Thesis Objectives  

2  Visualizing Personal Genomes  
   2.1  Related Work  
      2.1.1  Information Visualization  
      2.1.2  Genome Browsers & Other Information Visualization Tools  
   2.2  Savant Genome Browser  
      2.2.1  Features and Design Principles  
      2.2.2  Versions  
      2.2.3  Programming Language  
      2.2.4  Files and Formatting  
      2.2.5  Datasources  
      2.2.6  Architecture  
      2.2.7  User Interface  
      2.2.8  Visual Encodings for Genomic Data  
      2.2.9  Plugins  
      2.2.10  Evaluation  
   2.3  Summary
### 3 Interpreting Personal Genomes

#### 3.1 Related Work
- 3.1.1 Variant Visualization 83
- 3.1.2 Variant Search 85

#### 3.2 MedSavant
- 3.2.1 Features and Design Principles 94
- 3.2.2 Versions 94
- 3.2.3 Programming Language 96
- 3.2.4 Architecture 97
- 3.2.5 Server 99
- 3.2.6 High Performance Database 101
- 3.2.7 Client 111
- 3.2.8 Apps 115
- 3.2.9 Evaluation 130

#### 3.3 Summary 134

### 4 Conclusions and Future Work

#### 4.1 Summary 137

#### 4.2 Limitations & Future Work
- 4.2.1 Use in Secure Environments 138
- 4.2.2 Protecting Personal Health Information 139
- 4.2.3 Installation Support 140
- 4.2.4 Compatibility and Reproducibility 141
- 4.2.5 Scalability 141

#### Bibliography 143
1 Background
1.1 Genetics

Genetics is a discipline that studies the composition, inheritance, and variation of genomes: the genetic material within living organisms. A genome is a complex and elegant system that is comprised of a set of discrete DNA molecules called chromosomes. The genome guides the development and maintenance of cells within an individual throughout its life. DNA is described by the sequential arrangement of four basic molecular building blocks called nucleotides: they are Adenine, Thymine, Cytosine, and Guanine (abbreviated A,T,C,G, respectively). These are arranged in a complementary fashion so that the sequence on one strand of DNA can be inferred by the other. The nucleotides are the alphabet for describing the manufacturing of proteins, physical molecules that do a wide array of molecular functions. Proteins are encoded by regions in the genome called genes, which are expressed through serial processes of transcription — which creates intermediary RNA copies of genes’ DNA sequences — and translation — which transforms RNA molecules into proteins. The process of translation is aptly named as it involves the conversion of nucleic acid sequence, which are read in triplets called codons, into amino acid sequence via the Genetic Code. The relationship between DNA, RNA, and protein and the processes by which they are converted between each other form what is known as the Central Dogma of Molecular Biology diagramed in Figure 1.1.

1.1.1 Coding DNA

Genes are comprised mainly of exons and introns, the latter being segments that are transcribed into RNA but are removed before being translated into protein. Because exons encode protein elements, they are called coding DNA. The set of all exons is collectively referred to as the exome.

![Figure 1.1 Central Dogma of Molecular Biology.](image)
1.1.2 Noncoding DNA
Not all DNA within a genome encodes protein and these segments are referred to as noncoding DNA. In general, noncoding DNA tends to be less conserved than coding DNA as will be described in the sections that follow. The function of noncoding DNA — formerly referred to by some as “junk DNA” — is gradually being appreciated, for example, in understanding its role in regulating transcription and protecting from chromosomal deterioration [1].

1.2 Genome Evolution
The success of DNA in producing a world full of complex and diverse species is owing its abilities to (i) instruct the building of molecular machinery in the form proteins using the Genetic Code, (ii) self-replicate, and most importantly, (iii) evolve by natural selection. Natural selection is a law of nature whereby a characteristic that confers a selective advantage to its bearer — an advantage that helps one to survive and reproduce — tends to be conserved over generations, while a characteristic that confers a selective disadvantage does not. While natural selection operates on observable characteristics of organisms — i.e. phenotypes — it effectively operates on genomic characteristics — i.e. genotypes — whose subtle variations give rise to the observable differences between individuals. Genetic variation therefore serves as the raw material for evolution to happen.

1.3 Genetic Variation
Genetic variation refers to differences between genomes, both between species and within species. While there is a significant amount of evidence to show that meaningful differences are encoded in the physical structure of genomes [2, 3], genomic variation is almost universally described with respect to differences in the linear sequence of nucleotides using a chromosomal coordinate system that will be described later.

1.3.1 Single Nucleotide Variants
Single Nucleotide Variants, or SNVs, represent a class of variation where genomes differ in the nucleotide at a single position. The term allele is one of a number of alternative forms of a gene or genetic locus. Technically an allele should be called a polymorphism only when it occurs above some frequency within a population however the term Single Nucleotide Polymorphism
(abbreviated as SNP, pronounced “snip”) is often used synonymously with SNV regardless of the allele’s frequency. SNVs can be caused by misincorporation of nucleotides during DNA replication.

### 1.3.2 Indels

Insertions and deletions describe events where an additional sequence is present in one genome with respect to another. In the absence of the progenitor sequence it is impossible to know whether the additional sequence was actually inserted into one genome versus deleted from the other. To reflect this ambiguity insertion and deletion events are commonly referred to as indels. Indels can be caused by slippage of the machinery involved in DNA replication, for example.

### 1.3.3 Structural Variants

While SNVs and small indels do not tend to affect the physical structure of a genome, structural variants do. These include large indels, duplications, inversions, and translocations. The size beyond which indels are classified as being “structural” is typically 1,000 basepairs although this choice is rather arbitrary. Duplications occur when DNA segments are repeated in the genome. Inversions represent chromosomal rearrangements where a continuous stretch of DNA is reversed end-to-end. Translocations involve the movement of genetic material within or between chromosomes.

Structural variants like translocations and inversions are referred to as balanced events, because they do not result in a net gain or loss of genomic sequence, but others like duplications lend to what are known as Copy Number Variations, or CNVs, which occur where the number of times a sequence is present varies between genomes.

The basic types of genetic variation are illustrated in Figure 1.2. It is important to appreciate that the several types of genetic variation are caused by distinct mechanisms, and so they are not homogeneously distributed across the genome and occur at different frequencies. In general, changes that are large or in coding regions of the genome are more disruptive and therefore less tolerated by natural selection than small changes or ones in noncoding regions.

### 1.4 The Human Genome

The human genome is one of the most studied genomes. It is made up of 22 pairs of homologous chromosomes and 2 additional sex chromosomes, X and Y. Human cells are diploid, meaning
that they two copies of each chromosome. The sex chromosomes are slightly exceptional since females inherit two homologous X chromosomes but males inherit only one X chromosome, in addition to one Y. The cumulative length of the chromosomes in a human genome is over 3 billion basepairs, making the combined number of nucleotides from the set of all homologous pairs exceed 6 billion. A karyotype of a male human genome is shown in Figure 1.3.

The human genome contains over 20,000 genes, which constitute approximately 2% of the total genetic material. The remaining 98% of the genome is noncoding, and is highly repetitive. It is estimated that over 40% of the human genome is made up of repetitive DNA elements (e.g. LINEs and SINEs, long and short interspersed elements) [4]. The remainder of this thesis refers mainly to human genomes and diseases, unless specified otherwise.
1.5 Genetic Disease

While genetic variants can confer selective advantages they can also cause marked deleterious effects, resulting in genetic disease. In cancer for example, genetic alterations disable a cell's ability to regulate itself thereby resulting in antagonistic tumor growth. The ways in which genetic variants often contribute to disease are now discussed.

1.5.1 Classes of Genetic Diseases

Due to the vastness and complexity of genomes, there is a countless number of unique events that can cause its malfunctioning, ultimately resulting in disease. However, genetic diseases can be classified by their causes as described below:

**MONOGENETIC DISORDERS**

Monogenic disorders are conditions caused by mutations in a single gene. Sickle cell disease and cystic fibrosis are examples of monogenic disorders, caused by harmful mutations in the genes HBB and CFTR, respectively.

**MULTIFACTORIAL INHERITANCE DISORDERS**

Multifactorial disorders are caused by a combination of genetic variations in multiple genes and are often influenced by external environmental factors. Type 2 diabetes and most cancers are examples of multifactorial disorders.

**CHROMOSOME DISORDERS**

Chromosome disorders are caused by structural or copy number abnormalities in the genome. These are typically the result of copy number variants or aneuploidy, a condition where the chromosome number is different from the norm. Down syndrome is an example of a chromosome
disorder, which occurs when a genome includes three copies of chromosome 21 (called trisomy 21) despite no individual gene on these copies being abnormal [5]. Prader-Willi syndrome, on the other hand, can be caused by copy number variation of a specific group of genes within chromosome 15 [6]. Copy number alterations that reduce the abundance of regulatory or oncogenes are often found in cancer genomes, which are notoriously unstable [7-9].

1.5.2 Inheritance Patterns
Redundancy makes genomes robust against changes that can result in disease. For example, the complementarity of nucleotide base pairing makes single-stranded DNA damage repair straightforward; redundancy in the Genetic Code ameliorates the affects of SNVs or mistakes in translation. In some cases, the redundancy provided by having multiple homologous chromosomes allows for an organism to survive even if not all forms of an allele are functional.

The outcome of individuals with a disease-causing allele is determined by its inheritance pattern. There are five main inheritance patterns for monogenic disorders which are explained below. Pioneering work in identifying these inheritance patterns was done by Gregor Mendel in the 1800s, and so they are often called Mendelian Inheritance Patterns. They describe whether the gene responsible for the disease is located in an autosomal region (i.e. non-sex determining chromosomes) or on sex chromosomes (i.e. on the X or Y chromosome) and whether the condition is manifest in individuals having one mutant allele, called a dominant disorder, or if it requires all alleles to be dysfunctional, called a recessive disorder.

**AUTOSOMAL DOMINANT**
Autosomal Dominant conditions are expressed in individuals that have at least one copy of the mutant allele, where the allele is located within an autosomal region. Generally, males and females have equal probabilities of being affected by an autosomal dominant disorder and transmitting it to their offspring. If only one parent has the mutant allele (and by virtue of dominance is affected) the probability of his or her offspring inheriting the mutant allele and being affected is 50%. Huntington’s disease, a neurodegenerative disease, follows the autosomal dominant inheritance pattern [10]. However, epigenetic mechanisms like genetic imprinting — a process whereby certain genes are silenced or expressed in a manner that is specific to the parent of origin — or de novo CNV events can alter disease outcomes from classic model of Mendelian Inheritance Patterns as is the case in Angelman syndrome [11, 12].
AUTOSOMAL RECESSIVE

Autosomal Recessive conditions are manifest only in individuals having mutant alleles on both copies of a genetic locus, where the locus is located within an autosomal region. Recessive conditions can result from an individual being homozygous at a single position or having heterozygous mutations at different positions, called compound heterozygotes. Like with autosomal dominant disorders, there is no bias between sexes, though the probability of the disease phenotype being observed is reduced. If only one parent is a carrier, i.e. has the mutant allele but is unaffected by virtue of recessiveness, there is no possibility of his or her offspring being affected. Only when both parents carry a mutant allele can the disease be observed in progeny. Cystic Fibrosis, the most common fatal genetic disease affecting children and young adults in Canada, is an autosomal recessive disorder [13].

X-LINKED DOMINANT

X-linked Dominant disorders affect individuals that have have at least one copy of the mutant allele, where the allele is located on the X chromosome. Because males have only one chromosome X and one chromosome Y, daughters of fathers affected by X-linked dominant disorders must be affected while sons cannot be (unless they receive a mutant allele from the maternal chromosome X). Fragile X syndrome, the most common inherited cause of intellectual disability among boys, follows the X-linked dominant inheritance pattern [14].

X-LINKED RECESSIVE

X-linked Recessive disorders affect individuals that have mutant alleles on all copies, where the allele is located on the X chromosome. Because men have only one X chromosome, males with the mutant allele are affected. As with X-linked dominant disorders, there is no father to son transmission, but there is father to daughter and mother to son and daughter transmission. Color blindness, which is most common among men, is an X-linked Recessive disorder [15].

Y-LINKED

Y-linked disorders are caused by mutations on chromosome Y, and therefore affect males only. Y-linked disorders are rare, as they often confer male infertility [16] and cannot therefore be transmitted. Defects on chromosome Y have also been associated with Retinitis Pigmentosa [17].

Figure 1.4 illustrates the transmission and expression of most of these types of inheritance patterns.
Figure 1.4 Transmission and expression of Mendelian disorders. Downloaded from http://ghr.nlm.nih.gov/handbook/
1.5.3 Heritability of Mutations
The heritability of a genetic mutation is dependent upon which cells in the body it occurs in. Germline mutations occur in germ cells, i.e. sperm and eggs, and can be transmitted to offspring. Somatic mutations occur in somatic cells, i.e. non-germ cells, and cannot be transmitted to descendants. Somatic mutations are acquired over the lifetime of an individual, and can be induced by lifestyle or environmental factors such as smoking, exposure to radiation, or toxins.

1.6 DNA Sequencing
Genomic medicine attempts to predict and ameliorate the effects of genomic disease. A first step in prevention or treatment of genomic diseases is understanding the causal relationship between genotypes and phenotypes. While phenotypes are directly observable, and can be easily collected by a trained physician, genotype information is more difficult to collect, requiring molecular assays such as DNA sequencing.

DNA molecules have complex three-dimensional structures that cannot be disentangled and read completely from start to finish in a linear fashion. Instead, technologies have been developed to read the sequence of nucleotides in DNA molecules piecewise, relying on overlapping sequences to inform the assembly of the pieces.

The traditional approach to sequencing fragments was developed by Frederick Sanger in 1977, based on the differential speed at which DNA molecules which are randomly terminated during polymerisation migrate within an electrophoretic gel. Sanger sequencing is accurate but expensive. The Human Genome Project is perhaps the most celebrated project to sequence and assemble a genome to near completion. The project — which relied heavily on Sanger sequencing — ended in 2003, 13 years after it began, and cost over $3B USD [18].

Over the last decade new methodologies have emerged that have increased the speed and efficiency of DNA sequencing. Most of these so-called High Throughput Sequencing (HTS) technologies operate by capturing, using high-speed photography, the sequences of colour emitted by fluorescently labeled nucleotide reagents when they are incorporated in DNA polymerization. Popular HTS technology vendors include Illumina, Roche, Helicos, Complete Genomics, Pacific Biosciences, and Life Technologies.
The pace of improvement in throughput and economy of HTS technologies has exceeded Moore's law over the past decade, due to the commercialization of significant technological improvements made from 2008 to 2012, as shown in Figure 1.5 [19]. In January 2014, just more than a decade since the completion of the Human Genome Project, Illumina announced the HiSeq X system which claims to be the first DNA sequencing platform to be able to deliver whole genome sequencing of a human individual at a cost of $1,000 in consumables [20]. Such advances in DNA sequencing technologies have enabled the acquisition of genetic information from thousands of genomes and in doing so have catalyzed growth in the field of genomic medicine, described later.

1.7 High Throughput Sequencing Data & Analysis
Sequencing has become a ubiquitous tool with varied inputs (e.g. DNA, RNA), scopes (e.g. whole genome sequencing, whole exome sequencing), techniques (e.g. shotgun sequencing, paired sequencing), technologies (e.g. single base sequencing, dibase sequencing), and purposes (e.g. genome assembly, gene expression analysis, genotyping). Figure 1.6 provides an overview of processes involved in genotyping, i.e. compiling a list of genetic variants that an individual
possesses, one of the most important workflows in genomic medicine. The subsections that follow describe essential concepts in this and closely related workflows in HTS data analysis.

### 1.7.1 Sequencing Scope

Sequencing can be performed on whole genomes, referred to as whole genome sequencing (WGS), or in select regions, called targeted sequencing. The targeted approach involves the use of molecular probes specially designed to physically capture regions of interest. Captured sequences are subsequently amplified and sequenced. A common approach is to perform targeted sequencing of exomes which is known as whole exome sequencing (WES). Preconfigured kits for targeted enrichment of the exome are commercially available [21]. Because the exome represents a small proportion of the genome, WES has traditionally been less expensive than WGS despite the additional overhead needed for the targeted capture.

### 1.7.2 Basecalling

Limitations in current sequencing technologies prohibit the reading of long contiguous pieces of the genome. Instead, short fragments called reads — which range in length from about 50 basepairs to 1,000 basepairs depending on the technology used — are sequenced at a time. The process of determining the identity of individual nucleotides in a read, called basecalling, involves signal processing and is sometimes erroneous. A quality score is assigned to each base called that is related to the error and is typically computed as a Phred Score:

\[ Q = -10 \log_{10} P_{\text{miscall}} \]

where \( Q \) represents the quality value and \( P_{\text{miscall}} \) represents the probability of the base being called erroneously. A base quality greater than or equal to 20 is typically considered to be high quality, which corresponds to \( P_{\text{miscall}} \) of 1%.
FASTQ FILE FORMAT
Read sequences are stored along with their base qualities in the FASTQ file format [22]. It is an extension of the FASTA format which is used to store nucleotide sequences without quality values (e.g. the reference genome). Each read sequence is represented by a set of four consecutive lines in a FASTQ file, as follows:

- Line 1: begins with a '@' character and is followed by a sequence identifier and an optional description.
- Line 2: contains the sequence of nucleotide letters.
- Line 3: begins with a '+' character and is optionally followed by the same sequence identifier.
- Line 4: encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence. Each quality value is encoded into a single character by using the ASCII character, from the table shown in Figure 1.7, that corresponds to the integer value of the quality (plus some constant).

An example representation of a read is shown below:

<table>
<thead>
<tr>
<th>Read ID</th>
<th>@READ_58</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read sequence</td>
<td>GATTTGGGTTCAAGCAGTATCGATCAAATAGTAATCCATTGTCAACTCAC</td>
</tr>
<tr>
<td>Read ID (or +)</td>
<td>+</td>
</tr>
<tr>
<td>Base Qualities</td>
<td>!''<em>((((</em><strong>+))%%%++)(%%%%).1</strong>*-+*''))**55CCF&gt;&gt;&gt;&gt;&gt;&gt;CCCC</td>
</tr>
</tbody>
</table>

1.7.3 Paired Sequencing
The short length of HTS reads makes certain applications difficult. For example, there is increased ambiguity in read assembly, a process whereby read fragments are computationally aligned and merged together, due to short overlaps. Alignment of short reads to a reference genome is also complicated, especially where reads originate from highly repetitive or low complexity genomic regions, due to the increased number of possible matches.

To assist in these and other scenarios longer range information can be obtained through sequencing pairs of reads through paired-end sequencing. An essential property of paired reads is that the nucleotide distance that separates them, called the insert size, is known to some level of precision. Figure 1.8 shows a schematic of paired-end reads and illustrates how pairing information can be used to assist alignment in repetitive regions of a reference genome by using the unique alignment of one read in a pair as an anchor to inform the placement of the other.
The size of the inserts is chosen by design prior to sequencing and are normally between 100 to 500 basepairs. In a FASTQ file, paired reads are usually listed consecutively and share a common prefix, for example @READ_58/1 and @READ_58/2.

Much longer insert sizes can be achieved by performing mate pair library preparation, a process by which DNA strands are first circularized to bring distantly located sequences close together before being isolated and subjected to paired-end sequencing. Mate pair library preparation methodologies can enable insert sizes greater than 5 kilobases [23].

1.7.4 Alignment

The Human Genome Project pioneered the construction of the human reference genome, a representative example of a species' genome. The reference genome — though artificial and incomplete — serves as a standard to which other genomes can be compared. Reference guided alignment is a process whereby reads generated from a sequenced individual are matched to locations on the reference genome based on sequence similarity. Reference guided alignment is both conceptually and computationally simpler than de novo assembly, which requires the scaffolding and merging of reads without the aid of a reference assembly.
As will be discussed later, discrepancies in sequence identity and abundance evidenced through read alignments support the existence of genetic variation in the sequenced individual with respect to the reference genome. However, there are many other reasons why a read may not align perfectly or map to the wrong location on the reference genome:

- read amplification artifacts
- read sequencing errors
- contamination by external sequence
- bugs in the alignment algorithm or issues with its scoring scheme
- errors in the reference genome

The causes for some of these issues are consistent (e.g. Illumina technology tends to introduce sequencing errors for reads with homopolymers) but others are random (e.g. PCR sometimes alters and amplifies erroneous read sequences) [24]. It is therefore critically important for one to be aware of these possibilities when assessing inferences made from read alignments.

As with basecalling a quality score is assigned to each alignment to represent the confidence that a read was matched to the correct location, reported as a Phred Score:

\[ Q = -10 \log_{10} P_{\text{misalign}} \]

where \( Q \) represents the quality value and \( P_{\text{misalign}} \) represents the probability that the read was misaligned. A mapping quality greater than or equal to 20 is typically considered to be high quality though there are inconsistencies in the way mapping qualities are computed, especially when a read aligns equally well to multiple locations on a genome.

![Figure 1.8](http://www.illumina.com/technology/next-generation-sequencing/paired-end-sequencing_assay.ilmn)
SAM / BAM FILE FORMATS
Read alignments are represented by the standard Sequence Alignment/Map (SAM) file format [25]. A binary compressed but otherwise equivalent version of SAM, called BAM, was created to improve performance. Each read alignment is represented by a single entry in a SAM or BAM file, having the following mandatory fields:

- **QNAME**: query name of the read or the read pair
- **FLAG**: bitwise flag (pairing, strand, mate strand, etc.)
- **RNAME**: reference sequence name
- **POS**: 1-based leftmost position of clipped alignment
- **MAPQ**: mapping quality (phred-scaled)
- **CIGAR**: extended CIGAR string (operations: MIDNSHP)
- **MRNM**: mate reference name (‘=’ if same as RNAME)
- **MPOS**: 1-based leftmost mate position
- **ISIZE**: inferred insert size
- **SEQ**: query sequence on the same strand as the reference
- **QUAL**: query quality (phred-scaled)

An example representation of a read alignment is shown below:

```
@HD VN:1.0
@SQ SN:chr20 LN:62435964
@RG ID:L1 PU:SC_1_10 LB:SC_1 SM:NA12891
@RG ID:L2 PU:SC_2_12 LB:SC_2 SM:NA12891
read_28833_29006_6945
99 chr20 28833 20 10M1D25M = 28993 195 AGCTTAGCTAGCTACCTATATCTTGGTCTTGGCCG <<<<<<<<<<<<<;<<<<</,\22;<<<<
```

The CIGAR string specifies the rules for aligning the query sequence (e.g. the read) to the target sequence (e.g. the reference genome), starting from the specified position in the target. The CIGAR string is constructed from pairs of base lengths and operation pairs —
similar to run length encoding — where the operations include: match/mismatch, insertion, deletion, skip, soft clip, hard clip, and padding. It is important to note that the operation $M$ denotes either a match or a mismatch, which can only be inferred by comparison to the target sequence.

### 1.7.5 Genotyping

The process of reference guided read alignment represents a best effort to match genomic sequence from an individual being studied to the reference genome on the basis that the two genomes are nearly identical. However, genetic variations in the sequenced genome manifest in predictable discrepancies in sequence identity, topology, and abundance with respect to the reference and are evidenced in read alignments. Various techniques for genotyping an individual, i.e. compiling a list of his or her relative genetic variants, based on read alignments are now described.

#### DETECTING SINGLE NUCLEOTIDE VARIANTS

Single Nucleotide Variants occur where genomes differ in the nucleotide at a single position. SNVs are evidenced at positions where read alignments consistently mismatch the reference. Mismatches caused by true SNVs must be differentiated from mismatches caused by spurious sequencing errors and other artifacts, as shown in Figure 1.9. Genotyping software like the Genome Analysis Toolkit (GATK) [26] thus use statistical models that take into account coverage, i.e. the total number of reads mapping to a location, and base quality scores to predict the existence of a SNV at each position.

#### DETECTING INDELS

Indels occur where an additional sequence is present in one genome with respect to another. Although it is impossible to know whether over the course of time additional sequence was inserted into one genome’s lineage versus another without knowledge of the progenitor sequence, it is custom to label indel events with respect to the reference sequence. That is, additional sequence present in the sequenced genome is considered an insertion while absent sequence is considered a deletion.

Deletions are detectable via methods of gapped alignment (for small events) and split mapping (for large events), both of which allow two non-overlapping parts of a read to align at some distance apart — which corresponds to the size of the event. Theoretically, deletions of
any length can be identified through this approach as long as a read spans the breakpoint, the chromosomal origin of the event, and significantly overlaps either side to allow for confident partial alignment.

Insertions can also be found through single read mapping analysis, though the length of such events is limited by the length of reads. Specifically, if a read generated from the sequenced genome completely contains the inserted fragment, it is possible to align the parts of the read to reveal the insertion breakpoint and sequence. This technique requires a read to contain the inserted sequence and for flanking parts to be mappable to the reference genome. The described approaches for indel detection using single read mapping analysis are illustrated in Figure 1.10.

**DETECTING STRUCTURAL VARIANTS**

Structural variations, including large indels, duplications, inversions, and translocations, are more complex events that alter the abundance and arrangement of genomic sequences. Two complementary approaches can be used to detect changes in each.
A genomic segment that has increased or decreased in abundance (e.g. for CNVs) will have more or less reads generated from it compared to copy-number stable regions. The depth of coverage (DOC) approach identifies such events where the coverage, i.e. the number of reads aligning to the genome, is discrepant from the average. While the DOC approach can predict the relative abundance and breakpoints of the inserted or deleted sequence, it cannot recover its topology.

The paired-end mapping (PEM) approach uses information from paired-end alignments to detect genomic rearrangements. A genomic segment that moves or changes orientation within a genome will generate read pairs that do not align concordantly with the known insert size or the relative orientation of paired-end reads. The PEM identifies structural variations where clusters of discordantly mapped read pairs correspond to known signatures as shown in Figure 1.11. The technique is conceptually elegant but it is complicated in practice due to the repetitive nature of genomic sequence around breakpoints and the tendency for structural events

---

**Figure 1.11** Paired-end mapping signatures for structural variants. Read pairs are represented by arcs, where the arc endpoints represent the mappings of the component reads. Pairs that align normally, i.e. respecting read pair orientation and insert size, are shown in black. Pairs that map with proper orientation but with discordant size are coloured purple. Pairs that map in the wrong orientation with respect to sequencing are shown in orange, and the offending endpoint is denoted by a *. Pairs that map in the wrong order are shown in yellow.
to occur in combination. In previous work, we showed that DOC and PEM methods can be used in a complementary fashion to predict changes in abundance and arrangement of structural variants simultaneously [27].

**VCF FILE FORMAT**

Most genetic variants can be easily represented in the file-based Variant Call Format (VCF) [28]. The VCF is a multisample format, meaning it is designed to contain information from any number of different individuals (or different samples within an individual). Each variant is described in a separate line of a VCF file, having the following fields:

- **CHROM**: chromosome
- **POS**: position within the chromosome
- **ID**: unique identifier(s)
- **REF**: reference sequence(s)
- **ALT**: non-reference sequence(s)
- **QUAL**: quality score (phred-scaled)
- **INFO**: additional information, in key-value pair form
- **FORMAT**: datatypes and order of the genotype fields

These fields are followed by genotype fields containing sample-specific information according to the **FORMAT**. Like the **INFO** field, the genotype fields are represented in delimited key-value pair form. An example listing of a variant is shown below:

```
Header
First 7 fields
INFO
FORMAT
Genotype field
```

The VCF format is able to represent SNVs, indels, and structural variants though the latter are sometimes awkward to represent due to the length of sequences involved and their topological complexity.
1.7.6 Applications of Genotyping

The described workflow for genotyping individuals using data from high-throughput and cost-effective sequencing technologies makes possible deep cataloguing of human variation. Massive sequencing efforts like the UK10K [29] and The Cancer Genome Atlas [30] are sequencing and genotyping huge numbers of individuals from a wide spectrum of healthy and disease cohorts. The catalogues of putative variants obtained from these and other large-scale sequencing efforts establish a foundation upon which we can better understand human genetics and disease.

The intersection of genomic variants datasets with diverse datasets (e.g. phenotypes, drugs, environment) allows for associations to be gleaned. The genetic etiology of disease phenotypes have already been resolved in many Mendelian disorders [31] and progress is being made for complex multifactorial disorders like cancer [32] and autism [33]. Pharmacogenomic research, which examines the effect of genetics on drug therapies, has revealed specific genetic mutations that render an individual unable to properly metabolize drugs and instead produce possibility life-threatening toxic byproducts [34]. Significant associations between genetics and behavioural or environmental factors like smoking, pollution, and sun exposure are well studied and form the subject of genetic epidemiology.

The analysis of genotypes acquired from sequencing in context of diverse datasets gives rise to applications and resources that are ultimately enabling the translation of genomic medicine into clinical settings, depicted in Figure 1.12 (adapted from [35]), so that personal genomes can be used to care for patients more precisely than was ever before possible.

1.8 Genomic Medicine

The increased fidelity with which personal genomes can be examined through sequencing has potential to revolutionize methodologies of disease assessment (e.g. prenatal screening, risk profiling, preventative therapy), detection (e.g. diagnosis, classification, prognosis), and treatment (e.g. drug prescription, response) to better care for individuals affected by genetic conditions. The term genomic medicine is used to describe a yet unrealized model of medical practice where the course of an individual’s medical treatment is guided by his or her personal genetic makeup.
Genomic medicine promises to have immense medical and economic benefits, including the replacement of multiple expensive genetic tests with a single low cost genome sequence, reducing disease morbidity in patients with high genetic risk through early detection and treatment, as well as reducing costly adverse drug reactions.

1.8.1 Challenges in Enabling Genomic Medicine

Despite its clear benefits to human health, there are a number of challenges that currently prevent the broad implementation of genomic medicine. These are conceptual, technical, political, ethical, and regulatory in nature, and are discussed briefly here:

CONCEPTUAL AND TECHNICAL ISSUES

Despite the capacity of new sequencing technologies to generate huge volumes of raw sequence data, it remains a substantial informatics challenge to efficiently analyze it. Key technical problems lie in the storage and processing of sequencing and related datasets generated through genotyping pipelines. HTS technologies already produce data at a rate that exceeds Moore's Law: the trend in computer hardware to double in power every two years. The capacity for storing this data is increasing at a slower pace, however [19]. In an effort to reduce digital storage requirements of genotyping pipelines it is common practice to store only highly reduced summaries of the data (e.g. genetic variants) and permanently delete raw upstream forms of it (e.g. reads or read alignments).
Computational processing of genomic data is another significant challenge. Most software tools for HTS data analysis are developed by the research community as utilities that loosely interoperate using standardized file formats. There are often multiple tools that can be used to perform a particular task (e.g. there are over a half dozen popular read aligners) and most research institutions take discretion on which to use based on factors such as speed, ease-of-use, support, familiarity, or popularity. There is no agreed upon off-the-shelf solution for end-to-end genomic data processing and so a significant amount of human resources is required to design, build, debug, and maintain environments for running computational pipelines to process genomic data.

Optimizing these pipelines is particularly time-consuming. While some application developers publish a set of best-practices that specify a set of default instructions for use of their specific tool, it is often necessary to adjust parameters to achieve better results for specific data types or applications. It is thus typical to build custom computational workflows using ad-hoc combinations of command-line scripts whose parameters are manually tuned through many iterations of trial-and-error. The amount of time spent manually configuring, rerunning, and debugging workflows is unpredictable, and is often more time consuming than computational processing time itself as shown in Figure 1.13.

Once genotypes have been identified, the problem of associating genetic variation to a phenotype, especially one with clinical relevance, requires a lot of effort and expertise in human genetics. This is because most candidate variant sites identified in such studies are either not real variants (due to errors in the prediction pipeline) or are real but have no functional effect. With existing tools, variant interpretation is a tedious iterative process of specifying quality and functional impact filters on variant sets and inspecting them for relevance.

Genetic researchers who are skilled at assessing the clinical relevance of genetic variants typically lack the computational expertise while most utilities for the analysis of genomic data — being command-line tools with many parameters and resource requirements — need significant informatics expertise to run. A study conducted in 2011 by Golden Helix, a genomics software vendor, found that less than 10% of all genetics analysis software had been maintained over a year’s time and that only about 25% of these had a graphical user interface [36], meaning that less than 2% of all genetics analysis software are well-maintained graphical programs; these
are the narrow subset of tools that are most accessible to genetic researchers and clinicians without formal informatics training. 

Notwithstanding a few exceptions, users are often forced to choose between powerful software that needs to be run on the command-line or less powerful but graphical applications. Spreadsheet software like Microsoft Excel is a common example of the latter, which is a general-purpose desktop application that is accessible by genomics researchers who are not trained in informatics despite its inability to handle large datasets, produce appropriate visualizations, or integrate dynamically with external datasources. In the absence of a system for genetic data
analysis that is both powerful and easy to use, a multidisciplinary team is required where there is constant communication between the people configuring and running the computational tools (e.g. bioinformaticians) and those interpreting their results (e.g. technicians and geneticists). Coordination between various stakeholders in the process can be a significant bottleneck.

**POLITICAL, ETHICAL, AND REGULATORY ISSUES**

There are also significant political, regulatory, and ethical issues that need to be considered for responsible implementation of genomic medicine. Issues around intellectual property rights — for example, the right to patent diagnostic tools, genes, and biomarkers — remain controversial. While opponents of patents argue that they prohibitively restrict medical progress, proponents argue that patents are required to protect financial investments that are needed to fund research and development. In June 2013 the United States Supreme Court ruled that naturally occurring genes cannot be patented while synthetic DNA, i.e. edited or artificially made, can [37].

Another key concern is the need to respect patient privacy and confidentiality. The data being collected through genomic medicine may be personally identifiable not only to the individual being assessed but also his or her close relatives; hence policies must be devised to properly consent individuals and to ensure that systems which deal with patient data make efforts to protect their identities. The Health Insurance Portability and Accountability Act (HIPAA), enacted in the United States in 1996, specifies standard measures that must be taken to protect the privacy of individually identifiable health information [38]. Moreover, in 2008 the Genetic Information Nondiscrimination Act (GINA) was passed in the United States which restricts the use of genetic information by employers or insurers [39].

Genomic medicine may divulge sensitive information that could have negative emotional and psychological effects on an individual and his or her family. For example, there is significant debate over the disclosure of incidental findings, previously undiagnosed medical conditions that are discovered unintentionally or during the course of investigating a different condition. In March of 2013 the American College of Medical Genetics (ACMG) issued recommendations that laboratories performing clinical sequencing seek and report incidental findings in select medically actionable genes to the attention of the ordering clinician [40]. Controversially, the original recommendations were that findings “be reported without seeking preferences from the patient and family and without limitation due to the patient's age”. In a press release issued in
April of 2014 it was announced that the recommendations would be updated, giving patients the opportunity to opt-out of testing and reporting on these genes before it takes place [41].

The potential to use genetic risk analysis to make positive interventions through preventative medicine and lifestyle changes favours the disclosure of incidental findings for actionable genetic conditions. This topic has received significant public attention owing to revelations by public figures, notably actress Angelina Jolie [42] and Google founder Sergey Brin [43], both of whom have been influenced by genetic risk analysis to take preemptive measures to reduce their high inherited risks for disease. An excerpt taken from Brin's blog, in which he discusses his genetic risk analysis, is shown in Figure 1.14.

“I feel fortunate to be in this position. Until the fountain of youth is discovered, all of us will have some conditions in our old age only we don’t know what they will be. I have a better guess than almost anyone else for what ills may be mine -- and I have decades to prepare for it.”

— Sergey Brin

Figure 1.14 Excerpt taken from the blog of Sergey Brin, founder of Google, who discovered through genetic testing that he has inherited a mutation in the gene LRRK2 which confers higher risk of developing Parkinson’s disease.

1.9 Thesis Objectives

This thesis intends to develop novel ideas and technologies for visualization and analysis of personal genomes through the application of techniques in computer science including data structures, databases, algorithm design, data visualization, user interface design, and user experience design. The work results in two software platforms that focus on assisting the inspection and interpretation of genetic variants by genomic researchers. These tools motivate a conceptual advance in the navigation of genomes (which has traditionally been done using a linearized coordinate system) to enable exploration by leveraging secondary, but complementary datasets. The power of systems designed under such a framework will continue to increase as the bodies of knowledge in other areas of biological research (for example, in gene, protein, phenotype, and disease associations) continues to expand. These tools are developed with the intent that they, in addition to the concepts they embody, may be useful towards the realization of genomic medicine.
2 Visualizing Personal Genomes
The amount of genomic information that is generated by sequencing technologies exceeds that which can be analyzed entirely by hand. The need to efficiently analyze enormous amounts of information has promoted growth in the field of bioinformatics, which aims to use the power of computation to help solve biological problems, like the etiology of genomic disease. A large number of bioinformatics tools have been developed for analyzing genomic data by way of statistics, machine learning, or similar computationally-intensive processes. While many genome analysis tasks can be performed through automated computational pipelines, some steps (e.g. read alignment quality assessment, genotyping quality assessment, and variant interpretation) continue to require human judgement. As depicted in Figure 1.13, these steps are often rate limiting in the absence of tools that facilitate manual analysis.

Human interpretation is significantly aided by information visualization. Graphics leverage the human visual sensory system to quickly detect patterns and reason from images. The most effective way for researchers to explore, summarize, and communicate a dataset – however large – is often through a graphical representation of it.

The role of information visualization is as critical to genomic workflows as it is for other quantitative sciences like finance, astronomy, and meteorology. In these domains, the processes being analyzed are complex and multivariate, and are often too poorly understood to be characterized by a computational approach, at least initially. Visualizations help researchers to quickly explore, characterize, and explain trends: knowledge that can eventually be used to develop and refine automated computational tools.

2.1 Related Work
The following section describes relevant concepts in information visualization, before discussing specific software tools for information visualization in the field of genomics.

2.1.1 Information Visualization
A common misconception is that computational methods for data analysis are more powerful than manual ones. However, visual inspection of information is tremendously effective for a number of tasks such as pattern recognition and outlier detection, perhaps more so than computational techniques. This is evidenced by considering Anscombe’s quartet: four datasets whose summary statistics (mean, standard deviation, and line of best fit) are identical when
computed, yet when visualized as in Figure 2.1 are clearly generated by different underlying mechanisms.

When designing scientific visualizations, especially for large multivariate datasets, it is useful to employ established techniques to effectively draw viewers' attention to salient features (e.g. patterns or outliers) while managing visual complexity. Fundamental techniques for doing so include: exploiting preattentive processing, minimizing data-ink ratio, creating task-specific visualization modes, and revealing increasingly detailed information through progressive disclosure.

PREATTENTIVE PROCESSING

Our natural ability to identify patterns and outliers in visualizations is driven by preattentive processing, a constant unconscious analysis of the visual field that detects basic visual features. These attributes are referred to as preattentive features and include colour, closure, size, shape, and orientation; a few of these are shown in Figure 2.2. A feature is considered to be processed preattentively if target detection, the ability to detect the presence or absence of an element having a unique visual property within a field of distractor elements, can be accomplished in <200ms, i.e. faster than eye movement. Preattentive processing is done quickly, effortlessly, and in parallel without any special attention being focused on any one element within an image.

---

*Figure 2.1* Graphical visualization of Anscombe's quartet, four datasets having equal mean, standard deviation, and regression lines. Downloaded from [http://en.wikipedia.org/wiki/Anscombe's_quartet](http://en.wikipedia.org/wiki/Anscombe's_quartet).
Visual features that have the highest salience, i.e. elements that stand out the most, are selected for further and more complete analysis by conscious, i.e. attentive, processing [44].

Preattentive processing has been shown to facilitate the following tasks:

- target detection: detect the presence or absence of an element having a unique visual property within a field of distractor elements
- boundary detection: detect the boundary between two groups of elements, where all of the elements in each group have a common visual property
- region tracking: track one or more elements having a unique visual property as they move in time and space
- counting and estimation: count or estimate the number of elements with a unique visual property

Use of preattentive processing can be effective in media, advertising, and in information visualization. However, preattentive features must be used judiciously. As illustrated in Figure 2.3 they often cannot be used conjunctively: target detection where the target is unique in either colour or shape is preattentive, but not when the target is unique only when considering both colour and shape.
Text is not preattentive as shown in Figure 2.4 [45]; it is often useful to adjust the slant, weight, or colour of important words within text to make them more prominent as done in this corpus.

**DATA-INK RATIO**

The visual complexity of an image can be characterized by the number and variety of visual elements in it. It is considered good design to reduce the complexity of an image while maintaining its information content as much as possible, a balance that can quantitatively assessed by computing data-ink ratio: a measure coined by Edward Tufte, a pioneer in the field of data visualization, for the proportion of a graphic’s ink devoted to non-redundant display of data information [46]. Data-ink ratio is computed as follows:

\[
data-ink \text{ ratio} = \frac{\text{data-ink}}{\text{total ink used to print the graphic}} = \frac{\text{the proportion of a graphic’s ink devoted to the non-redundant display of data-information}}{1.0 – \text{proportion of a graphic that can be erased without loss of data-information}}\]

A principle put forth by Tufte is to maximize the data-ink ratio, within reason, exemplified in Figure 2.5. This includes removing superfluous graphic borders, axes, textures (which often create moiré effects), and other distracting decorations commonly referred to as chartjunk. There is of course a threshold beyond which removing non-data elements prevents the data from being easily understood; it may be better to present visualizations using a representation that is suboptimal with respect to data-ink ratio, for instance, if there is already a standard visualization and the introduction of a novel representation is confusing for viewers.
VISUALIZATION MODES

Scientific datasets are often complex and multivariate and the relevance of a specific facet of the data depends on the task that the visualization is being used for. For example, maps used for providing driving directions should display street names and intersections whereas maps used for hiking should clearly illustrate contour as shown in Figure 2.6. Where a dataset can be visualized for different purposes, creation of task-specific visualization modes helps to reduce visual complexity while focusing attention on most pertinent features.

Figure 2.5 Optimizing data-ink ratio. Three different visualizations of the same dataset with low (left), medium (middle), and high (right) data-ink ratios. Tufte’s recommendation is to maximize the data-ink ratio within reason.

Figure 2.6 Google Maps images of Santa Monica, California displayed using two different visualization modes. The traffic mode (top) displays information relevant to commuters while terrain mode (bottom) displays information that is helpful for hikers and urban planners.
PROGRESSIVE DISCLOSURE
Visualization modes reduce visual complexity by emphasizing the small number of facets of datasets relevant for the particular task at hand. Even still, the sheer size and depth of some datasets often introduces far more complexity to be useful when viewed all at once. The colloquial term information overload describes one’s difficulty in understanding and making decisions from a dataset caused by the presence of too much information. For example, network visualizations like the one in Figure 2.7 that display connections between all nodes in a graph are often referred to as hairballs because they are messy and impossible to visually disentangle without using techniques to reduce their complexity.

In these situations it is useful to manage the amount of information displayed at a given time through progressive disclosure: a technique in interaction design where increasingly detailed information is revealed as it is requested by the viewer. For example, it would be difficult to use a map that displays street names when visualized at a global scale; only when zoomed into specific regions does this level of detail become useful. Progressive disclosure typically follows the notion of presenting data from abstract to specific. When this concept is applied to zooming, where zooming in reveals increasingly more detailed information, it is called semantic zooming. Resolution-dependent data smoothing is another embodiment of progressive disclosure that is often used in the interactive visualization of continuous data such as time-
series stock price shown in Figure 2.8. Progressive disclosure can apply to the order of presentation features of software in addition to information in visualizations.

The above-described techniques aid in the design of information visualization and are particularly applicable for the presentation of large, quantitative datasets that emerge from genomic workflows. The extent to which a particular software tool adheres or violates these principles is also useful in evaluating its effectiveness. Relevant tools for genomic information visualization are now presented in this context.

2.1.2 Genome Browsers & Other Information Visualization Tools
Visualization of genomic data gives researchers the benefit of looking at information in a more natural and interpretable way compared with a textual representation. There are many different tasks in genome analysis that are facilitated by visualization including: (i) integration of multiple related datasets into a single view, to gain insight into the interaction between genomic features, (ii) algorithm development, where visualization of many putative calls (e.g. genomic variants, promoter sites, intron–exon boundaries, etc.) helps with debugging and identification of true and false positives; and (iii) exploration of various genomic regions for specific signatures of functional sites that may be difficult to describe within a computer program (e.g. two closely

![Figure 2.8 Resolution-dependent data smoothing of financial data on Google Finance. Zoom level 1d (top) shows detailed stock data on an hourly basis. Zoom level 1y (bottom) shows less detailed stock data aggregated over monthly periods.](image-url)
spaced peaks in ChIP-seq data indicating adjacent binding sites). Without the convenience of a visualization tool, in each of these settings all regions of interest would have to be painstakingly considered via manual analysis of the supporting data. The following subsections survey relevant and popular software for information visualization of genomic datasets, beginning with the most commonly used class of tools: genome browsers.

GENOME BROWSERS
A genome browser is an instrument for visualizing annotations (e.g. genes, nucleotide conservation across species, genetic variants, etc.) within the context of a linearized genome. This setup assumes a reference genome, i.e. a pre-assembled nucleotide sequence that is designated as the gold standard to which all other individuals and annotations are compared. It should be emphasized that the designation of a genome as the reference is convenient yet arbitrary, since lineages evolve in parallel and so there is no true concept of a correct genome to which others should be compared. The reference genome is assumed to represent that of a healthy individual devoid of genetic disease, and so individuals who are affected by a genetic disease will have differences in their genomes with respect to the reference where causal mutations exist,. However, these mutations must be disentangled from non-harmful variations that have arisen naturally during the course of evolution. Due to errors in the genotyping process and gaps in our understanding of the human genome, this distinction is not trivial.

UCSC GENOME BROWSER
The UCSC Genome Browser was one of the first genome browsers for navigating human genomes. Though it was initially designed to visualize highly-curated annotations associated with sequenced genomes (not individual genomes themselves), it was pioneering in data delivery and interface layout and has served as an exemplar for many successful genome browsers that derived from it, including this work.

The UCSC Genome Browser was published in 2002 almost concurrently with the end of the Human Genome Project [47]. It thrived as a visualization medium for highly curated datasets associated with the human genome and continues being developed as such for many sequenced organisms. Most genome browsers adhere to a standard layout that is embodied by UCSC Genome Browser, where coordinates of a linearized reference genome run along the horizontal axis and other annotations are visualized in ideograms called tracks that are vertically stacked on top of each other. This basic template is demonstrated in Figure 2.9.
Because visual displays have much fewer pixels than there are positions in a genome (the human genome has over three billion), all genome browsers allow users to view subregions using a simple genomic coordinate system. The subregion being displayed is typically highlighted on an ideogram of the chromosome being examined. Each track contains data of one of a few general types: point (e.g. for SNVs), interval (e.g. for genes), and continuous (e.g. for showing interspecies conservation).

The visual encoding of biological elements is relatively standard across genome browsers: the reference genome is displayed as a sequence of letters; a gene is encoded as a series of filled boxes and directed lines (where boxes represent exons, lines represent introns, and the direction denotes which of the two DNA strands the gene is found); continuous tracks are displayed as filled line graphs or bar charts. Visual representation of continuous datatypes is inspired by earlier work on sequence alignment visualizations called Vista Plots [48]. The visual encodings for these basic elements as currently implemented in the UCSC Genome Browser are shown in Figure 2.10.

Navigation can be performed through a number of controls: zoom-in and zoom-out buttons, pan left and pan right buttons, interactive chromosome ideogram, and a genomic coordinates field. Genomic coordinates (based on a linearized genome) are specified in the following format:

\[
<\text{chromosome}>:<\text{start position}> - <\text{end position}>
\]
Seeking to a region of interest is most commonly performed through specification of genomic coordinates and these are typically generated by automated computational analyses (e.g. a CNV prediction algorithm outputs a list of genomic regions it predicts to be variable).

**NON-LINEAR GENOME VISUALIZATION TOOLS**

The linearization of genomes is an artificial abstraction that is often forgotten. DNA molecules are known to adopt complex three-dimensional conformations such that positions thousands or even millions of basepairs away (in the linear sense of distance) are physically close together. Trans-acting relationships have been found to be very important for the proper functioning of genomes [49] but are not well represented by existing genome browsers. A three-dimensional genome browser named Sockeye was developed and published in 2004 [50]; surprisingly, the effort still linearized the genome and the extra dimension was sub-optimally occupied with superfluous visual effects. A screenshot of Sockeye is shown in Figure 2.11.

A more successful deviation from the standard genome browser model is Circos, a program that visualizes tracks along a circularized genome [51]. Long distance genomic relationships (e.g. large structural variations or trans-acting regulation) can be encoded by bands that pass through the space near the center of the circle. Bands are rendered semi-transparently, giving an indication of density where there is overlap. An example image produced by Circos is shown in Figure 2.12. The circularization concept was likely inspired from illustrations of bacterial genomes, which are in fact circular. Nevertheless the generalization of this technique for all genomes is aesthetically pleasing and better suited to capture continuous or long distance relationships compared to the linear approach. Unfortunately Circos only

---

**Figure 2.10** Visual representations of genomic sequence, genes, and continuous valued tracks in UCSC Genome Browser,
generates static global images and does not allow for dynamic interaction in the form of zooming, panning, selecting, or filtering. This limitation hinders its ability to supplant traditional genome browsers as an exploratory tool for genomics data at nucleotide resolution.

WEB-BASED GENOME BROWSERS
Besides the artificiality in representing genomes linearly, there is another important shortcoming exhibited by the first genome browsers. Many of these were developed using traditional synchronous web technologies, where a centralized web-server renders static visualizations that are sent to and viewed by connected clients on demand. The synchronous request model restricts the level of interactivity that can be achieved between the user and the data due to the latency involved in communicating with an intermediary server. Every time the genomic viewport in the UCSC Genome Browser is shifted by any amount, for example, the webpage is entirely refreshed after a few seconds. The navigation experience is thus jarring and requires significant cognitive work by viewers to reorient themselves and resume their task. Traditional web-based genome browsers are generally considered to be slow and non-interactive.

Figure 2.11 Sockeye, a 3D genome browser. The coordinate system is still linearized, and most of the space gained by the additional dimension is occupied by chartjunk. Downloaded from http://www.bcgsc.ca/project/bomge/sockeye on August 1 2012.
Web-based genome browsers that make use of asynchronous communication protocols, as in JBrowse [52], have been developed to deliver a more fluid genome browsing experience even for large datasets of the kind generated by HTS machines. However, the web-based model is fundamentally flawed for the purpose of displaying HTS datasets, which are typically large, often contain sensitive patient information, and are locally stored. Visualization of HTS datasets using web-based genome browsers is problematic because: (i) a large amount of locally stored data must be uploaded to servers across the internet, (ii) doing so could compromise the security of sensitive patient information, and (iii) once uploaded, the data being visualized cannot easily be manipulated or computed with.

While the web-based rendering model works well for communicating highly curated information related to the reference genome, it is not ideal for use with HTS datasets which are normally stored on hard drives local to client machines, are very large in size, and contain sensitive patient information that must be kept confidential and within the firewalls of whatever research institution is performing the analysis. A competing paradigm is the client-side rendering model, which places the rendering engine directly on the client machine, thereby removing the need to transmit data through an intermediary, and bettering the ability for

Figure 2.12 Circos genome visualization. Tracks are circularized; long distance relationships connect positions along the perimeter through the center. Downloaded from http://circos.ca on August 1 2012.
interaction between users and their data. A schematic comparing the two paradigms is shown in Figure 2.13.

**GENOME BROWSERS FOR HTS SEQUENCING DATA**

The 1000 Genomes Project is a large international project launched in January 2008 that sequenced and genotyped the genomes of over 1,000 individuals from various ethnic groups in an effort to establish the most detailed catalogue of human variation compiled to date [53]. The individuals chosen for this study consented to have their genetic information made publicly but anonymously available online.

**Figure 2.13** Two common architectural paradigms for genome browsers. The web-based rendering model places the rendering engine on a centralized server. Local data is uploaded to the server over the internet and rendered images are downloaded back. The client-side rendering model places the rendering engine on the client machine. Local and remote datasets are downloaded and rendered on the same machine as viewing.
As sequencing continues to be economized, it has become practical to perform sequencing for disease cohorts, families, and single individuals in a clinical setting. However, it is atypical for patients who are subscribed for sequencing in a clinical setting to consent to their data being made publicly available. Most clinical genetics labs have very strict policies that restrict patient data which may be personally identifiable from being transferred (over the internet or otherwise) outside of their internal networks.

The need to analyze large amounts of private and locally stored data marked an important paradigm shift in genetic analysis requiring labs to develop internal infrastructures to quickly store, analyze, and visualize large datasets. For reasons discussed previously, existing web-based genome browsers are not appropriate for this purpose and at the onset of this work, in 2009, there was significant need for a genome browser that could:

- visualize read alignments and genetic variants (in the BAM and VCF file formats)
- allow viewing of tracks stored locally and remotely
- be fast, allowing navigation with minimal rendering times
- be secure, not uploading sensitive information to the internet
- support the tasks of assessing base and read alignment qualities
- support the tasks of performing visual variant calling or assessing predictions made by external algorithms

A list of the most popular genome browsers developed to date are provided in Table 2.1. As can be seen, most genome browsers that have been developed since 2009 support HTS datasets. Perhaps the most widely-used of these is the Integrative Genomics Viewer developed at the Broad Institute, which was developed simultaneously and sometimes collaboratively with the work discussed in this chapter.

**INTEGRATIVE GENOMICS VIEWER**

The Integrative Genomics Viewer is a cross-platform genome browser, architected under client-side rendering model, with support for concurrent visualization of diverse datasets, including HTS datasets. IGV was first published in January 2011, which described its techniques for visualizing read alignments and paired information to aid in the visual inspection of basic genetic variants like SNVs and deletions.
Table 2.1 List of popular genome browsers.

*unpublished, earliest release date shown

** the work described in this thesis

<table>
<thead>
<tr>
<th>Name</th>
<th>Website</th>
<th>Published</th>
<th>Platform</th>
<th>Cost</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artemis Genome Browser</td>
<td>sanger.ac.uk/resources/software/artemis/</td>
<td>2000</td>
<td>Client</td>
<td>Free</td>
<td>General purpose, retrofitted for HTS data in 2012</td>
</tr>
<tr>
<td>Viral Genome Organizer</td>
<td>troy.bioc.uvic.ca/tools/VGO</td>
<td>2001</td>
<td>Client</td>
<td>Free</td>
<td>Virus genomes</td>
</tr>
<tr>
<td>Gbrowse</td>
<td>gmod.org/wiki/GBrowse</td>
<td>2002</td>
<td>Web</td>
<td>Free</td>
<td>General purpose, retrofitted for HTS data in 2013</td>
</tr>
<tr>
<td>UCSC Genome Browser</td>
<td>genome.ucsc.edu</td>
<td>2002</td>
<td>Web</td>
<td>Free</td>
<td>Genes and other genomic annotations</td>
</tr>
<tr>
<td>BugView</td>
<td>doolittle.ibis.gla.ac.uk/Leader/BugView</td>
<td>2004</td>
<td>Client</td>
<td>Free</td>
<td>Bacterial genomes</td>
</tr>
<tr>
<td>Ensembl</td>
<td>ensembli.org</td>
<td>2004</td>
<td>Web</td>
<td>Free</td>
<td>General purpose, uses Genoverse</td>
</tr>
<tr>
<td>Microbial Genomic Viewer</td>
<td>mg2.cmbi.ru.nl</td>
<td>2004</td>
<td>Web</td>
<td>Free</td>
<td>Microbial genomes</td>
</tr>
<tr>
<td>VISTA genome browser</td>
<td>pipeline.lbl.gov</td>
<td>2004</td>
<td>Web</td>
<td>Free</td>
<td>Sequence comparison</td>
</tr>
<tr>
<td>CGView</td>
<td>wishart biology ualberita.ca/cgvie/</td>
<td>2005</td>
<td>Web</td>
<td>Free</td>
<td>Circular genomes</td>
</tr>
<tr>
<td>Argos Genome Browser</td>
<td><a href="http://www.broad.mit.edu/annotation/argo/">www.broad.mit.edu/annotation/argo/</a></td>
<td>2007</td>
<td>Client</td>
<td>Free</td>
<td>Genome annotation</td>
</tr>
<tr>
<td>DiProGB</td>
<td>diprog.fil-leibniz.de</td>
<td>2009</td>
<td>Client</td>
<td>Free</td>
<td>General purpose</td>
</tr>
<tr>
<td>HuRef</td>
<td>huref.jcvi.org</td>
<td>2009</td>
<td>Web</td>
<td>Free</td>
<td>Reference comparison</td>
</tr>
<tr>
<td>Integrated Genome Browser</td>
<td>bioviz.org/igp/</td>
<td>2009</td>
<td>Client</td>
<td>Free</td>
<td>General purpose, retrofitted for HTS data in 2010</td>
</tr>
<tr>
<td>Jbrowse</td>
<td>jbrowse.org</td>
<td>2009</td>
<td>Web</td>
<td>Free</td>
<td>Embeddable, retrofitted for HTS data in 2012</td>
</tr>
<tr>
<td>Avadis</td>
<td>avadis-rsg.com/features/genome_browser</td>
<td>2010</td>
<td>Client</td>
<td>Paid</td>
<td>HTS data</td>
</tr>
<tr>
<td>Gaggle Genome Browser</td>
<td>gaggle.systemsbiology.net</td>
<td>2010</td>
<td>Web</td>
<td>Free</td>
<td>General purpose</td>
</tr>
<tr>
<td>Savant Genome Browser**</td>
<td>genomesavant.com</td>
<td>2010</td>
<td>Client</td>
<td>Free</td>
<td>HTS data, integrated analytics</td>
</tr>
<tr>
<td>The Genomic HyperBrowser</td>
<td>hyperbrowser.uio.no/hb/</td>
<td>2010</td>
<td>Web</td>
<td>Free</td>
<td>Integrated analytics</td>
</tr>
<tr>
<td>Annmap</td>
<td>annmap.cruk.manchester.ac.uk/</td>
<td>2011</td>
<td>Web</td>
<td>Free</td>
<td>Affymetrix microarray data</td>
</tr>
<tr>
<td>Dalliance</td>
<td>biodalliance.org</td>
<td>2011</td>
<td>Web</td>
<td>Free</td>
<td>Embeddable browser, HTS data</td>
</tr>
<tr>
<td>Genome Wowser</td>
<td>cbmi.chop.edu</td>
<td>2011</td>
<td>Mobile</td>
<td>Free</td>
<td>Wraps UCSC</td>
</tr>
<tr>
<td>GenomeView</td>
<td>genoview.org</td>
<td>2011</td>
<td>Client</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>GenPlay</td>
<td>genplay.einstein.yu.edu</td>
<td>2011</td>
<td>Client</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>Integrative Genomics Viewer</td>
<td>broadinstitute.org/igv/</td>
<td>2011</td>
<td>Client</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>WashU EpiGenome Browser</td>
<td>epigenomemegateway.wustl.edu</td>
<td>2011</td>
<td>Web</td>
<td>Free</td>
<td>Epigenetics</td>
</tr>
<tr>
<td>Golden Helix Genome Browser</td>
<td>goldenhelix.com/GenomeBrowse</td>
<td>2012</td>
<td>Client</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>NextBio Genome Browser</td>
<td>nextbio.com</td>
<td>2012</td>
<td>Web</td>
<td>Paid</td>
<td>HTS data</td>
</tr>
<tr>
<td>TGAC Browser</td>
<td>tgac-browser.tgac.ac.uk</td>
<td>2012</td>
<td>Web</td>
<td>Free</td>
<td>General purpose</td>
</tr>
<tr>
<td>Trackster</td>
<td>wiki.galaxyproject.org</td>
<td>2012</td>
<td>Web</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>UGENE</td>
<td>ugene.uniprot.ru</td>
<td>2012</td>
<td>Client</td>
<td>Free</td>
<td>Integrated analytics</td>
</tr>
<tr>
<td>GeneWall</td>
<td>wobblebase.com</td>
<td>2013</td>
<td>Mobile</td>
<td>Paid</td>
<td>General purpose</td>
</tr>
<tr>
<td>Genome Maps</td>
<td>genomemaps.org</td>
<td>2013</td>
<td>Web</td>
<td>Free</td>
<td>HTS data</td>
</tr>
<tr>
<td>Genoverse</td>
<td>genoverse.org</td>
<td>2013</td>
<td>Web</td>
<td>Free</td>
<td>Embeddable</td>
</tr>
</tbody>
</table>
IGV represents reads and their alignments in a manner that is standard across most genome browsers. A read is depicted by a rectangle, which denotes the read sequence, that is terminated by a pointed end, which denotes the strand to which the read was mapped. Reads are positioned along the x-axis according to their mapped position, and piled vertically — e.g. by a greedy algorithm — to avoid visual overlap between reads. In the standard visualization mode, positions within a read are coloured differently if the base at that position in the read is mismatched with the base to which the read was aligned in the reference genome. A schematic of this representation is shown in Figure 2.14.

This standard visualization mode makes identification of SNVs by eye simple, even preattentive. A SNV causes all reads generated from a sequence to contain the same mismatch at the variable location and so an SNV is evidenced visually in a read alignment track as a position being consistently mismatched as seen in Figure 2.15.

The vertical packing of reads in the above-described way yields a distribution that approximates coverage, the number of reads sequenced from the genome at a specific position, and so prediction or validation of structural variations based on the depth-of-coverage approach is possible and is exemplified in Figure 2.15. An alternate visualization mode in IGV can be used to display paired information. In this view, a line is drawn between paired read representations, and so the length of the line corresponds to the inferred insert size. Pairs of reads and the line connecting them are displayed on the same vertical level, and packed so that none of the pairs overlap as in Figure 2.16. As discussed in the previous chapter, pairs of reads map discordantly if: the inferred insert size is significantly different from what is expected given the library preparation, the reads align with aberrant orientation with respect to each other, or reads align to entirely different chromosomes. In IGV the read representations are coloured for discordant pairs. Users can specify the range of insert sizes that should be considered by the
software as non-discordant. This visualization mode helps to identify breakpoints for simple structural variants that can be identified using the paired-end method.

Visualization of HTS read alignments assists in manual validation of results of read alignment and genotyping algorithms. Once the output of these automated tools are deemed acceptable, usually after many iterations of parameter refinement and manual inspection, the task of variant interpretation remains. For example in disease sequencing studies a researcher,

Figure 2.15 Visual identification of genetic variants in IGV. Two read alignment tracks from a tumor/normal pair. Read alignments from the normal and tumor samples are shown in the first and second tracks, respectively. There is evidence for a G>C SNV at the position zoomed-in on by the left box. There is a significant reduction in coverage for a contiguous region near the middle of the display, providing evidence for a ~10kb deletion. Downloaded from http://www.nature.com/nbt/journal/v29/n1/images/nbt.1754-F2.gif

Figure 2.16 Paired read visualization mode in IGV. In this mode, a line is drawn between paired read representations, and so the length of the line corresponds to the inferred insert size. Pairs of reads and the line connecting them are displayed on the same vertical level, and packed so that none of the pairs overlap.
usually one with expertise in genetics, examines the (possibly filtered) list of genetic variants in order to identify the one or few that are causing a patient’s disease, based on intersection with previously implicated genes or variants, or some other information (e.g. pedigree analysis). For this purpose raw read alignments provide too much information, and it suffices to show only the putative genetic variants. In 2012, IGV was augmented to support loading of VCF files and the visualization of variants. Variant tracks are visualized as a matrix, where rows represent samples and columns represent positions along the genome as shown in Figure 2.17. Where variants were identified in a sample, the corresponding cell is coloured according to its zygosity.

IGV is the closest related work to Savant Genome Browser, the work described in this chapter. The two software tools were developed simultaneously and often with collaboration between respective development teams at the Broad Institute and the University of Toronto.

2.2 Savant Genome Browser

Genome browsers have become integral parts of HTS analysis pipelines, where they are used to assess the reliability of computational predictions, validate findings in specific regions and guide refinement of automated tools. While a number of genome browsers have been developed for visualizing genome-based annotations, there are however several shortcomings in these tools.
leaving opportunity to improve the genome browsing experience especially with respect to
speeding navigation, aiding the visual discovery and validation of genetic variants, and
integration with realtime analytical tools and external datasets to complement visual inspection.

The UCSC and Ensembl genome browsers, for example, are popular online tools that
have traditionally been used to display various biological datasets, such as genomic variants,
expressed sequence tag (ESTs) and functional genomic data, in the context of high-quality,
manually curated annotations [47, 54]. While both have been recently updated to support
display of HTS data, their use for this purpose is not ideal because the server-side rendering
model upon which they are engineered creates performance limitations and privacy
vulnerabilities as previously discussed.

Other visualization programs such as the Integrative Genomics Viewer, Artemis and
Tablet [54-56] are designed to run on conventional desktop computers and thus can make use of
low-latency computing resources and secure local storage to overcome the shortcomings of web-
based browsers. While these popular browsers allow for interactive visualization of HTS data
they have limited analytic capability, neither are they extensible through the addition of
modules created by third-party developers.

Despite the proliferation of programs that deal with HTS data, most have been
developed for either automated (e.g. read mapping, genotyping) or visualization (e.g. genome
browsing) purposes, but not both. Yet visual and automated approaches are most powerful
when used together, such that users can seamlessly inspect and perform computation on their
data, iteratively refining their analyses. A substantial barrier for researchers who use genomic
visualization tools, for all types of data, is the disconnect between the processes of visualization
and computer-intensive analyses [57], a void which is caused by visualization tools being
programmatically inaccessible.

Lastly, the interaction and visualizations of genome browsers can be made more efficient
and informative through judicious use of the aforementioned information visualization
techniques: using preattentive processing, minimizing data-ink ratio, creating task-specific
visualization modes, and avoiding information overload through progressive disclosure. The
successful application of these techniques results not only in more aesthetically pleasing
visualizations but also in a more dynamic and goal-oriented application.
2.1.1 Features and Design Principles

The Savant Genome Browser has been developed since 2009 and has made contributions to the genomics research community that are both conceptual and practical: it has helped advance the field of genomic visualization by introducing new algorithmic techniques and visual representations, in addition to exemplifying a model by which visualization and analysis can be synergized; as a fully functional software tool, it has been widely accepted and used by the research community to examine and make discoveries from genomics datasets.

Key features that make these contributions possible include:

- allowing quick access to large locally stored HTS datasets
- accessing data securely, for example supporting authenticated data access
- being highly interactive, allowing for direct and indirect navigation and manipulation of data
- using state-of-the art algorithms for data storage and fast retrieval
- creating beautiful illustrations of data, making use of information visualization techniques: managing data-ink ratio, using progressive disclosure, exploiting preattentive processing
- creating novel visualizations for HTS datasets, particularly for visual identification and validation of read alignments and genomic variants
- creating a flexible plugin framework for third-party developers to create analytic or data retrieval apps
- being freely available

2.2.2 Versions

The first version of Savant Genome Browser was released in March of 2010 and published in Bioinformatics in June of 2010 [58]. It was one of the first genome browsers to enable fast visualization and navigation of reference genomes and corresponding genomic datasets — including HTS datasets. Moreover, it included a number of innovative features: it contained new navigation elements, supported progressive disclosure for applicable datatypes, had the ability to include externally developed plugins at runtime, and had novel visualization modes particularly for HTS read alignments.
The second version of the Savant Genome Browser was released in February of 2012 and published in Nucleic Acids Research in May of 2012 [59]. Savant2 advanced upon its predecessor and existing genome browsers by introducing a number of innovative visualizations and navigation interfaces, particularly for genetic variant datasets identified from HTS genotyping pipelines, and allowing for seamless integration of diverse external datasets. The plugin functionality was significantly expanded, and a number of analytic, visualization and datasource plugins were developed to exemplify the power of the new system. These plugins, developed by both the core Savant Development Team and by the greater Savant User Community, help synergize visual and automated analyses of genomic data. A summary of release and presentation timeline for this project is shown in Table 2.2.

The latest version of Savant Genome Browser is made available for download, with installers for Windows, Linux, and Mac, at genomesavant.com/p/savant/download and the source code is available at github.com/compbio-UofT/savant.

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009 October</td>
<td>Poster Presentation</td>
<td>Savant presented at Genome Informatics</td>
</tr>
<tr>
<td>2010 March</td>
<td>Major Release</td>
<td>Savant 1.0 released</td>
</tr>
<tr>
<td>2010 March</td>
<td>Poster Presentation</td>
<td>Savant presented at Genomic Disorders</td>
</tr>
<tr>
<td>2010 June</td>
<td>Publication</td>
<td>Savant published in Bioinformatics</td>
</tr>
<tr>
<td>2010 July</td>
<td>Oral Presentation</td>
<td>Savant presented at HitSeq</td>
</tr>
<tr>
<td></td>
<td>Poster Presentation</td>
<td>Savant presented at ISMB</td>
</tr>
<tr>
<td>2010 September</td>
<td>Oral Presentation</td>
<td>Savant presented at Genome Informatics</td>
</tr>
<tr>
<td>2010 November</td>
<td>Publication</td>
<td>Savant featured on the cover of Genome Research</td>
</tr>
<tr>
<td>2011 March</td>
<td>Poster Presentation</td>
<td>Savant presented at VIZBI</td>
</tr>
<tr>
<td>2011 May</td>
<td>Oral Presentation</td>
<td>Savant presented at McGill-Toronto Computational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Molecular Systems Biology Retreat</td>
</tr>
<tr>
<td></td>
<td>Poster Presentation</td>
<td>Savant presented at McGill-Toronto Computational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Molecular Systems Biology Retreat</td>
</tr>
<tr>
<td>2012 February</td>
<td>Major Release</td>
<td>Savant 2.0 released</td>
</tr>
<tr>
<td>2012 May</td>
<td>Publication</td>
<td>Savant published in Nucleic Acids Research</td>
</tr>
<tr>
<td>2013 November</td>
<td>Minor Release</td>
<td>Savant 2.0.4 released</td>
</tr>
</tbody>
</table>

Table 2.2 Significant publications, releases, and public presentations of Savant Genome Browser
2.2.3 Programming Language

The Savant client is coded in the Java Programming Language. The Java platform was chosen for the project because of its speed, security, extensibility, ease of development, and popularity in the bioinformatics community. A comparison of popular bioinformatics algorithms written in various programming languages found that Java implementations were consistently among the fastest [60]. Unlike many other compiled languages that can run across all major platforms (Windows, Linux, Mac) though, Java has an extensive Graphical User Interface toolkit that can be used to construct applications that look native to whichever platform it is running.

Savant was designed to quickly visualize very large datasets. The key to making this possible is having libraries to read and write binary compressed data formats that permit fast random access. A Java implementation of data compression and indexing of SAM/BAM is made available through the Picard library [61] and a more general toolkit for formatting generic tab-delimited files is provided by the Tabix framework [62]. A Java implementation of Tabix file reader existed, but not a writer. A Tabix writer was developed by the Savant Development Team and shared with the community.

2.2.4 Files and Formatting

Savant supports a number of common text-based formats, which are described in Table 2.3. However, because text-files do not enable fast random access, Savant formats and saves each file so as to provide very efficient search operations at runtime. The speed with which Savant can sift through large datasets is enabled by the way in which it formats and indexes data. In particular, formatting involves converting text records into an indexed binary data structure specific to each data type. Sequence and continuous tracks are stored as fixed-width records, enabling direct lookup of records of interest. Annotation ranges (such as genes) are stored using a binning scheme similar to the one used in the UCSC Browser [47] and in BAM files [25], so retrieving all ranges corresponding to some region usually requires only one, and at most O(log n) disk seeks, where n is the length of the chromosome. File formatting can be done directly through the application itself. Savant keeps its memory usage low by adjusting its sampling rate depending on the size of the visualized range.

The time and space requirements for the processes of data formatting and visualization were measured for a collection of human chromosome 1 datasets, including a genetic sequence,
genes, SNVs, mammalian conservation and alignments of sequenced reads from an individual from the 1000 Genomes Project (~40× coverage). The tests were performed on a Lenovo T61p laptop computer running Windows 7 with an Intel Core 2 Duo CPU at 2.40 GHz and 3.0 GB of RAM. The results are summarized in Table 2.4. Formatting of the gene, SNV, sequence and conservation tracks took less than 10 minutes total, while the computation of a coverage track from a large set of read alignments took about an additional 40 minutes. The latter conversion is optional, and allows for dynamic switching between an alignment view and a coverage view. Runtime performance was assessed by measuring the time taken to navigate to ranges of various sizes. Each measurement was performed on a newly started instance of Savant and the start location of the range was randomized. For seeking arbitrary ranges of sizes 10 million to 10 kb, Savant took less than a second to fetch and render data. The performance was worst for ranges just slightly shorter than 10 kb long, where the large number of BAM records that are displayed require 2s to be fetched from disk. Savant renders virtually instantaneously for regions having sizes on the order of hundreds of base pairs, where most fine-scale visualization is done.

2.2.5 Datasources
In addition to working with local files, Savant supports the use of remote (either network mounted or over the internet) files and datasources. All remote resources are cached locally, to enable rapid visualization upon re-loading of a previously visited region. While many commonly

<table>
<thead>
<tr>
<th>File Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTA</td>
<td>Standard format for nucleotide sequences</td>
</tr>
<tr>
<td>BED</td>
<td>Format for describing coordinates of localized features on genomes, such as genes</td>
</tr>
<tr>
<td>SAM/BAM</td>
<td>Standard format for read alignments</td>
</tr>
<tr>
<td>VCF</td>
<td>Standard format for genetic variants</td>
</tr>
<tr>
<td>WIG</td>
<td>Standard format for continuous-valued data. Useful for GC percent, probability scores and transcriptome data</td>
</tr>
<tr>
<td>GFF</td>
<td>General feature format for annotations of ranges</td>
</tr>
<tr>
<td>Tab delimited</td>
<td>Any tab-delimited file containing point, interval or continuous genome annotations</td>
</tr>
</tbody>
</table>

*Table 2.3 File formats supported by Savant Genome Browser. If not already done, Savant converts text-based files into a binary compressed format to allow for efficient random access of data.*
used datasets (e.g. reference genomes, genes) are accessible through a public repository, tracks can also be quickly loaded directly from the UCSC Genome Database, without a need for manual download, as described below.

### 2.2.6 Architecture

As a client-side rendering genome browser, the Savant client software is entirely responsible for coordinating navigation, data retrieval, and visualization. The software is comprised of several components which execute each of these tasks. A schematic of how these components work together is provided in Figure 2.18. Upon a navigation instruction being issued by the user, the target coordinates are received by the Navigation Controller. These are communicated to the Datasource Controller, in addition to being used to update other parts of the display (e.g. the

<table>
<thead>
<tr>
<th>Data files</th>
<th>Formatting</th>
<th>Retrieval and visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10M</td>
</tr>
<tr>
<td></td>
<td>File name</td>
<td>Size</td>
</tr>
<tr>
<td>chr1.genes.bed</td>
<td>1.1 MB</td>
<td>7.5K genes</td>
</tr>
<tr>
<td>chr1.snp.point</td>
<td>26 MB</td>
<td>1.5M SNPs</td>
</tr>
<tr>
<td>chr1.sequence.fasta</td>
<td>242 MB</td>
<td>249M nucs</td>
</tr>
<tr>
<td>chr1.conservation.wig</td>
<td>1.2 GB</td>
<td>249M vals</td>
</tr>
<tr>
<td>NA12878.chr1...bam</td>
<td>25 GB</td>
<td>262M reads</td>
</tr>
<tr>
<td>NA12878.chr1...cov</td>
<td>Computed from BAM file</td>
<td>1.2 GB</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>53m</td>
</tr>
</tbody>
</table>

*Tracks with continuous values are smoothed before rendering at large ranges. Sequence and point tracks are not rendered beyond certain ranges, denoted with a hyphen (–). Read alignments (BAM file) are replaced by coverage (precomputed from the BAM file) when visualizing longer regions.*

Table 2.4 Savant Genome Browser file formatting and retrieval performance for various data types. The Data Files section describes input files. The Formatting section shows time required to format each input file and the resulting formatted file size. The Retrieval and visualization section shows the time taken to retrieve data from ranges of various sizes and the number of records retrieved and drawn. All operations require <50MB of memory.
chromosome ideogram). For each track, the Datasource Controller communicates with the appropriate datasource to fetch only the data related to the target coordinates. Datasources may be remote (e.g. webserver), local (e.g. local filesystem), or implemented by a plugin. The only requirement of a Datasource is that it returns data of the appropriate type for the given range of coordinates. The Datasource Controller communicates track data back to the Rendering Controller — which renders the data to the display according to selected visualization modes — and to any Plugins that have subscribed to receive data updates. While data retrieval and rendering are the most computationally intensive processes these can each typically be achieved in fractions of a second to deliver smooth transitions between positions in the genome.

2.2.7 User Interface
Savant has a simple and intuitive interface, which is customizable through the use of a modular docking framework. Figure 2.19 shows a screenshot of Savant, and illustrates its various components, expanded upon in the following subsections.

DOCKING FRAMEWORK
Savant uses a modular docking framework, similar to those used in most Integrated Development Environments (IDEs). Each module within the application appears as a separate window that can be shown, hidden, maximized, minimized, resized, closed or rearranged in any configuration the user desires. Modules can also be detached from the main interface and moved.

![Figure 2.18](image)

*Figure 2.18 High-level architecture diagram for Savant Genome Browser. Numbers represent the order of communication upon receiving a navigation instruction from the user.*
to a separate location, which is useful for maximizing screen usage on setups having multiple displays. Savant includes a number of modules which are described in subsections that follow.

NAVIGATION

There are several ways a user can specify the genomic region to be displayed by the viewer. Coarse navigation is made possible through a range selection panel whose horizontal length represents the length of the loaded genome, from which subranges can be chosen using the mouse. Alternatively, fine navigation is possible by entering the desired genomic range into text fields. Neighbouring buttons enable zooming in and out and panning left and right. These

![Figure 2.19 Screenshot of Savant. (A) Range controls. Selection, zoom and pan controls for coarse navigation; text fields for fine navigation. Zooming and panning are also possible via keyboard and mouse commands. (B) Tracks. These represent the data in current range. Top: read alignments, with coloured pixels representing differences between the reads and the reference. Bottom: colour representation of the genome sequence. (C) Data View module, detached from the main interface. The Data View module is displaying the mapped reads with SAM format fields. (D) Bookmarks module.](image-url)
features are depicted in Figure 2.20. Each of these functions can also be engaged through mouse and keyboard shortcuts. Savant also uses a bookmarking framework to allow the user to switch between many regions of interest, as described lower.

**BOOKMARKS**
It is often useful to make note of interesting regions while using the browser, or to load a set of such regions in order to quickly navigate between them. As shown in Figure 2.21, the Bookmarks module helps keep track of these locations. Users may add, remove or seek to a bookmarked region by using buttons within the module or keyboard shortcuts. Furthermore, bookmarks may be tagged with a description and exported for future use or for sharing among colleagues.

**TRACKS**
A powerful use for genome browsers is the ability to visualize multiple datasets, which may be of the same or different types, simultaneously. All genome browsers based on the linearized coordinate system borrow from pioneering work on the Staden Package and Consed [63, 64], which organize multiple datasets into “tracks” that are vertically stacked and whose horizontal axes are aligned. These tools were developed prior to the advent of HTS technologies to assist in the finishing of reference assemblies, though the concepts introduced track layout, visualization, and navigation have persisted to modern-day browsers. In the track system, each track shows data of a single type, such as a genome, read alignment, gene set or generic annotation. A user
can specify the region of the loaded tracks to be displayed via the browser's many navigation controls, and the display of data within each track is updated accordingly.

Genome Browsers are tremendous tools for communicating datasets in both interactive and non-interactive settings. For instance, screenshots of genome browser tracks are often used in presentations and publications in a non-interactive setting. However, several visual elements need to be included to create fully self-described depictions of the data and these increase the complexity of visualizations and the user interface. Where possible Savant makes use of progressive disclosure to hide visual elements that detract from efficient visualization in the interactive setting. For example, by default axis markings are subtle and unobtrusive. This is contrast to axis markings in UCSC Genome Browser which are omnipresent and coloured starkly in blue and black. Since there are many horizontal lines visualized for gene and read alignment tracks, Savant does not additionally render a horizontal axis lines. Instead, Savant dynamically displays both the x- and y-coordinates of the mouse cursor in the lower left corner of the interface, an indicator that can be used to easily identify the position of any data element of interest. This information can be made more obvious through the activation of Crosshair Mode, shown in Figure 2.22, which displays x- and y-coordinates next to the mouse cursor. Similarly, a track's legend is shown only when it is selected, through clicking its title bar, and hidden otherwise.

In addition to the above-described techniques by which data is quantified through a subtle axis system, Savant’s colour scheme is significantly lighter and more lively than those of UCSC Genome Browser and IGV, which are visually dense and dull respectively. There is however a significant trade-off inherent in the production of colourful imagery in that these are

![Image](image.png)

**Figure 2.22** Crosshair mode in Savant Genome Browser (left) dynamically shows the x- and y-coordinates of the mouse cursor within tracks. Track legends and track-specific options (right) are displayed only when a track is selected.
significantly more costly to print and publish and there is a often dramatic difference between onscreen and printed versions. The colour scheme of Savant can be adjusted to account for this.

DATA VIEW
While Savant aims to provide the user with the ability to compute on the underlying data directly through a plug-in framework, in many cases the user may wish to identify the underlying data elements for export to an external program, e.g. identifying the genomic sequence within a window to align against another genome, or downloading all of the supporting reads for a SNV to make sure they do not align elsewhere in the genome. In Savant, the user can display the underlying textual data from any loaded track within the Data View. This module, shown in Figure 2.23, displays records as rows and fields as columns in a spreadsheet. For each read mapping, for example, the Data View displays the read name, mapped position, CIGAR string and other SAM fields. The data can be sorted in either ascending or descending order with respect to any field. The spreadsheet can also be exported for further analyses.

TRACK LOCKING
Individual tracks can be locked to a particular range so that they are not updated until they are unlocked. As exemplified in Figure 2.24, locked tracks can be used as overview profiles from which subregions can be selected to specify range changes for other tracks. Track locking also enables simultaneous viewing of high- and low-resolution profiles.

Figure 2.23 Data Table plugin displays records from tracks in a tabular format. For each read mapping, for example, the Data View displays the read name, mapped position, CIGAR string and other SAM fields. Selected elements are highlighted in green in both the track and the Data View.
2.2.8 Visual Encodings for Genomic Data

A challenge in developing a general purpose visualization tool like a genome browser is in designing graphical representations — i.e. how the data are encoded into shapes and colours at various scales — in a way that is suitable for a large number of datasets and purposes. In this section, unique visual encodings of genomic datasets that were developed for Savant are discussed.

VISUALIZING REFERENCE GENOMES

The genomic coordinates that are used to address genomic annotations are specified with respect to a specific reference genome. A reference genome must therefore be specified before any additional tracks can be loaded. Savant facilitates the loading of standard reference genomes for many organisms, including human, mouse, and yeast. Reference genomes are continually improved. The Genome Reference Consortium is a collective that actively improves and publishes new versions of the human reference genome (the latest being GRCh38 released in December of 2013). Upon the release of a new reference genome, all annotations (e.g. genes) must subsequently be readdressed.

In UCSC Genome Browser, the reference genome is encoded as a string of nucleotide letters. As discussed previously, however, text is not preattentive and so it is difficult to identify

![Figure 2.24 Track locking in Savant Genome Browser. The top track shows read alignments, locked at a region that is 4,000bp in size. The subsequent three tracks show the reference sequence, genes, and read alignments (from the same track as the first) at fine-grained resolution. Activation of the plumbline feature, which draws black vertical lines at corresponding coordinates on the x-axis, allows the user to synchronize positions across locked and unlocked tracks.](image)
patterns and outliers from a purely textual representation of the reference genome, as illustrated in Figure 2.25. Patterns in genomic sequence are not coincidental, they often have functional implications and their identification helps to explain other nearby annotations: for example, the genome is highly repetitive and these regions can be hotspots for recombination events, processes by which DNA molecules exchange genetic information [65]. In Savant the nucleotide sequence of the reference genome is encoded as a sequence of colours. This encoding preserves sequence information (and at high resolution both colours and nucleotide letters are shown) but it also makes patterns in genomic sequence easier to detect, since patterns in colour can be subconsciously perceived while patterns in text cannot.

**UCSC Reference Genome**

![UCSC Reference Genome](image)

**Savant Genome Browser**

![Savant Genome Browser](image)

**Figure 2.25** Visual representations of reference genome in UCSC Genome Browser (top) and Savant Genome Browser (bottom). The genome sequence being visualized in both is the same. Patterns in sequence are difficult to perceive from the textual representation compared to the colourful representation.

**VISUALIZING READ ALIGNMENTS**

HTS read alignment data are varied in quality, quantity, and distribution. Furthermore, the objectives of viewing these datasets are diverse: read alignment browsing can be used to assess the coverage of sequencing, and to assess the quality of read data, alignments, and genotype predictions. A number of task-specific visualization modes for read alignments are implemented in Savant and are herein described.

**Assessing Base and Mapping Qualities**

The processes of sequencing and alignment are prone to error and these are reflected in base and mapping qualities, respectively. In order to highlight reads and nucleotides that are of highest quality and therefore are most informative Savant users can choose to shade whole reads or individual positions based on these metrics. These visualization options make the disentanglement of high and low confidence supporting data simpler, and are illustrated in Figure 2.26.
**Visualizing Coverage**

Savant dynamically adjusts its resolution — the amount of information it displays — to optimize both nucleotide- and genome-scale visualization of tracks. This is demonstrated in Figure 2.27. Read alignments can be visualized as a coverage track on-demand or when the number of base pairs within a region is too large to enable the visualization of individual reads. Savant uses the technique of semantic zooming to seamlessly switch between coverage and read-alignment views. In addition to presenting a more intuitive visualization, this feature reduces the program's memory footprint and improves overall speed.

**Visualizing SNVs and Indels**

In the simplest terms the determination of whether a position is variable depends on the percentage of reads with differing nucleotides at the position. Instead of requiring the user to infer this percentage from raw sequenced reads, this can instead be visualized as per-nucleotide histograms in the SNP visualization mode. However HTS platforms also exhibit unique and reproducible biases that arise due to imperfect sequencing chemistries and/or library preparation. For example, strand-specific errors are common, and support from reads sequenced from both strands of DNA is often a requirement to separate true from false variants. In Savant, a user can choose to separate coverage profiles by strand within the Strand-SNP mode, making

![Image](image-url)

**Figure 2.26** Visualization modes and options for read alignments. The Standard Mode displays reads using a pointed arrow as in IGV without additional markings. In Mismatch Mode, positions within reads that are mismatched with respect to the reference are coloured. In either of these modes, indications of mapping quality or base quality can be shown. When activated, these options make reads transparent in a way that is proportional to the read or base qualities, respectively.
the identification of SNPs with support from both strands straightforward, as illustrated in Figure 2.28, while also highlighting issues pertaining to strand-specific coverage biases.

**Visualizing Structural Variants**

Structural variations that exhibit marked changes in coverage can be detected using the coverage visualization mode. However, not all structural variants can be identified from coverage alone, though paired-end information can often be used to identify and resolve breakpoints of such events with precision.

The standard representation of paired information as shown in Figure 2.16, where pairs of reads are connected by a line and piled in a similar fashion as in the visualization of individual reads, is supported by Savant and mimics that of IGV. In the paired-end-method, discordant mappings are identified by looking for pairs with anomalous inferred insert sizes (i.e. the distance between mapped reads) and abnormal orientations of component reads. In the standard read pair visualization, the length of the line is equal to the inferred insert size and reads are coloured when their mapped orientation is different from what is expected based on the sequencing method used to produce them. While the task of visually identifying clusters of reads that are discoloured is easy, identification of clusters of pairs with abnormal line lengths is not. This is because outliers are not encoded with visual features that make them salient enough to be identified by preattentive processing. This task is especially difficult to perform even attentively for heterozygous structural events, where half of the read pairs have normal mapped distances.

Savant introduces a novel mode for representing paired reads that shows arcs between the mapped locations of paired reads, where the height of each arc is proportional to the

---

**Figure 2.27** Read alignments, visualized at various resolutions and using two modes. (A) Chromosome-wide view of read mappings, showing the overall coverage (with no coverage in the centromere). (B) Regional view, still visualized as a coverage map, showing higher coverage in certain regions of the genome. (C) Local view, the reads are shown separately and differences between the reads and the reference genome are colored. Reads on the forward and reverse strand are shown with different shades of blue.
inferred insert size. This visualization is compared to the standard representation in Figure 2.29. Arcs for anomalously mapped pairs, such as those suggestive of inversions or duplications, are coloured differently. In this way, the task of visually identifying clusters of discoloured and/or discordantly sized lines is made to be preattentive, significantly reducing the cognitive load of identifying structural variants through the paired-end method. Arc visualization of pairs combined with the corresponding coverage plot makes structural rearrangements easily interpretable as shown in Figure 2.30, where absent read coverage indicates a deletion in the genome, which is confirmed by discordant (over-stretched) read-pairs joining the two ends of a deletion.
VISUALIZING PREDETERMINED GENOTYPES

The decreased costs and increased throughput of HTS have enabled the sequencing of large cohorts, both from specific disorders and general populations [66, 67]. Genome browsers, in turn, need to support visualization of data that has been agglomerated from many genomes. Savant has comprehensive support for multi-individual genotype datasets in the standard VCF format. However the visualization of variation data introduces additional resolution complexities, as SNPs typically appear every 100–1,000 bp, depending on the number of individuals and the locus, but can also appear at adjacent nucleotides. Thus, viewing SNP data over a 10 kb region would require drawing each SNP as less than 1 pixel, to maintain the genome scale and prevent overlaps between adjacent SNPs. Savant introduces original visualizations and navigation interfaces tailored for the efficient perusal of genotyped cohorts. Data from a genetic variant dataset is shown in two areas of the Savant interface: in the main genome-scale track browser, as well as in a new navigation component that agglomerates data from larger areas of the genome.
 Variant tracks are visualized as a matrix, as shown in Figure 2.31, where each row represents an individual or sample from the file and each column represents a genomic position that is vertically aligned with the rest of the tracks. Each cell in the matrix is coloured according to the non-reference nucleotide in the corresponding sample and position, or is transparent if the allele is reference. Because it is difficult to identify trends from bands of mixed colours when datasets contain data for many samples, as in the 1000 Genomes Project, a summative view is also provided by an Allele Frequency mode, which shows the frequency of each allele per position in a way that is analogous to SNP mode for HTS read alignments.

The Variant Navigator interface, shown in Figure 2.32, is located on the right side of the browser and is used to display variant data on an independent scale, allowing the visualization of SNPs over larger segments of the genome. In the simplest mode, the Variant Navigator lists variants in text or visual form. Within the same window users can also perform case–control analysis by assigning samples to one of two cohorts, and visualizing the respective allele frequencies distributions at the variable positions. Linkage disequilibrium, a measure of allelic
correlation across variant positions, can also be computed and visualized with this component as done in Haploview [68].

2.2.9 Plugins

The Savant Genome Browser is further extensible by plugins. These plugins can conduct computations using currently loaded data, visualize results and navigate the browser to regions of interest, all while utilizing external datasets. The Savant Application Programming Interface (API) provides plugins with extensive

- visualization functionality: to display graphics that are either superimposed on top of tracks or in a separate reserved space, whose visibility can be toggled.
- analytic functionality: to perform computation on and manipulation of the data. If the computations are fast, plugins can visualize the results in realtime alongside track navigation. Otherwise, they may load the results as a track upon completion.
- navigation functionality: to provide interfaces for quickly loading genomic regions of interest. This is particularly useful for using external datasets (e.g. a list of genes) for guiding genome navigation.
- datasource functionality: to enable retrieval of track data from alternate data sources. This functionality is useful for loading tracks directly from public or private databases, or from external programs.

Each plug-in can be one of two general types. Interactive plug-ins are allocated dockable modules on which graphical user interface (GUI) elements such as buttons or text fields can be placed to respond to user input and retrieve data. Non-interactive plug-ins are not designated a
GUI component but still have extensive access to the innards of the browser. Plug-ins can be used, for example, to prototype a SNP finder by identifying variable columns currently in view, or for computing genome-wide statistics, such as the fraction of SNPs in exons.

**PLUGIN API**

Plug-in development is straightforward and requires implementation of one Java interface. The Bookmark Intersection Plug-in, shown in Figure 2.33, is an example of an interactive plug-in. The plug-in allows the user to select two tracks, intersect them and load the intersecting regions into the list of bookmarks, enabling easy navigation to all of the regions of interest. A downloadable Software Development Kit (SDK) is included on the project webpage and includes

![Figure 2.32 Visualizations of genetic variant data. (A) A view of the Allele Frequency page of the Variant Navigator, which compares allele frequencies of genetic variants from two cohorts from the 1000 Genomes Project. At most positions the frequencies are similar between cohorts, though there are positions that exhibit different frequencies. (B) An LD plot of variants in the same range as on the left. Blue and red cells represent low and high correlation between variant positions, respectively.](image)
public class BookmarkIntersectionPlugin
    implements InteractivePlugin {

    // set by UI
    GenericIntervalTrack sourceTrack, targetTrack;

    RangeController rc;
    BookmarkController bc;

    // constructor
    public BookmarkIntersectionPlugin(JPanel p,
            SavantPluginAdapter pluginAdapter) {
        /* UI Setup Code:
        * Create a UI in JPanel to select a src and a 
        * target track. Details skipped for simplicity
        */
        rc = pluginAdapter.getRangeController();
        bc = pluginAdapter.getBookmarkController();
    }

    /* called on button press */
    public static void bookmarkTrackIntersections() {
        List sourceIntervals =
                sourceTrack.getRecords(rc.getMaxRange());

        for (GenericIntervalRecord r : sourceIntervals) {
            Range sourceRange = r.getInterval().getRange();

            List targetIntervals =
                    targetTrack.getRecords(sourceRange);

            if (targetIntervals.size() > 0) {
                bc.addBookmark(new Bookmark(sourceRange,
                                targetIntervals.getDescription()));
            }
        }
    }

Figure 2.33 Code used to make Bookmark Intersection Plug-in. The details of the UI that allows the user to select two tracks have been omitted. Once the two tracks are selected, the bookmarkTrackIntersections() method is run, which, for each interval of one track, finds overlapping intervals of the other, and saves intervals with overlap to the bookmark panel.

source code for sample plugins and a full documentation of the rich API enables the development of such plugins by the user community.

PLUGIN REPOSITORY

New plugins can be contributed to a public repository, and are made available for download to all users through Savant's built-in Plugin Manager. Table 2.5 summarizes a selection of Savant Plugins, some developed by the core Savant Development Team, and others by external users. Several of these are explained in more detail in the following sections.

UCSC PLUGIN

Since Savant runs on client machines instead of on a centralized server it has the important advantage of maintaining complete privacy of sensitive data, such as the genomes of specific
patients. Nevertheless, it is often necessary to examine this data in the context of the wealth of publicly available genomic information. The UCSC Genome Database is perhaps the most extensive repository of this type and contains the underlying data for all available tracks displayed on the UCSC browser. The UCSC Explorer, shown in Figure 2.34, is a plugin that makes it possible to open UCSC tracks within Savant without downloading the raw data files—

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromatogram</td>
<td>Shows Sanger sequencing chromatograms overlaid on the reference genome</td>
</tr>
<tr>
<td>edgeR</td>
<td>Detects and visualizes differential enrichment from RNA-seq or ChIP-seq data</td>
</tr>
<tr>
<td>Data table</td>
<td>Shows textual data from track records in tabular form</td>
</tr>
<tr>
<td>GATK</td>
<td>Predicts and visualizes genotypes from read alignment tracks</td>
</tr>
<tr>
<td>Gene ontology</td>
<td>Guides navigation based on ontology terms</td>
</tr>
<tr>
<td>PING</td>
<td>Guides navigation based on protein–protein interaction databases</td>
</tr>
<tr>
<td>Remote commander</td>
<td>Issue navigation and other commands remotely through external tools</td>
</tr>
<tr>
<td>Ribosome</td>
<td>Shows the amino acid translation of gene tracks</td>
</tr>
<tr>
<td>RNA-seq analyzer</td>
<td>Reconstructs and estimates isoforms from RNA-seq data</td>
</tr>
<tr>
<td>Snapshot</td>
<td>Exports track images at every bookmarked genomic region</td>
</tr>
<tr>
<td>SNPFinder2</td>
<td>Predicts and visualizes SNVs from read alignment tracks</td>
</tr>
<tr>
<td>SRMA</td>
<td>Realigns HTS read alignments</td>
</tr>
<tr>
<td>UCSC explorer</td>
<td>Provides a graphical interface for loading UCSC tracks</td>
</tr>
<tr>
<td>WikiPathways</td>
<td>Guides navigation based on biological pathways</td>
</tr>
</tbody>
</table>

Table 2.5 A list of some plugins available for the latest version of Savant.

Figure 2.34 UCSC Explorer Plugin makes it possible to open UCSC tracks within Savant directly without downloading any data files.
only the relevant data is downloaded via a direct connection to the UCSC Database, and is immediately presented to the user. The tracks are categorized in a manner that mimics their presentation within the popular web-based UCSC Genome Browser.

**RNA-SEQ ANALYZER PLUGIN**

RNA sequencing has been transformative in transcriptomics — the study of RNA and their functions — by simplifying the process of determining the identity and abundance of isoforms — different transcripts that are produced from the same genetic locus — within the cell. The plugin, shown in Figure 2.35, accepts input from previously reconstructed isoforms, through programs such as Cufflinks [69], or alternatively performs isoform reconstruction and abundance estimation from the set of read alignments directly using the iReckon algorithm. The plugin overrides the default coloring scheme in Savant and instead colors each read according to the most probable isoform from which it was generated. For any gene of interest, a multi-coverage profile is provided for comparing read support for each isoform. A pie chart summarizing the relative proportions of the isoforms is also provided. Finally, the plugin can incorporate two datasets simultaneously, and allows for their comparison by visualizing the differences in expression. This work was published in Genome Research in 2013 [70].

![Figure 2.35 The RNA-seq Analyser Plugin predicts the identity and abundance of gene isoforms from RNA-seq data. Each isoform it predicts is assigned a unique colour, and reads are coloured according to which isoform they are most likely to have been generated from.](image-url)
**EDGER PLUGIN**

The analysis of quantitative HTS data (e.g. from RNA-seq or ChIP-seq) relies on statistical procedures that highlight differential regions. For example, the density of mapped reads in a particular genomic region may represent enrichment level of a protein–DNA interaction (ChIP-seq), or gene expression level (RNA-seq). The edgeR plugin is a wrapper for software written in the R statistical programming language for the detection of significantly differentially enriched regions or expressed genes, relative to observed biological variation, directly within Savant [71]. The plugin computes on multiple BAM tracks, some designated as Case and others as Control, and provides a table of ranked results, including the region locations, log-fold-changes, P-values and estimated false discovery rates of the change between conditions.

**WIKIPATHWAYS PLUGIN**

WikiPathways is an open collaborative platform for the curation of biological pathways [72]. The WikiPathways plugin, shown in Figure 2.36, provides an interface to search, browse and visualize the over 1500 pathways available from this platform, and to use pathways to guide navigation to relevant genomic locations within Savant. The use of functional annotations for navigation through large genomes represents a significant departure from existing navigation techniques, which are almost entirely based on linear scanning.

**PING: PROTEIN INTERACTION NETWORK TO GENOME PLUGIN**

To better understand the functional consequences of sequence variants it is necessary to look beyond the genome, to gene products and their interactions, especially when dealing with complex (i.e. non-monogenic) diseases. Given a query gene, the PING plugin allows one to view the partner genes (interactors) that engage in known protein–protein interactions, mapped across the genome. The program provides hyperlinks to further information: to Entrez for information about each interactor, to iRefWeb [73] for query-interactor information and to DAnCER [74] for gene annotations including associations to disease and GO [75].

**SNP FINDER2**

High Throughput Sequencing technologies have enabled the sequencing at single basepair resolution, which can be used to reliably detect SNPs and other genetic variants. However, genotyping algorithms need to be fine-tuned to the specific sequencing technologies whose data is being used, often requiring many iterations of running. The SNP Finder Plugin was developed to enable realtime adjustment of genotyping parameters and visualization of results. This plugin
The SNP Finder predicts the existence of a SNP, the most likely alternative allele, and its zygosity. Upon discovering a SNP, the plugin displays a vertical bar on the corresponding BAM track at the location of the variant as shown in Figure 2.37. Moreover, it adds a bookmark to the Bookmarks component of Savant with helpful statistics (e.g. the amount of support for the alternate allele).
APPLICATION PLUGINS

A large number of computational tools have been developed for analyzing HTS data but nearly all require command-line invocation and have file-based input and output. These programs can be easily chained together and run on large datasets, however such convenience hampers the ability to efficiently fine-tune their parameters and severely restricts their use mainly to computational specialists. The SSDK has been expanded to support the incorporation of a wide array of genomic tools within Savant, thereby unlocking opportunity for performing many computational analyses within a powerful visual environment. Now, with a minimal amount of effort—namely, specification of the program's input, output and parameters—virtually any command-line tool that computes on genomic data can be incorporated as a plugin within Savant, and its results rendered as a track immediately upon completion. This functionality is similar to that provided by Trackster [76] as part of the larger Galaxy framework [77], the key distinction being that Galaxy is a web based, server-side package, whereas Savant is client-side. To illustrate this ability we have built wrappers for two popular applications: the Unified Genotyper of the Genome Analysis Toolkit (GATK) and the Short Read Micro Realigner (SRMA).

Figure 2.37 The SNP Finder plugin can be used to genotype from any read alignments track loaded into Savant. It has parameters to adjust the sensitivity of the genotyping algorithm, and for the opacity of the rendered results.
GATK plugin

GATK predicts SNPs and indels from HTS read alignments [26]. While this genotyper is modelled to account for technology-specific biases automatically, it is still highly tuneable, allowing users to carefully adjust the sensitivity of the underlying detection algorithm. The GATK plugin is an XML specification of the input, output and parameters of the Unified Genotyper as shown in Figure 2.38. These parameters are specified within Savant, which invokes the genotyper, and subsequently visualizes the resulting VCF file. The plugin can quickly compute and visualize genotypes for a segment of the genome, allowing for rapid and dynamic experimentation with program parameters, prior to invoking the tool on a whole-genome scale.

SRMA plugin

Read alignment tools consider each read independently. In the absence of additional information the precise positioning of indels within a mapped read is difficult, particularly towards the ends of reads where sequencing quality tends to deteriorate. SRMA is a tool that performs realignment of previously mapped reads based on a local consensus [78], with the aim of sharing information across reads so as to remove false positives and properly place aberrantly positioned variants. Like the GATK plugin, the SRMA plugin is a wrapper around this command-line tool that facilitates its running within the visual environment of Savant, making possible real-time realignment of track reads and further deployment on a whole genome scale.
2.2.10 Evaluation

Savant has been developed since 2008 and in that time has been refined through the release of numerous major and minor versions. The software has been downloaded over 6,500 times globally since download tracking began in 2010, and the latest version has been downloaded over 1,000 times. A summary of downloads for major versions of Savant is provided in Table 2.6.

Savant is actively used and is one of the most popular genome browsers for visualization of HTS datasets. It is most used in the United States, Canada, United Kingdom, Germany, and China. Figure 2.39 displays the global distribution of users by country.

The features of Savant have been collected and implemented through processes of user centred design. We have performed participatory design, surveys, workshops, and informal interviews to solicit user requirements and feedback. User requirements were collected from discussions with close collaborators, especially those at The Centre for Applied Genomics in Toronto. User requirements were encoded into tickets, prioritized, implemented, as included in internal versions for testing by early adopters.

Savant has been taught as part of the visualization module of the HTS Data Analysis Workshop offered through the Canadian Bioinformatics Workshop Series every year since 2010. The students include graduate-level researchers, principle investigators, and clinicians having a broad range of expertise in genomic data visualization. This venue has been instrumental in evaluating the relevance and functionality of Savant, which has generally been reviewed positively as shown in Figure 2.40.

**USAGE STATISTICS**

A feature that collects anonymous usage statistics was introduced in version 2.0.4, released in November 2013. The feature, which can be turned on or off through Savant preferences, tracks basic information such as operating system, Java version, session times, track types opened, and so forth. The following table summarizes the data collected:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date Range</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Versions (1.0+)</td>
<td>November 12 2010 - September 7 2014</td>
<td>6,546</td>
</tr>
<tr>
<td>First Major Version (1.0 - 2.0)</td>
<td>November 12 2010 - February 21 2012</td>
<td>2,082</td>
</tr>
<tr>
<td>Latest Major Version (2.0+)</td>
<td>February 22 2012 - September 7 2014</td>
<td>4,464</td>
</tr>
<tr>
<td>Latest Version (2.0.4)</td>
<td>November 12 2013 - September 7 2014</td>
<td>1,336</td>
</tr>
</tbody>
</table>

*Table 2.6 Downloads of Savant Genome Browser since download tracking began, the last major version was released, and the last minor version was released. Data collected on September 7 2014.*
visualization modes used, etc. The collection of such detailed usage information reveals important insights about users and their behaviour and their preferences that would be

**Informatics on HT-Sequencing Data workshop, 2012**

![SurveyMonkey](image)

Each of the following content areas was relevant to me and will be useful in my work:

<table>
<thead>
<tr>
<th></th>
<th>Don't know</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neither Agree Nor Disagree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>Response Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Savant</td>
<td>5.4% (2)</td>
<td>0.0% (0)</td>
<td>0.0% (0)</td>
<td>10.6% (4)</td>
<td>40.5% (15)</td>
<td>43.2% (16)</td>
<td>37</td>
</tr>
</tbody>
</table>

**Figure 2.39** Downloads of Savant Genome Browser by country since download tracking began in November 2010. Top five countries are United States, Canada, United Kingdom, Germany, China. Data collected on September 7 2014.

**Figure 2.40** Survey results from 2012 HTS Data Workshop delivered as part of the Canadian Bioinformatics Workshop Series. Results for 2013 and 2014 are not provided, as for these years the visualization module of this workshop included multiple tools.
impossible to discover as quantitatively through other methods of user engagement. The following insights have been acquired through analysis of 5,511 recorded Savant sessions.

**Platform**
The user community for genome browsers is diverse. It includes bioinformaticians, technicians, and clinicians among others. Since there is no operating system that is overwhelmingly preferred by members of each group, let alone as a whole, Savant was developed using Java to facilitate distribution across multiple popular platforms. Analysis of usage statistics, summarized in Figure 2.41, confirm that there is significant use of Savant on all major platforms.

**Navigation**
A diverse set of navigation controls are provided in Savant to support fluid navigation around the genome. These controls were reviewed previously, and include mechanisms for coarse-and fine-grained navigation. Usage statistics, shown in Figure 2.42, confirm that users tend to use coarse-grain navigation commands (e.g. genomic coordinate system) followed by iterative use of fine-grained navigation commands (e.g. zooming and panning). By frequency of use, panning is the most commonly used navigation method. Moreover, users are about equally likely to utilize controls on the navigation bar as their are to use direct manipulation controls via mouse drag and scroll.

**Data Type and Visualization Modes**
Savant was primarily designed for the visual exploration of HTS datasets, though it can be used for visualization of various data types. As shown in Figure 2.43, analysis of track types opened in Savant suggests that it is in fact used mostly for visualization of read alignment datasets.
Besides sequence tracks, which must be opened before loading any other tracks, general interval annotations (e.g. loaded from GFF) are popular tracks visualized by the software. Though support for variant visualization was significantly augmented in Savant2, visualization of this type of data is relatively infrequent.

As described previously a number of visualization modes have been developed, particularly for read alignment tracks. As shown in Figure 2.44, popular visualization modes for read alignments include the Read Pair (Arc), SNP/Strand SNP, and Mismatch. Note that Mismatch mode is the default visualization mode for this track type, and since the usage
collection feature only records when a user explicitly selects one from the Dropdown Menu, its overall use is underrepresented in this figure.

SUPPORT & COMMUNITY
Support for Savant is provided in three ways. First, if Savant encounters an error, a user is prompted to send details which are sent to the development team by email. Second, users are encouraged to send issues and other forms of feedback via email to support@genomesavant.com. Finally, there is an online community where users can ask questions and provide responses publicly. Since its inception in 2010 the community has grown to include over 140 members.

2.3 Summary
Despite the proliferation of automated tools for computation on HTS data, human interpretation is still necessary for its analysis. A lack of tools that support visual analysis of HTS data has precluded efficient interpretation of data, especially for biologists without significant informatics expertise. The Savant Genome Browser is a desktop visualization tool that is developed for genome researchers of all levels of computational expertise that supports viewing of tracks stored locally and remotely; is fast, allowing navigation with minimal rendering times; is secure, not uploading sensitive information to the internet; supports the tasks of assessing base and read alignment qualities and performing visual variant calling and assessing predictions made by external algorithms using novel visualization modes; and encourages synergy between visualization and analytic components through its plugin.
framework. As one of the first and most popular genome browsers for visualization of HTS datasets, Savant represents an important contribution to the genome research and visualization community.
3  Interpreting Personal Genomes
Information visualization has been of monumental importance throughout the history of genetic research, as detailed in Figure 3.1. In the 1800s, Charles Darwin used trees drawn with pen and paper to map the evolutionary relationships between species. In the 1900s, James Watson and Francis Crick used x-ray crystallographic imagery to discover the helical nature of DNA molecules. In the early 2000s, the first genome browsers empowered researchers to look at genomes at a nucleotide-scale. Genome Browsers empower researchers to visualize a wealth of information in the context of a linear reference genome; many have been created which exhibit incremental improvements to the basic model. Yet fundamental issues with the current model remain that have yet to be satisfactorily addressed. Genome browsers:

- compare datasets to a single reference genome
- don’t encode spatial, temporal, or epigenetic information well
- do not scale for multi-genome analysis
- cannot be used to search for regions of interest (besides via chromosomal coordinates)

Because genome browsers are separate tools from the analytic processes that generate their input — referring to both the data being visualized and regions of interest to be explored — genome browsing has not achieved the same level of integration as other methods of large data exploration, like web browsing. An important realization is that the results being explored in genome visualization tools are often not generated dynamically based on criteria that are interactively specified by the person who is interpreting them (e.g. a geneticist) but rather these are usually loaded from static files that are the output by a combination of command-line

**Figure 3.1 History of Genome Visualization.** In the 1800s, Charles Darwin used trees drawn with pen and paper to map the evolutionary relationship between species. In the 1900s, x-ray crystallographic imagery exposed the helical nature of DNA molecules to James Watson and Francis Crick. In the 2000s, HTS genome browsers have empowered researchers to look at genomes at a nucleotide-scale. Now, there is need for tools that can be used to efficiently explore many genomes simultaneously.
utilities executed by another individual (e.g. a bioinformatician). Regions of interest (e.g. locations of putative disease-causing genomic variants, splice sites, promoter regions, etc.) are often generated in this way, and are encoded using the chromosomal coordinate system. While genomic coordinates carry little or no meaning themselves, they are used manually by the user to connect analytic and visualization tools together, mostly by copy-and-paste. The use of genomic coordinates as a currency to exchange information between tools is a significant usability block that can be likened to browsing the internet through the use of URLs prior to the advent of search engines like Altavista, Yahoo, and now Google. Search engines revolutionized web browsing by making it possible to search the web using natural language, to obtain search results quickly, and to be able to explore them in a highly integrated way. These features have increased both the precision and efficiency at which people of all levels of computation expertise can browse the internet. In order to meet the throughput required to realize the promise of genomic medicine, tools for genomic data exploration need to undergo a similar evolution.

While the process of obtaining an individual's genome has become routine and inexpensive, its interpretation remains a significant challenge and costly as a consequence [79]. In the context of disease sequencing, for example, finding the one or few genetic variants that are causal among the potentially millions identified in an individual is a complicated process that currently requires many different tools and people. A large number of specialists are required to: annotate genomic variants; filter these for high-quality, clinically relevant, and plausible (e.g. given inheritance patterns); queue these for validation with Sanger sequencing; and communicate findings back to the attending physician. The provision of actionable information for a patient further requires considering the intersection between his or her genetic profile with available medical treatments and lifestyle interventions.

The task of genetic variant analysis in a disease sequencing study is especially laborious because a large proportion of variants identified through HTS sequencing analyses are irrelevant, because they are either not real (due to errors in the prediction pipeline) or are real but have no functional relationship to the condition of interest. One of the most challenging problems is thus in identifying the few genetic variants that are actually causal in disease.
Discovering disease-causing genetic variations is mostly exploratory and cannot currently be performed exclusively using either automated or visual tools. The process, summarized in Figure 3.2, usually involves several iterations of guess-and-check, where investigators apply filters to variants on any number of attributes (e.g. genomic region, quality, population frequency, predicted harmfulness, etc.) and then assess the relevancy of the results, which guides further adjustment of filters.

The next generation of genome visualization tools ought to facilitate data exploration (particularly the task of identifying casual genetic mutations) in a more dynamic, integrated, and scalable manner than is possible using traditional model of genome browsing. At the onset of this work, in 2011, there was significant need for a system that could:

- support storage of large volumes of genetic data, particularly genetic variations
- annotate genetic variants with datasets that help in their interpretation
- host these data in a way that is secure and enforces data integrity and minimizes redundancy
- enable interactive querying with conditions that are based on various facets of the data, especially those that help retrieve high-quality and relevant variants in disease-sequencing settings

Figure 3.2 Schematic of variant filtration. Filtration of common variants and those with no obvious effect on protein are simple and effective filters. The processes of iterative biological interpretation and assessing for biological relevance and actionability require careful analysis and significant expertise. Downloaded from https://www.ingenuity.com on September 17 2014.
• do so in a way that is obvious to users without computational expertise (e.g. use natural language instead of scripts with mathematical formulae or shorthand)
• integrate with external knowledge bases and visualization tools (preferably in such a way so as to minimize copying-and-pasting of genomic coordinates between tools)

3.1 Related Work
The remainder of this section surveys related work in this area, before describing a system for interpretation of personal genomes.

3.1.1 Variant Visualization
The traditional model of displaying one sample or annotation per track in a genome browser has quickly become obsolete. Both IGV and the Savant Genome Browser have been retrofitted to support the VCF file format, a standard for the storage of SNVs and short indels accrued from multiple samples [59, 80]. IGV and Savant both visualize genetic variation datasets as a matrix within a single track, where each row represents an individual or sample from the file and each column represents a genomic position that is vertically aligned with the rest of the tracks. Cells in the matrix are coloured — in Savant, using the same colour scheme as mismatches in read alignment tracks — where variants exist in a sample, as shown in Figure 3.3. It is difficult to identify trends from the matrix when datasets contain information from a large number of samples, so Savant also provides an alternative view that illustrates the frequency of each allele per position across all samples as a stacked bar chart. This view provides valuable information about the rarity of a specific mutation within a population, which indirectly serves as one metric for its harmfulness.

![Figure 3.3](image_url) Visualization of genetic variants in IGV. The space is configured into a matrix, where rows are samples and columns are positions. However the display of variants along the linear axis does not afford a useful comparison between polymorphic positions, because genomic variants are sparsely distributed.
Despite providing valuable information within a polymorphic position — i.e. a genomic position that tends to be variable among individuals — the display of variants along the linear axis does not afford a useful comparison between polymorphic positions, because genomic variants are sparsely distributed. In human genomes the SNV rate is roughly 1 per 1,000 nucleotide positions. Savant succinctly displays variants on a secondary canvas that has an independent non-linear axis. In its most basic mode variants are encoded in a matrix akin to their display in tracks, but with the intervening space between variable positions removed. This compact view makes correlations between nearby variant positions more apparent, especially through linkage disequilibrium (LD) plots as discussed in the previous chapter.

Still, neither of these genome browsers inherently enables fast search or filtration functionality for large variant datasets, or any other datatype. Even retrofitted genome browsers including IGV and Savant thus must be used in combination with some other analytic tool that suggests regions of interest for users. This is performed by iterative filtration of genomic variants based on restrictions placed on various facets of the data and their annotations (e.g. allele frequency, predicted functional effect, intersection with known disease genes).

![Faceted search interface for eBay Motors (left) and Amazon (right). The interface layout — with search criteria displayed in sections on the left panel and results on the right — is commonly used for iterative search of multi-faceted data, which may be quantitative (e.g. display size) or qualitative (e.g. exterior colour).](image)

Figure 3.4 Faceted search interface for eBay Motors (left) and Amazon (right). The interface layout — with search criteria displayed in sections on the left panel and results on the right — is commonly used for iterative search of multi-faceted data, which may be quantitative (e.g. display size) or qualitative (e.g. exterior colour).
3.1.2 Variant Search

Faceted search – i.e. exploration through the application of combinations of filters – is commonly implemented for e-commerce websites such as eBay, Amazon, and Kayak to allow visitors to progressively hone in on products of interest. A common user interface for faceted search, as exampled in Figure 3.4, has tuneable widgets for distinct facets of the data. Allowable values for qualitative features tend to be small in number, and presented as checkboxes, whereas quantitative features can be selected from a range slider. A number of software tools — some visual and others command-line — are available for faceted search (or filtration, colloquially) of genetic variation datasets. New tools in this category are being developed both in academic and commercial spheres. The following are some of the most popular and foundational and are summarized in Table 3.1.

**GENOME ANALYSIS TOOLKIT**

The Genome Analysis Toolkit (GATK) is developed at the Broad Institute and comprises a suite of genetic analysis tools for read (e.g. base quality recalibrator, indel realigner, read pileup) and variant (e.g. genotyper, haplotype caller, variant filtration) analysis [26]. GATK contains a Variant Filtration component that takes as parameters a VCF file and an arbitrary number of filter expressions that are constructed using shorthand notation. For example, the filter expression “DP < 5” is meant to remove variants where the value of the DP field, which

<table>
<thead>
<tr>
<th>Platform</th>
<th>Cost</th>
<th>UI</th>
<th>Natural Language Queries</th>
<th>Boolean Logic in Queries</th>
<th>Shared Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATK</td>
<td>Command-line</td>
<td>Free for academic use</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Microsoft Excel</td>
<td>Desktop</td>
<td>Paid</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>VarB</td>
<td>Desktop</td>
<td>Free</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>VarSifter</td>
<td>Desktop</td>
<td>Free</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (but not default)</td>
</tr>
<tr>
<td>Golden Helix SVS</td>
<td>Desktop</td>
<td>Paid</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Ingenuity Variant Analysis</td>
<td>Web</td>
<td>Paid</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GEMINI</td>
<td>Command-line</td>
<td>Free</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 3.1* A comparison of popular, foundational tools for faceted search and interpretation of genetic variants.
represents the combined depth across samples, is less than 5. Filter expressions must be specified in Java Expression Language, or JEXL, a format that supports boolean logic and mathematical operations. The language for specifying expressions in JEXL is expressive but difficult to write for casual users as it requires input of short, cryptic, and case-sensitive terms. For example, DP4 is a term that can be used to represent the sum of the number of (1) forward ref alleles, (2) reverse ref, (3) forward non-ref, (4) reverse non-ref alleles, used in variant calling. DP4 can be smaller than DP because low-quality bases are not counted. GATK Filter expressions can be given a name and in the previous example “Low Coverage” would be appropriate. An example command that can be issued to GATK, demonstrating the specification of several filter expressions, is shown in Figure 3.5. Querying for variants with respect to specific samples is more complicated, requiring use of functions and a separate sample name flag. For example, to find all novel, non-reference variants in sample named 1101 with allele frequency < 0.25, the JEXL expression would be:

```java
!vc.getGenotype("1101").isHomRef() && (vc.getID() == null || vc.getID().equals(".")) && AF < 0.25 && vc.isSNP() -o variants.vcf -sn 1101
```

Figure 3.5 GATK Variant Filtration command. Filter expressions must be specified in Java Expression Language, or JEXL, a format that supports boolean logic and mathematical operations. GATK Variant Filtration outputs all variants into a VCF file, with passing variants annotated with a PASS key included in the INFO field.
GATK Variant Filtration outputs all variants into a VCF file, with passing variants annotated with a PASS field included in the INFO field. Because of the size and format of these files, they are difficult to view manually, and so are usually parsed through a subsequent step to extract only passing variants. If the results are to be viewed by a non-bioinformatician, extracted variants are sometimes reformatted as tab-delimited or comma-separated files so that they can be opened in standard spreadsheet software like Microsoft Excel.

**MICROSOFT EXCEL**

Microsoft Excel is a graphical spreadsheet application developed by Microsoft for Windows and Mac and is included as part of Microsoft Office. It features calculation, graphing, pivot tables, and is programmable with macros through the Visual Basic Programming Language. It is an industry standard for exploring tabular datasets and is often the program of choice for examining appropriately formatted variant datasets by genetic researchers who lack command-line expertise.

Genetic variants can be converted into a tabular format compatible with Microsoft Excel using a utility called vcflatten [81]. Alternatively ANNOVAR [82], one of the most popular tools that performs variant annotation, also outputs its results in tab-delimited format. Resulting tables usually arrange variants into rows, and display annotations (either from the VCF INFO field or from external datasets) as columns. An example Excel document containing variant information is shown in Figure 3.6. Once loaded into Excel a variant dataset can be searched, sorted, and filtered. The ability to perform column-specific filtration is particularly useful. For any feature, a user can specify allowable values from among a list or by specifying boolean rules (e.g. “starts with” for string values or “greater than” for numeric ones). Specification of such rules is guided by graphical widgets using natural language.

General-purpose graphical spreadsheet tools like Excel are significantly more accessible for casual exploration and filtration of genomic datasets than doing the same through GATK. The tabular format of spreadsheets makes information more easily comparable across variants and filtration is made significantly more intuitive by the use of graphical widgets that use natural language instead of JEXL expressions requiring case-sensitive shorthand notation. However, Microsoft Excel is much less powerful, expressive, and domain-specific than command-line alternatives. For example, it is difficult or otherwise kludgy to filter variants in Excel using
the “or” operator between facets (e.g. SIFT score greater than 0.6 or Polyphen-2 score greater than 0.6). Finally, Excel is resource intensive as it keeps the entire dataset being analyzed in memory. In order to maintain performance there is an application-wide threshold on the number of records that can be stored in Excel; the latest version can accommodate up to 1,048,576 rows which is significantly fewer than is required for analysis of even medium-sized genomic datasets.

**VARSIFTER**

VarSifter is an open-source desktop graphical software tool developed at the National Institutes of Health in the United States which allows investigators of varying computational expertise to easily and quickly sort, filter and sift through sequence variation data in a spreadsheet format [83] as shown in Figure 3.7. It provides some domain-specific features, for example the ability to filter variants using a gene list from a file, but is otherwise similar to Microsoft Excel.

By virtue of its design as a desktop tool that loads the complete volume of genotyped variants into memory, VarSifter has significant performance limitations. Like Excel it consumes a significant amount of memory for reasonably small datasets. It is reported to use a total of 5.3GB RAM to load all detected variants from 160 exomes, which is comparable to the amount of memory used by Excel for the same dataset. Although the total number of variant calls loaded for this study is not disclosed in the Varsifter paper, it can be inferred to be less than 1,048,576 as this is the maximal number of rows that can be loaded into Excel for a fair comparison to be possible.
The Variation Browser, or VarB, is a graphical tool developed for visualizing multi-sample variant data and for performing simple faceted searches on variant attributes [84]. Like in Savant and IGV (and unlike Excel and VarSifter) variants are visualized in VarB in a matrix where rows correspond to samples and columns to genomic positions. Similar to the alternative variant view in Savant the horizontal axis is non-linear, showing only positions where genetic variations exist. Strangely, VarB uses different colours to represent a variant’s zygosity — the degree of similarity of the sample’s genotype compared to the reference. Fully saturated primary colours are used to encode the four possible zygosity states, which produce moiré effects that are arguably visually unpleasant. A screenshot of VarB is shown in Figure 3.8. VarB provides a simple interface containing sliders and checkboxes for discarding irrelevant variants but the number of tuneable attributes is too few to be used for disease-causing mutation discovery. VarB is also a desktop tool, though it uses a windowing technique as in genome browsers to conserve memory.
Golden Helix SNP & Variation Suite (SVS) is a paid collection of tools for monitoring, manipulation, analysis and visualization of genomic variants. It is licensed commercially on an annual subscription basis.

The suite provides a wide range of features covering visualization (e.g. genome browser, histograms, Linkage Disequilibrium, heatmaps, box plots, scatter plots, histograms, pie charts, Venn diagrams) as well as complex statistics and methods of sequence analysis [19]. SVS supports visualization and filtration of genetic variants imported from VCF. It extracts annotations that are nested within the VCF INFO field and these fields are made searchable through a graphical interface shown in Figure 3.9. Initially all variants from an imported VCF file are shown in a tabular format where samples are displayed across rows with information about genotyped positions shown in columns. In SVS, a user performs faceted search on variants by first choosing an annotation and then specifying a rule against its values. Importantly, these rules can’t be applied in combination so it is impossible to straightforwardly retrieve the union of results of rules for different facets.

Golden Helix’s SVS is a desktop tool that stores and analyzes data locally. The software therefore has a large memory footprint and is limited in its ability to scale, support remote data access, and support collaboration across multiple users.

INGENUITY VARIANT ANALYSIS
Ingenuity Variant Analysis is a paid web-based solution for interpretation of genetic variants with emphasis on disease-gene discovery for genetic researchers without bioinformatics training.
The platform is commercial and tiers varying in the number of features they support are available for purchase on an annual subscription basis.

The platform supports uploading of VCF files, and subsequently annotates them with relevant datasets, collectively referred to as the “Ingenuity Knowledge Base”. It displays genetic variants and their annotations in a tabular format. A select set of filters are automatically applied to datasets based on the following facets, in order: quality (remove low quality), allele frequency (remove common), and predicted deleterious. Subsequent filtration can be performed interactively using the interface shown in Figure 3.10. Like each of the aforementioned graphical platforms though, rules are specified in serial and thus full boolean logic with respect to filtration rules is not supported. The system features aggregation of variants (e.g. to conduct statistical enrichment testing) based on genes, pathways, and diseases by navigating the tabs that are located above the spreadsheet.

GEMINI

GEMINI (for GEnome MINIng) is a command-line tool that supports annotation, storage, and complex querying of genetic variants [85]. GEMINI annotates genetic variants, originally in VCF format, with a diverse and customizable set of additional metadata. GEMINI uses a relational
database, specifically SQLite, to store variants and their annotations. Unlike GATK which analyzes VCF files directly, the use of a database with typed fields significantly improves speed of queries, as will be shown later.

GEMINI accepts a broad set of queries to screen variants against faceted search criteria and complex combinations of these. GEMINI also provides a simple language for screening variants by inheritance patterns if familial relationships are specified.

### 3.2 MedSavant

Next-generation DNA sequence analysis holds the promise of improving diagnostics and treatment for individuals affected with genetic diseases. A significant challenge to fulfilling its promise is in implementing data analysis strategies that can be used to easily and efficiently identify causative genetic mutations from the large number of variants discovered through genotyping HTS datasets. A common approach is to annotate variants with informative metadata (e.g. genomic context, predicted harmfulness), filter for potentially causal genetic mutations based on complex criteria, and iteratively refine the previous steps after manual
inspection of the results. This method utilizes a combination of ad-hoc and loosely integrated computational scripts that rely on flat files as the data-transfer medium.

This existing paradigm for genomic variant analysis, which involves serial processing of flat files with manual inspection as an endpoint, inherently requires a substantial amount of informatics expertise to run, is non-interactive and time-consuming, and thus does not scale to the extent needed for clinical and other large-scale sequencing applications. It is well appreciated in bioinformatics – and in other scientific domains – that visually-guided real-time exploration significantly aids the understanding of big datasets, as is evidenced by the success of tools like ABYSS-Explorer [86], Savant Genome Browser [58], and Galaxy [77]. These tools deliver computationally-intensive analytics through accessible user interfaces.

As previously described, a few freely available graphical applications for variant searching have been developed; however, their desktop architecture places significant limitations on performance and restricts access to the data to a single computer. For example, VarSifter loads the complete volume of genotyped variants into memory, and for this reason is practical only for exome analysis. There is an unmet need for a freely available and accessible software to perform dynamic visual analyses of large volumes of genetic variant data detected through sequencing, with an emphasis on facilitating discovery of disease etiology.

MedSavant is an integrated solution for the storage, annotation, filtration, prioritization, and visual inspection of variants that is entirely graphical, interactive, and scalable to manage datasets generated by large sequencing projects. MedSavant is a client-server application that uses a custom high-performance database engine to store and perform faceted search queries on data. This design results in significantly improved performance and user-friendliness over standard flat file or client-side alternatives. MedSavant delivers interactivity even at scale, allowing very fast ad-hoc querying of millions of genomic variants, while being easy to use, providing multiple visualization mechanisms that allow the inspection of large datasets at various levels of granularity. It follows the success of platforms like iOS, Savant [58], and Cytoscape [87] in supporting third-party app development to enable enticing extensions to be developed by the community.
3.2.1 Features and Design Principles

MedSavant is a contribution to the research and recently the clinically-oriented genomics community that has been developed since 2011. Its outcomes are both conceptual and practical: it serves as a proof-of-concept for a complete system for variant interpretation (including variant annotation, storage, querying, visualization, and reporting), and proposes techniques for improving upon performance and usability compared to status quo. As a freely available software tool, it can be downloaded and used in various and customized settings.

Key features that the system embodies include:

• client-server architecture to support secure, remote, collaborative access to large datasets
• parallelized annotation of genetic variants using relevant datasets and algorithms
• unique database representation for annotated genetic variants that makes faceted search fast
• integration of patient, phenotype, and ontology data
• simple user interface for specifying complex queries using natural language
• various apps to perform visualization (e.g. genome browser) and analysis (e.g. Mendelian inheritance)
• freely available

3.2.2 Versions

The first version of MedSavant was released September 2011, with a number of improvements released to date. A paper describing the work is in submission. A summary of release and presentation timeline for this project is shown in Table 3.2. It is the first, to our knowledge, freely available system to enable interactive visual exploration (including faceted search and visualization) of large genetic variant datasets.

The platform has evolved substantially since the first version which supported efficient storage and basic search of genomic variants. The original user interface for variant search was akin to those of most e-commerce sites, where searchable facets are listed on the left and allowable options for each are configured using nearby widgets. A table of resulting variants were shown in a spreadsheet to the right.

Important functionalities were gradually introduced based on testing and consultation with prospective users. For example the Savant Genome Browser was integrated with variant
search results to improve the workflow of inspecting supporting read alignment datasets; the user interface for specifying queries was significantly redesigned to allow for more complex queries to be specified, as shown in Figure 3.11; an app-centric dashboard was implemented to provide a familiar interface for launching applications and to improve navigation; app-to-app communication was added to the API to tighten integration of app functionalities.

Simultaneously, a number of apps were developed to demonstrate the potential of the system to be used both in research and clinical research settings. A number of installations of MedSavant have been deployed for evaluation by users across the world.

<table>
<thead>
<tr>
<th>Date</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011 September</td>
<td>Minor Release</td>
<td>MedSavant 1.0.0 alpha released</td>
</tr>
<tr>
<td>2011 July</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at HitSeq</td>
</tr>
<tr>
<td>2011 October</td>
<td>Poster Presentation</td>
<td>MedSavant Presented at Personalized Medicine in the Genomics Era</td>
</tr>
<tr>
<td>2011 November</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at Genome Informatics</td>
</tr>
<tr>
<td>2012 March</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at VIZBI</td>
</tr>
<tr>
<td></td>
<td>Poster Presentation</td>
<td>MedSavant Presented at VIZBI</td>
</tr>
<tr>
<td>2012 September</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at TorBUG</td>
</tr>
<tr>
<td>2012 October</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at BioVIS</td>
</tr>
<tr>
<td></td>
<td>Poster Presentation</td>
<td>MedSavant Presented at BioVIS (best poster)</td>
</tr>
<tr>
<td>2013 October</td>
<td>Oral Presentation</td>
<td>MedSavant Presented at Genome Informatics</td>
</tr>
<tr>
<td>2013 December</td>
<td>Major Release</td>
<td>MedSavant 1.1.0 released</td>
</tr>
<tr>
<td>2014 February</td>
<td>Minor Release</td>
<td>MedSavant 1.2.0 released</td>
</tr>
<tr>
<td>2014 June</td>
<td>Poster Presentation</td>
<td>MedSavant Presented at ICCG</td>
</tr>
<tr>
<td>2014 July</td>
<td>Oral Presentation</td>
<td>MedSavant presented at Big Data for Health</td>
</tr>
<tr>
<td>2014 November</td>
<td>Minor Release</td>
<td>MedSavant 1.3.6 released</td>
</tr>
<tr>
<td>2015 February</td>
<td>Manuscript</td>
<td>MedSavant manuscript in submission</td>
</tr>
</tbody>
</table>

Table 3.2 Significant releases, manuscripts, and public presentations of MedSavant
The latest version of MedSavant is made available for download, with installers for Windows, Linux, and Mac, at genomesavant.com/p/medsavant/download and the source code is available at github.com/compbio-UofT/medsavant.

3.2.3 Programming Language

Most of the MedSavant client and server is coded in the Java Programming Language. The Java platform was chosen for this project because a large number of related bioinformatics libraries programmed in Java already existed, and since it facilitates development of client-server applications. In particular, Java Remote Method Invocation (RMI) allows for secure, remote invocation of methods on different virtual machines and automatic cross-serialization between them. RMI facilitates creation of public APIs and shared data models; these are otherwise tedious to maintain — e.g. compared to standard web applications (at the time the project started in 2011, there weren’t as well-established frontend frameworks) — especially for data-driven applications that involve visual inspection and object manipulation.

Data is stored in a database for persistence, integrity, and query performance. The server interacts with the database via the the Structured Query Language (SQL). SQL is a standard language for relational databases, is well-known by developers, and otherwise is easy to learn and write. The next section describes the database design and technologies used for this application.
3.2.4 Architecture

While standardized text-based flat file formats have commonly been used as a data-transfer medium for genomic datasets (e.g. BED, GFF, and the VCF) and enabled interoperability between computational tools, they do not facilitate faceted search — i.e. searches based on arbitrary metadata affiliated with the records. This makes the interactive refinement of causal genetic mutations difficult, as such queries are typically based on a complex set of quality, contextual, and other criteria that necessitate reprocessing these flat files whenever the search parameters change. As a result, sets of candidate causal mutations often contain large numbers of poor quality or otherwise irrelevant variants that are frequently manually filtered, rather than resorting to further parameter refinement and reprocessing. This problem is exacerbated as both the size and diversity of genomic datasets continue to increase.

The MedSavant system utilizes a custom database and format to store genetic variants and related datasets to simultaneously improve data integrity and increase query performance substantially, as will be explained later, compared to file-based approaches. Indeed, there is a non-trivial amount of computational time that is required to convert variant files into a database-compatible format. However given the iterative nature of variant exploration, there are substantial long-term savings of both computational and person time during analyses of these data as suggested by Figure 3.12. A comparative performance analysis is provided later.

Systems that operate on files also suffer from redundancy and versioning issues. It is impossible to create persistent subsets of files, e.g. filtered variants, without creating additional files containing duplicated data. Moreover it is not uncommon to have several VCF files produced in a variant analysis pipeline that must be explicitly versioned, usually done in an ad-hoc manner by appending to the filename or with a text-based string in the file’s header. As will be seen, MedSavant makes use of the ability for subviews of databases to be generated and for these to be communicated between separate analytic steps. This architecture simultaneously improves data integrity and increases query performance, avoiding wasted disk space and time spent managing file-based approaches.

Desktop systems are advantageous when the amount of data to be analyzed at one time does not exceed available computational resources. Genomic variant analyses often involve filtration through very large volumes of genotypes that quickly consume computational
resources as is the case for Varsifter and Microsoft Excel. MedSavant is a client-server application as depicted in Figure 3.13, where workloads of data processing, storage, and querying are performed by a server component and specific user requests and information visualization are done by connecting clients. The server component makes use of a high-performance database that runs on the same physical computer as it. The client-server architecture lends to better shared data access and theoretical scalability that far outperforms what is possible using desktop-based architectures.
3.2.5 Server

The MedSavant server is a Java application that manages sessions from connecting clients, including authentication, authorization, variant processing, and querying. It also serves as a
mediator between client requests and the database. Important server components are now described.

**AUTHENTICATION AND AUTHORIZATION**

A single MedSavant server can host multiple databases, each of which can have multiple projects. For example, it is possible to host two research groups' databases of projects on the same server instance. Besides requiring usernames to be unique across an installation, these databases are otherwise agnostic of each other's existence, which makes the system amenable to provide for multiple working groups.

The authentication and authorization component controls who has access to which database and what privileges each user is entitled to have. There are 3 levels of user, which are chosen upon user creation, with the following privileges:

- **guest**: read-only access
- **user**: edit cohorts, edit region sets (plus guest privileges)
- **administrator**: upload variants, edit patient information, manage users, configure projects (plus user privileges)

This component manages sessions for connecting clients which are persistent as long as the client is connected or the analytic tasks they issue through the interface (which may require long-running asynchronous processing) are running.

**VARIANT PROCESSOR**

It is useful to both search and analyze genetic variants based on their intersection with other relevant datasets, for example, which genes they effect or whether or not they have been previously identified and implicated in disease. The process of attaching such metadata to variant records is called variant annotation and can be accomplished by several command-line bioinformatics tools such as ANNOVAR [82], Jannovar [88], and SNPEff [89].

Variant annotations can apply to single positions (e.g. positional conservation) or chromosomal ranges (e.g. genic vs. non-genic). Annotations at single positions can also be allele-specific (e.g. SNV allele frequency). Where it is possible to precompute values for all positions (and in the latter case, all possible alleles at all positions), lookup tables have been generated to speed the process of annotation. For example, a long list of precomputed annotation databases that include allele frequencies from a number of population sequencing studies, harmfulness
prediction scores, and genes are available for annotation with ANNOVAR. These are also supported by MedSavant, and a subset is listed in Table 3.3.

Sometimes it is not possible or practical to precompute annotations, as is the case for indel annotation, since the number of potential variants is infinite. In these cases, it is preferred to compute annotations at runtime for each variant observed in the dataset. Jannovar is a Java library for applying mutational effect annotations, among others, that computes annotations at runtime for variants affecting coding sequences, splice junctions, UTR sequences, and non-coding RNA transcripts. Mutational effect annotations indicate whether a variant causes stoploss, stopgain, or frameshift events, for example, and are informative for implicating a variant in altering a protein’s function. Jannovar uses an interval tree (a data structure also used in representing and searching reads in BAM files) to identify gene transcripts affected by a given variant and then predicts an effect on a case-by-case basis.

The MedSavant server contains within it a variant processor that performs comprehensive variant annotation on uploaded VCF files before loading their records into the high performance datastore. MedSavant performs transcript-based annotation using Jannovar and uses an internally-developed, parallelized annotation tool that performs similar functionalities as ANNOVAR. In fact, all annotations that can be applied using ANNOVAR can be applied by the variant processor within MedSavant. A default set of annotations are applied to each MedSavant project, however, these are customizable by an administrator. It is also possible for an administrator to add custom annotation tables.

3.2.6 High Performance Database
MedSavant relies on a carefully constructed database that is able to execute and deliver query results fast enough to deliver interactive exploration of even very large genomic datasets. The use of a structured database is an important departure from previously popular pipelines that prescribe filtration on self-described VCF files using command-line tools. A number of database technologies were considered for use with MedSavant, including relational (MySQL, PostgreSQL), columnar (MonetDB, Infobright), and NoSQL databases, which include various models for storing and accessing data that is “Not Only SQL”.

As their name suggests, columnar databases store data column-wise. The differences between how data is stored in record-oriented versus column-oriented databases are illustrated
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avsift</td>
<td>whole-exome SIFT scores for non-synonymous variants (obsolete and should not be used any more)</td>
</tr>
<tr>
<td>ljb26_sift</td>
<td>whole-exome SIFT scores with missing values imputed</td>
</tr>
<tr>
<td>ljb_pp2</td>
<td>whole-exome PolyPhen scores</td>
</tr>
<tr>
<td>ljb26_cadd</td>
<td>whole-exome CADD scores</td>
</tr>
<tr>
<td>ljb_phylop</td>
<td>whole-exome PhyloP scores</td>
</tr>
<tr>
<td>ljb_lrt</td>
<td>whole-exome LRT scores</td>
</tr>
<tr>
<td>ljb_mt</td>
<td>whole-exome MutationTaster scores</td>
</tr>
<tr>
<td>ljb2_fathmm</td>
<td>whole-exome FATHMM scores</td>
</tr>
<tr>
<td>ljb2_siphy</td>
<td>whole-exome SiPhy scores</td>
</tr>
<tr>
<td>ljb_gerp++</td>
<td>whole-exome GERP++ scores</td>
</tr>
<tr>
<td>ljb23_metasvm</td>
<td>whole-exome MetaSVM scores</td>
</tr>
<tr>
<td>ljb23_metalr</td>
<td>whole-exome MetaLR scores</td>
</tr>
<tr>
<td>ljb26_vest</td>
<td>whole-exome VEST scores</td>
</tr>
<tr>
<td>ljb26_cadd</td>
<td>whole-exome CADD scores</td>
</tr>
<tr>
<td>ljb_all</td>
<td>whole-exome LJBSIFT scores</td>
</tr>
<tr>
<td>ljb2_all</td>
<td>whole-exome SIFT scores</td>
</tr>
<tr>
<td>ljb23_all</td>
<td>whole-exome SIFT scores</td>
</tr>
<tr>
<td>ljb26_all</td>
<td>whole-exome SIFT scores</td>
</tr>
<tr>
<td>cg69</td>
<td>allele frequency in 69 human subjects sequenced by Complete Genomics</td>
</tr>
<tr>
<td>esp5400_all</td>
<td>alternative allele frequency in all subjects in the NHLBI-ESP project with 5400 exomes</td>
</tr>
<tr>
<td>esp6500_all</td>
<td>alternative allele frequency in all subjects in the NHLBI-ESP project with 6500 exomes</td>
</tr>
<tr>
<td>1000g2012apr</td>
<td>alternative allele frequency data in 1000 Genomes Project</td>
</tr>
<tr>
<td>snp138</td>
<td>dbSNP with ANNOVAR index files</td>
</tr>
<tr>
<td>popfreq_max</td>
<td>A database containing the maximum allele frequency from various population tables</td>
</tr>
<tr>
<td>refGene</td>
<td>FASTA sequences for all annotated transcripts in RefSeq Gene</td>
</tr>
<tr>
<td>knownGene</td>
<td>FASTA sequences for all annotated transcripts in UCSC Known Gene</td>
</tr>
<tr>
<td>ensGene</td>
<td>FASTA sequences for all annotated transcripts in ENSEMBL Gene</td>
</tr>
<tr>
<td>gerp++elem</td>
<td>conserved genomic regions by GERP++</td>
</tr>
<tr>
<td>gerp++gt2</td>
<td>whole-genome GERP++ scores greater than 2 (RS score threshold of 2 provides high sensitivity while still strongly enriching for truly constrained sites.)</td>
</tr>
</tbody>
</table>

**Table 3.3** A subset of precompiled annotation tables available in MedSavant. A default set of annotations are applied to each MedSavant project. The set of annotations that are applied to variants within a project is customizable by an administrator, however. It is also possible to add custom annotation tables.
in Figure 3.14. The column-oriented format lends to improved data compression because data within fields are often more alike than data between columns. Moreover, when query conditions are specified against a limited number of fields, column stores yield significant compute savings since only the relevant columns must be read from disk rather than retrieving entire rows of data as traditional relational databases do. Columnar databases generally scale better than record-oriented databases but typically sacrifice mutability of data for high-compression and query performance. These limitations are often not practical for use cases where records are updated frequently, and so columnar databases are best suited for historical or otherwise permanent records.

NoSQL refers to a group of semi-structured storage technologies designed to scale better than relational databases for specific use cases but which in exchange are awkward or impossible to query as flexibly. Key-value databases are a type of NoSQL that use the associative array (i.e. map or dictionary) as the primary data model. Data is represented as a collection of key-value pairs, where a given key maps to some unique value in the collection. These databases are useful for applications where fast lookups based on one or few identifiers is needed, for example a

![Comparison of record-oriented and column-oriented table storage. In record-oriented storage, entire records are appended to each other. In column-oriented storage, values from the same column are appended to each other. Column-oriented databases generally yield higher compression and better performance where conditions are on few fields compared to record-oriented databases, however, make trade-offs like requiring tables to be read-only.](image)

103
phonebook application. Document-oriented databases represent another class of NoSQL that manage document-oriented information, which may differ widely in content and format. Document databases are useful for applications that have loosely coupled, unstructured datasets like blog posts or encyclopedias. Graph databases form a third class of NoSQL that are designed for data whose relations are well represented as a graph (elements interconnected with an undetermined number of relations between them). Graph databases excel for social networks and map applications, for example. None of key-value, document, or graph databases map well to the task of broad faceted search, requiring record lookup based on arbitrary fields, none of these was considered for the primary datastore for MedSavant.

DATABASE EVALUATION
The remaining database platforms (mainly, record-oriented and column-oriented databases) were evaluated based on disk usage and query speed. Other limitations, including cost, mutability of tables, and query complexity were considered. The record-oriented databases tested included MySQL and PostgreSQL while column-oriented platforms included Infobright (both Enterprise and Community editions) and MonetDB.

To evaluate the resource use of candidate databases, a large number of variants were loaded into each. A total of 71 VCF files from the Autism Sequencing Project conducted at The Centre for Applied Genomics — collectively containing 5,855,350 variant records and occupying 810 MB of disk space — were converted to CSV files and loaded via the LOAD DATA LOCAL INFILE command in SQL. The amount of disk space and time of loading are compared in Figure 3.15. Both Enterprise and Community editions of Infobright exhibited superior compression, while all databases except PostgreSQL had low load times.

A set of typical queries on variant datasets were issued to each loaded database and query execution was timed:

- query 1: count the number of variants in the database
- query 2: retrieve the first 1000 variants reported on chromosome 1
- query 3: retrieve the first 500 variants between positions 51,220,174 and 51,223,040 on chromosome 19 (SHANK1 gene)

To avoid the possibility of performance gains from caching or adaptive query processing, queries were executed 10 times on freshly booted databases and averaged. The results of these
tests are visualized in Figure 3.15. Column-oriented databases executed quickly for all queries, and significantly faster than record-oriented databases for the gene query. Both editions of Infobright were slightly slower than MonetDB on the first two queries but slightly faster for the third and more complicated query.

Infobright Community Edition (ICE) exhibited over 4.5X compression and over 4X increase in query speed compared to regular MySQL in preliminary testing. As will be shown later, its performance can be improved through further optimizations, and can maintain its performance to deliver interactive faceted search (i.e. queries execute in few seconds or less) for up to 100,000,000 variant records (or, on the order of 1,000 exomes). Based on its combination of low cost, ease of use, high compression, and high performance, Infobright Community Edition (ICE) was chosen as the default database platform for MedSavant.

Figure 3.15 Comparison of contending database platforms. Disk use (top-left) of databases for 5,855,350 variant records from 71 VCF files, including indices, and their load times (top-right). Query performance (bottom) for various typical queries on this dataset. Based on its combination of low cost, ease of use, high compression, and high performance, Infobright Community Edition (ICE) was chosen as the default database platform for MedSavant.
of low cost, ease of use, high compression, and high performance, ICE was chosen as the default database platform for MedSavant.

**INFOBRIGHT**

Infobright Community Edition (ICE) is an open-source solution designed for analytic ad-hoc queries and large amounts of data [90]. It is primarily used for machine-generated data, i.e. data produced by automated systems often as a result of monitoring. This kind of data is typically rarely updated once it is stored and it is maintained for long periods of time for regulatory reasons. The database is easy to set up and does not require specialized equipment, database administration expertise, or complex manual tuning. All these aspects make it suitable for circumstances under which MedSavant is typically used.

ICE is queryable by SQL. Infobright provides the computing engine along with the column-oriented storage engine and a custom loader, while MySQL provides the main server implementation, management services, connectors and other tools.

The advertised advantages to Infobright Community Edition (ICE) include:

- no licensing fees
- market-leading data compression (from 10:1 to 40:1), drastically reducing I/O
- ideal for data volumes up to 50TB
- fast response times for ad-hoc analytic queries
- query and load performance remains constant as the size of the database grows
- no requirement for materialized views, complex data partitioning strategies, or indexing
- simple to implement and manage, requiring little administration
- runs on low cost, off-the-shelf hardware
- reduction in data warehouse capital and operational expenses by reducing the number of servers, the amount of storage needed and their associated maintenance costs, and a significant reduction in administrative costs

Internally, ICE is based on columnar data storage [91], which allows it to identify and read only the relevant columns off the disk rather than retrieve entire rows of data as traditional relational databases do. The engine features extreme data compression, achieving a reported 10:1 compression ratio on average [92]. Compared to the size of standard variant files
traditionally used in variant search process, this results in 10–20 times less disk space empirically.

Performance and scalability of ICE on commodity hardware are also achieved using adaptive query processing as well as the theory of rough sets. The crucial aspect of this approach is that the data is divided into rough rows, each representing a block of 64K original rows. Rough rows are tagged with information about their values on data columns; these are called data packs and each of them is compressed individually.

An important concept leveraged by Infobright is the knowledge grid, an in-memory structure that automatically creates and stores information about the data upon load and when queries are executed. The knowledge grid consists of data pack nodes as well as knowledge nodes which provides an effective alternative to indices, which are not supported by ICE. Data pack nodes point to individual data packs and contain a set of statistics and aggregate values for the data from the data pack – MIN, MAX, SUM, AVG, COUNT and the number of NULLs. Knowledge nodes further extend this metadata, describing numeric value occurrences, character positions, as well as column relationships between data packs. The information from the knowledge grid is leveraged by the computing engine, which reduces the amount of data that has to be accessed in order to answer a query – the actual data is accessed only when rough information is not enough to proceed.

ICE comes with many exciting features contributing to its superior performance on analytic queries over many commercial solutions. However, there are significant trade-offs. ICE only runs on a single machine and does not support data partitioning, replication or high availability in general. It also behaves as a read-only database, i.e. there is no support for INSERT, UPDATE and DELETE statements (the data can only be loaded from a file). There is also no support for ALTER TABLE statements and temporary tables visible in the scope of a single connection.

Another major limitation is the single-threaded nature of query processing – query execution is not parallelized, which prevents ICE from fully utilizing multiple processors on a machine. This limitation is valid with respect to parallelism at the level of individual queries; multiple connections are handled using multiple threads. Furthermore, ICE is not capable of executing queries while the data is being loaded.
The enterprise edition of the Infobright database (IEE) is devoid of many of the aforementioned problems of ICE. Its workload management system, for example, handles concurrent queries, thus improving performance. It also fully supports data manipulation language (DML), parallel data loading, replication and high availability. IEE is capable of executing queries while loading data and supports temporary tables. Unlike ICE, however, it is a commercial solution that requires a yearly per-instance license that is prohibitively expensive for broad distribution of MedSavant.

DATA DENORMALIZATION

The VCF file format supports multiple samples, where a single line contains information about the variant position itself, as well as an arbitrary number of sample-specific columns that follow:

```plaintext
<variant information> <sample 1 information> ... <sample N information>
```

It is both difficult and inefficient for data to be stored in ICE in this way: table alterations are not supported so the addition of new samples would require dumping and reloading of the variant table, and the benefits of storing data columnwise are lost because sample information — which contain identical fields — would be stored in different columns. Hence, in MedSavant, the technique of data denormalization is used to produce a flattened table containing pairs of variant and sample information. Each line of a VCF file is denormalized as follows:

```plaintext
<variant information> <sample 1 information>

... 

<variant information> <sample N information>
```

Data denormalization typically avoids the need to perform JOIN operations at runtime and is an efficient technique to apply if the combined view is needed frequently as is the case in MedSavant. The technique has apparent drawbacks though, including the introduction of redundantly stored data, the increase in database size, how this could affect query performance, and the difficulty in updating the database.

These concerns are negligible given the application of Infobright, though. In the proposed schema, variant information is indeed repeated, causing redundancy, but since columns are compressed this does not yield a significant increase in database size (for example, these
columns can be very efficiently run-length encoded). In fact, denormalization of variant information tends to produce better compression and query performance in ICE than a direct translation of the VCF format.

TOOL EVALUATION
To evaluate the performance of the MedSavant database and query engine, two datasets taken from individuals sequenced as part of the 1000 Genomes Project were loaded into Infobright and denormalized using the aforementioned protocol, and the database sizes and query performances were measured in comparison to two other previously described tools for variant search: GATK and GEMINI.

The first dataset, hereafter referred to as the Yoruban Trio dataset, included genetic variants compiled from whole-genome sequencing and genotyping from a Yoruban mother, father, and daughter. Yoruban people are an ethnic group from West Africa, and the genomes sequenced from this group are some of the most well-studied from the 1000 Genomes Project. A single multi-sample VCF file was downloaded from the 2010 pilot data release and loaded into MedSavant. A total of 9,779,945 non-reference calls from 4,502,439 variant records were loaded, where here a call represents a unique variant and sample pair.

The second dataset, hereafter referred to as the 1000 Genomes dataset, included all genotypes on chromosome 1 from half of the project’s individuals. The first 500 individuals from the multi-sample VCF file from the 2010 Phase 1 release were parsed and loaded into MedSavant. A total of 134,958,340 non-reference calls from 2,896,960 variant records were loaded.

These datasets were prepared for use with GATK and GEMINI using best practices for each. GATK operates on zipped or flat VCF files, which do not require any preprocessing, whereas GEMINI requires a step which instantiates and loads an SQLite database accordingly. All data processing and subsequent tests were performed on a machine with CentOS 6.4 having 8 Dual-Core AMD Opteron (8220 SE) Processors and 32 GB RAM.

Data Compression
To assess the efficiency with which each tool uses disk resources, the database sizes for each tool were measured for these two datasets. As previously described, the genotypes for the Yoruban Trio and 1000 Genomes datasets were formatted for use with GEMINI and MedSavant and the
resulting database file sizes compared. GATK does not utilize a database, but instead operates on flat or zipped VCF files as input. The resulting file and database sizes for each dataset and tool are illustrated in Figure 3.16.

The results reveal the relative strengths and weaknesses of the various storage techniques tested. The size of the flat VCF file for the Yoruban Trio dataset was similar to MedSavant’s compressed database, but was substantially larger (almost 10X) for the 1000 Genomes dataset. The bloating of multi-sample VCF files is owing in large part in storing sample information redundantly. For example, the genotype field for a common allele that is possessed by all individuals in the 1000 Genomes Project would be repeated 1000 times. Standard gzip compression, which works to condense frequently repeated strings, alleviates this issue and yields files of comparable size to the tested databases. MedSavant’s database sizes were competitive for the Yoruban Trio dataset and the smallest for the 1000 Genomes dataset. The GEMINI database required more resources than other formats for the Yoruban Trio dataset, but exhibited competitive compression for the 1000 Genomes dataset. It is important to note that like MedSavant, GEMINI applies and stores a large number of annotations to variant positions and these additional, helpful metadata are not provided in the input VCF files (and for some annotations, MedSavant). Moreover, since it compresses sample information, GEMINI is much less sensitive compared to flat VCF files and MedSavant to increases in the number of samples.
Search Performance

To assess the speed at which queries can be performed a few faceted search queries, for which MedSavant is designed, were issued against the Yoruban Trios dataset using each tool and execution times measured. As was done previously, to avoid the possibility of performance gains from caching or adaptive query processing, queries were executed 10 times on freshly booted databases and averaged when applicable. The queries were as follows, and the resulting query speeds for each tool are illustrated in Figure 3.17:

- query 1: count the number of variants in a specific individual (with ID “NA1939”)
- query 2: count the number of variants in this individual with high coverage (DP > 20)
- query 3: count the number of heterozygous variants in this individual with high coverage (DP > 20)

The results demonstrate significant performance gains through MedSavant’s use of a columnar compressed database. Both GATK and GEMINI took on the order of minutes and seconds, respectively, to perform these queries, while MedSavant completed in fractions of seconds, fast enough to enable interactive exploration via faceted search of variant datasets.

3.2.7 Client

MedSavant is intended to be a simple-to-use system that makes interactive analysis and visualization of genetic and related datasets possible to researchers without informatics training. A graphical desktop client was implemented to make MedSavant project management and data exploration extremely accessible. Client installers for all of the major operating systems are provided at www.genomesavant.com/p/medsavant/download.

Figure 3.17 Comparison of query speed by tools for variant search for common queries on the Yoruban dataset.
The client can connect to any number of MedSavant servers, each hosting specific projects' data securely over a network, either via an internal network or over the internet. Upon starting the client, users are prompted to specify the server database they wish to connect to in addition to providing their login credentials as shown in Figure 3.18.

A single server may host a number of databases, each containing a set of projects. For example, the MedSavant server running at the Hospital for Sick Children can be configured to host a database for each research group within the institution who requests access. These databases are completely agnostic of each other so it is impossible for one group to gain access to another group's data.

Each database contains a number of projects, each with its own set of patients and genetic datasets. For example, the Care for Rare database hosted by the server at the Hospital for Sick Children can be configured to have one project for each disease cohort that is sequenced. While data cannot currently be queried across projects within a database, they are not agnostic to each other. That is, a database user can list all projects within it and access
their data, subject to additional user-level restrictions listed previously. Users with administrative privileges can create projects and invite users. Upon initializing a project they are prompted to specify:

- a project name
- the reference genome (e.g. GRCh37, GRCh38)
- patient characteristics to be stored (e.g. age, weight, IQ, etc.)
- which VCF fields to index
- annotations to apply to variants (options listed previously)
- whether or not to include reference calls (i.e. where the sequenced sample matches the reference)

The project creation wizard is shown in Figure 3.19. All settings except the project name can be modified after creation.
Figure 3.20 Evolution of the MedSavant dashboard. The first version of MedSavant (top) supported basic data management and faceted search of genetic variants. The latest version of MedSavant (bottom) is multi-functional, supporting various apps in addition to the original faceted variant search application (now called the Variant Navigator). The app-based dashboard was implemented in response to feedback from early adopters who requested a simpler and more familiar interface for navigating the various functionalities of the client.
DASHBOARD

MedSavant was originally developed with a singular function: to enable faceted search of genomic variants. However, in numerous releases since the first, it has grown into a multi-functional application with many different use cases. The interface, and most dramatically the dashboard, has evolved to support launching of different applications. An app-based dashboard, which mimics the interfaces found on most mobile devices, was implemented in response to feedback from early adopters who requested a simpler and more familiar interface for navigating the various functionalities of the client. The dashboards for the first and most recent versions of MedSavant are contrasted in Figure 3.20.

3.2.8 Apps

MedSavant follows the success of other app-centric platforms like iOS and Android, for mobile devices, and genetic visualization tools like Savant Genome Browser and Cytoscape by having its functionalities extendable through an Application Programming Interface (API). Third-party application developers can leverage the power of the underlying analytics engine and the convenience of the MedSavant client environment to quickly develop and deploy rich visual applications for the exploration of data contained within the database, or to intersect these data with external datasets.

APP DEVELOPMENT

App development is straightforward and requires implementation of one Java interface. A Hello World starter-project is provided for download from the Developer section of the MedSavant website, and the main class implementation is shown in Figure 3.21. MedSavant apps are required only to specify a name, icon, and to populate a graphical component to be shown when the app launches. The app may optionally respond to load and unload events, which allow processes to be paused or terminated when a user leaves the application. Apps can utilize helper functions provided in the API for creating, searching, and manipulating annotations, cohorts, patients, projects, region sets, and variants. Developed apps can be submitted to the public App Library are thereby downloadable by any MedSavant user.

MedSavant comes with the following apps pre-installed:

- My Account
- Patient Directory
These default apps support core functionalities of MedSavant and cannot be uninstalled. A number of these are now described.
APP LIBRARY

The App Library, shown in Figure 3.22, lists apps that have been made widely available to the community. It supports the management — that is, the installation, update, and uninstallation — of such apps. Apps that are intended for internal use, for example, those which are still in development or which access private data, can be distributed and installed by file through the App Library as well.

PATIENT DIRECTORY

The Patient Directory app lists patients. While all patients have basic information (e.g. a unique identifier, a family identifier, phenotypes, a sex, and a status which indicates if they are affected by a disease), patient profiles can be extended through the addition of more tracked fields. These may be useful to store clinical information that is relevant to the diagnosis or stratification of individuals within a disease group, for example. Familial relationships can also be recorded and pedigree diagrams, showing the affected statuses of family members, can be generated as seen in Figure 3.23.

Collections of patients, called cohorts, can be created in the Patient Directory. Cohorts can be created based on arbitrary criteria; they are useful for stratifying patient populations by

Figure 3.22 The MedSavant App Library app supports the management of apps. Users can browse, install, update, and uninstall apps through this component. Apps that are intended for internal use, for example, those which are still in development or which access private data, can be distributed and installed by file through the App Library as well.
affected status or disease type, for example. Cohorts established in the Patient Directory can be referred to in other MedSavant apps, like the Variant Navigator.

**VCF UPLOADER**

The VCF Uploader facilitates the loading of genotypes into the platform. The interface for uploading VCFs, shown in Figure 3.24 has been substantially simplified over a number of iterations. In the latest version flat or zipped variant files can be dragged-and-dropped into a large component and subsequently uploaded for processing. Advanced options include the ability to toggle the application of transcript-based annotations, to perform phasing, and to set an email address to be notified when variants are ready for analysis.

**VARIANT NAVIGATOR**

The Variant Navigator app enables flexible exploration of genetic variants that have been loaded into the system. Its interface is shown in Figure 3.25 and is modelled after e-commerce websites.
like eBay and Amazon that enable faceted search and inspection of results. The app is organized into three components that together help users hone in on variants of interest through progressive disclosure: a search component, where users specify criteria for genetic variants of interest; a spreadsheet component, which provides a familiar mechanism to browse basic information about results; and an inspector component, which provides highly detailed information for specific variants of interest.

The search component enables simple construction of queries. While existing tools such as GATK and GEMINI require careful specification of search conditions using cryptic and case-sensitive strings (e.g. “DP4 > 2”), users can use the graphical interface to both discover the facets of the data that are searchable and to tune their parameters, as shown in Figure 3.26. Search conditions can be grouped, unioned, and intersected, yielding a very expressive query language.
Queries can be constructed using graphical widgets based on criteria such as patient features (e.g. age, sex, phenotype), genotype features (e.g. quality values, genomic region, functional effect) and annotations (e.g. Gene Ontology [75], OMIM [93], COSMIC [94]). The synergistic use of these types of data allows for segmentation of mutations based on disease subgroups, an important functionality when investigating complex diseases like autism spectrum disorders or cancer.

Upon executing a search, a user is given almost immediate feedback as to the effectiveness of the latest criteria as shown in Figure 3.27. Searches can be constructed incrementally, one facet at a time, with nearly instant feedback about parameter effectiveness at each step. The dynamic provision of guidance during filter construction allows for rapid exploration of the parameter space and is one of the most significant time-saving advantages of the MedSavant platform. Moreover, MedSavant saves search states internally, producing no intermediary files that need to be managed by the user. Once a search has been fine-tuned, it
can be saved and reused for reproducing results on different cohorts or on samples whose genotypes will be processed in the future.

Search results can be examined in various levels of detail using the results and inspections components as shown in Figure 3.28. The distribution of variants across the genome is depicted as a heatmap on a karyotypic ideogram. The distribution of variants can also be charted per searchable facet (as a histogram or pie chart), and between searchable facets (as a scatter plot). The full list of candidate variants is also represented in spreadsheet format with an associated inspector that displays detailed information about selected variants, including all information contained in the original VCF file and a list of nearby genes. Further, the Inspector
presents information regarding the function and relevance of nearby genes, including associated terms in the Gene Ontology, Human Phenotype Ontology (HPO) [95], and OMIM. It also utilizes GeneMania [96], a service that finds related genes based on protein interaction and other networks, to suggest other genes to consider.

**SAVANT GENOME BROWSER**

It is often informative to inspect variants in their genomic context to ascertain their predicted functional consequence. Links in the Inspector offer the ability to navigate to and browse the genomic context of a candidate variant (including the corresponding read alignments from which the genotype prediction was initially made) using the embedded Savant Genome Browser, shown in Figure 3.29. Savant is a next-generation genome browser that contains unique visualization modes for identifying and validating genetic variants. It contains multiple representations for exploring variant datasets: a traditional track view that displays variants per sample along a linear genomic coordinate system, and a streamlined non-linear display showing a condensed map, allele frequencies, and linkage disequilibrium derived from variable positions. Linking of the candidate variants list with the genome browser optimizes the inspection of results by removing the need to manually navigate to regions of interest as is required in traditional multi-tool workflows.

**Figure 3.27** Visualization of search effectiveness. The quick provision of query results and visual feedback such as this during filter construction allows for rapid exploration of the parameter space and is one of the most significant time-saving advantages of the MedSavant platform.
DISCOVERY APP

The Variant Navigator app is a powerful utility for exploring a wide range of hypotheses, but exposes a lot of functionality that can be streamlined for clinical applications. The Discovery app, shown in Figure 3.30, is a clinically-oriented derivative of the Variant Navigator that empowers clinicians to quickly identify rare and potentially causal variants in a particular patient’s genome by automatically filtering mutations based on variant quality, harmfulness predictions (e.g. from Polyphen-2 [97]), mutational effect (from Jannovar [88]), gene panels, and allele frequencies from population databases (e.g. 1000 Genome Project [53], 6500 exome...
Gene panel analysis is facilitated by this tool; for example a panel of the 56 genes recommended by the American College of Medical Genetics and Genomics (ACMG) to be reported on in the course of clinical sequencing [40] has been preloaded to assist incidental finding detection. Variants are annotated with modes of inheritance from ~2,800 genes, as described in the Clinical Genomic Database [99] and linked to Clinvar [100], to accelerate identification of clinically relevant variants.

The utility of this to suggest causal variants in relevant disease genes has been demonstrated for a patient having Retinitis Pigmentosa, an inherited degenerative eye disease that causes severe vision impairment and often blindness. The affected patient was sequenced and genotypes imported into MedSavant. Using default settings of the Discovery App, a total of 520 missense and loss-of-function (LOF) mutations were identified from this patient that passed quality control thresholds. Using the Human Phenotype Ontology to intersect these harmful

![Figure 3.29](image1.png)

**Figure 3.29** Savant genome browser app is integrated with the Variant Navigator and other MedSavant apps. Savant can be used to examine read alignments surrounding mutations of interest, to assess their quality and to view them in the context of other annotations.

![Figure 3.30](image2.png)

**Figure 3.30** The Discovery app, used to suggest the causal variant for a patient having Retinitis Pigmentosa, an inherited degenerative eye disease that causes severe vision impairment and often blindness. High-quality, harmful mutations are intersected with genes related to the HPO term for decreased central vision, yielding a single mutation in RPE65, a gene that has been previously implicated in the disease.
variants with genes associated with decreased central vision, a single missense mutation was observed in the retinal pigment epithelium-specific 65 kDa gene (RPE65) whose mutations have been previously implicated in causing the disease [101].

**ENRICHMENT APP**

A single phenotype may be caused by genetic mutations at different positions in the genome, and so it is informative to aggregate potentially deleterious mutations based on genes, gene functions, and pathways, to concentrate analyses on relevant and noticeably perturbed biological functions. The Enrichment App aggregates variants by user-specified gene-lists, and terms in the Gene Ontology [75], Human Phenotype Ontology [95], or OMIM [93]. Aggregation is a basic form of enrichment testing, and facilitates the process of identifying biological functions that are affected within the sequenced population, and ultimately understanding the genetic mechanisms of diseases.

**GOOGLE GENOMICS APP**

Growth in the size of sequencing repositories has inspired the development of cloud-based platforms for genome sequence data hosting and access. The MedSavant app for Google Genomics, shown in Figure 3.31, is a proof-of-concept that provides a visual interface for listing a large number of publicly accessible read alignment datasets remotely hosted through Google's implementation of the API [102] being designed by the Global Alliance for Genomics and Health (GA4GH), and loading them as tracks that can be interactively explored in MedSavant's genome browser. The Google Genomics App was the first implementation of the GA4GH reads API in a genome browser. While these APIs and client apps are still under development, they demonstrate the potential to efficiently deliver large data collections to researchers with minimal computational infrastructure by leverage cloud resources. Future versions will make use of Google and GA4GH APIs to integrate remotely hosted read and variant datasets into other MedSavant apps.

**MENDEL APP**

Cohort, pedigree, and inheritance model information help to resolve the segregation of variants with genetic conditions. For example, VAAST [103] is a popular probabilistic disease-gene finder that can use such information to discover the causal gene(s) for genetic disorders given a modest number of genotyped samples. The Mendel App is an internally-developed disease-gene finder that adds the ability to resolve disorders through case-control and pedigree analysis. Given
complete pedigree information, Mendel can also perform segregation based on known inheritance models. A case study using MedSavant is described in the next section.

Case Study: FORGE Canada

FORGE Canada is a national consortium whose objective is to discover the genes for a large number of rare childhood diseases [31]. For example, Joubert syndrome is a rare disease characterized by a distinctive cerebellar and brainstem malformation called the molar tooth sign [104] as shown in Figure 3.32. Common symptoms in infants affected by Joubert syndrome include intellectual disability, inability to coordinate voluntary muscle movements (ataxia), jerky eye movements (oculomotor apraxia), rapid breathing (hyperpnea), and decreased muscle tone (hypotonia). Joubert syndrome occurs in about 1 in 100,000 births, although there is a relatively high prevalence in the French-Canadian population, with several founder effects noted. SNP genotyping was previously performed on 9 affected individuals from seven families living in
the Lower St. Lawrence region, and from this data it was previously discovered that mutations in C5orf42 cause Joubert syndrome in this French Canadian population [105].

14 FORGE disorders, including Joubert syndrome, were chosen for study with Mendel based on the availability of genotype data, pedigree data, and consistency in their sequencing and genotyping pipelines. All patients from the chosen disease cohorts, including others that were used as controls, were sequenced and genotyped using a consistent bioinformatics pipeline at McGill University. Specifically, genomic DNA from patients was captured using the Agilent SureSelect 50 mb oligonucleotide library and sequenced with Illumina HiSeq2000, yielding 100bp paired-end reads. Putative PCR-generated duplicates were removed from the raw read data using Picard [61], alignment was performed using the Burrows-Wheeler Aligner (BWA) [106], and custom scripts for SamTools [25], Pileup, and varFilter were used to call variants.

A MedSavant database was created containing genotypes from all FORGE projects for which data was available, comprising 424 individuals and 138,640,418 variants identified from their samples. MedSavant was used to specify stringent quality filters (minimum coverage of 3; support from both strands; quality ≥ 50 for indels and ≥ 30 for SNVs; either exonic, splicing, or in UTR; allele frequency ≤ 0.05) yielding 427,765 variants. For each of the 14 chosen FORGE projects, Mendel was used to identify variants that segregate with the disorder using the remaining individuals, including family members of affected individuals and unaffected individuals from other projects, as controls. For Joubert syndrome, the Mendel expression

Figure 3.32 An axial section of the brain of a patient affected by Joubert Syndrome (left) compared to that of a normal individual (right). The molar tooth sign in the midbrain, emphasized as the location between the two arrows, is characteristic of individuals affected by the syndrome.
shown in Figure 3.33 produced missense and splicing mutations in C5orf42, with no other results. The positions of mutations identified by Mendel were manually inspected using the built-in genome browser. All 9 of the affected individuals were found to carry compound heterozygous mutations in this gene unlike any of the controls.

Using a similar workflow the causal gene was independently discovered for 13 of the 14 (93%) chosen disorders, listed in Table 3.4. For the disorder that could not be resolved using Mendel, two of the causal variants did not pass basic quality filters; they were validated with Sanger sequencing in the original study.

**APP ECOSYSTEM**

The MedSavant App framework supports realtime cooperation among installed apps. For example, from within the Patient Directory a user can select a patient, further filter the patient’s variants using the Variant Navigator, and inspect the results using the Genome Browser or further analyze them with Mendel as suggested by Figure 3.34. The process of interpreting genomes within this environment is conceptually different from current approaches, which involve serial processing of data using independent tools. The collaborative system simplifies development and integration of tools, and encourages the growth of an ecosystem that
<table>
<thead>
<tr>
<th>Disorder</th>
<th>Samples</th>
<th>Causal Gene</th>
<th>doi/ Pubmed ID*</th>
<th>Mendel Conditions**</th>
<th>Mendel Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESFUL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adrenal Insufficiency syndrome</td>
<td>1, affected</td>
<td>NNT</td>
<td>doi: 10.4137/JGE.511378</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; inspect for compound heterozygous</td>
<td>causal gene is 1 of 2 results, known gene</td>
</tr>
<tr>
<td>Congenital blindness, Leber congenital amaurosis 1</td>
<td>1, affected</td>
<td>NMNAT1</td>
<td>22842330</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; inspect for compound heterozygous</td>
<td>causal gene is 1 of 2 results, only after ignoring TTN, novel gene</td>
</tr>
<tr>
<td>Floating Harbour syndrome</td>
<td>5 affected, unrelated</td>
<td>SRCAP</td>
<td>23165645</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; gene has variants in at least 80% of affected</td>
<td>causal gene is only result, novel gene</td>
</tr>
<tr>
<td>French Canadian Joubert syndrome</td>
<td>9, all affected, 3 related, with 1 pair of siblings</td>
<td>CSORF42</td>
<td>22425360</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; gene has variants in at least 80% of affected</td>
<td>causal gene is only result, novel gene</td>
</tr>
<tr>
<td>French Canadian Joubert syndrome</td>
<td>3, all affected, 1 pair of siblings</td>
<td>TMEM231</td>
<td>23012439</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; gene has variants in all of affected; variant in both of siblings</td>
<td>causal gene is 1 of 4 results, only stopgain, novel gene</td>
</tr>
<tr>
<td>Hadju-Cheney syndrome</td>
<td>5 samples, some affected, parent-child and unrelated</td>
<td>NOTCH2</td>
<td>21681853</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; gene has variants in at least 80% of affected</td>
<td>causal gene is only result, known gene for a different disease</td>
</tr>
<tr>
<td>Hereditary leg dominant quadriparesis</td>
<td>3, 2 affected siblings and 1 parent</td>
<td>DDHD2</td>
<td>23176823</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; gene has variants in both of affected</td>
<td>causal gene is 1 of many results, but only gene with multiple nonsynonymous variants shared by the siblings, novel gene</td>
</tr>
<tr>
<td>Hunter syndrome</td>
<td>4, 2 affected, family</td>
<td>IDS</td>
<td>23844659</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; gene has variants in all of affected; step 3 inspect for chromosome X</td>
<td>causal gene is only result, known gene</td>
</tr>
<tr>
<td>Hutterite Syndromic ID</td>
<td>1, affected</td>
<td>THOC6</td>
<td>23821916</td>
<td>step 1 variant in at least 1 of affected gene has variants in none of controls; gene has variants in at most 1% of controls; step 3 inspect for homozygous</td>
<td>causal gene is 1 of 4 results, novel gene</td>
</tr>
<tr>
<td>Mandibulofacial Dysostosis with Microcephaly</td>
<td>4, all affected, unrelated</td>
<td>EFTUD2</td>
<td>22305528</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; gene has variants in at least 80% of affected</td>
<td>causal gene is 1 of 2 results, but only frameshift/stopgain, novel gene</td>
</tr>
<tr>
<td>MIC-CAP</td>
<td>5, 1 pair of siblings</td>
<td>STAMBP</td>
<td>23542699</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; gene has variants in at least 80% of affected; variants in both siblings</td>
<td>causal gene is 1 of 2 results, novel gene</td>
</tr>
<tr>
<td>MPPH-CM</td>
<td>5, 2 affected, 2 families</td>
<td>CCND2</td>
<td>24705253</td>
<td>step 1 variant in all of affected gene has variants in none of controls</td>
<td>causal gene is only result, novel gene</td>
</tr>
<tr>
<td>Multiple Intestinal Atresia</td>
<td>3, all affected, unrelated</td>
<td>TTC7A</td>
<td>23423984</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; variant in at least 2 of affected</td>
<td>causal gene is 1 of 4 results, novel gene</td>
</tr>
<tr>
<td>FAILED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perrault syndrome</td>
<td>2, both affected, siblings</td>
<td>HSD17B4</td>
<td>23181892</td>
<td>step 1 variant in at least 1 of affected gene has variants in at most 1% of controls; variant in none of controls</td>
<td>compound heterozygous variants could not be identified because one of these variants did not pass initial filters (QUAL value of 3.01)</td>
</tr>
</tbody>
</table>

Table 3.4 Sample information and results for the 14 FORGE disorders studied using Mendel.

* Where Pubmed IDs are not available, a doi is listed.

** For a given disorder cohort, the controls were chosen as the set of individuals not belonging to that cohort.
will increase in power as more apps continue being developed.

3.2.9 Evaluation

MedSavant has been developed since 2011 and in that time has been refined through the release of numerous major and minor versions. The client has been downloaded over 580 times and the server has been downloaded over 130 times. Demo accounts have been requested by over 100 individuals from users at academic, clinical, and industry organizations worldwide.

Like the Savant Genome Browser, the features of MedSavant has been deployed collected and implemented through processes of user centred design. We have performed participatory design, surveys, workshops, and informal interviews to solicit user requirements and feedback. User requirements were collected from discussions with close collaborators, especially scientists from the FORGE Canada and Care for Rare Consortiums, The Centre for Applied Genomics, the Molecular Diagnostics Lab at SickKids Hospital, the Laboratory Medicine and Pathology at SickKids Hospital, and the SickKids Genome Clinic. Installations have been established for each of these collaborators, who are evaluating the tool for regular use pending further development and integration. Feedback is regularly solicited from these early adopters, and new functionalities are encoded into tickets, prioritized, implemented, and included in internal versions.
MedSavant has been presented at numerous venues, including at international venues whose audience include bioinformatics tool developers, who are potential app developers, and researchers and clinicians, who are potential users.

**USER STUDY**

MedSavant was intended to make the process of genomic data analysis easier through simultaneous advancements in usability, specifically, by creating a simple interface for specifying queries with respect to genomic variants, and performance, by accelerating the speed at which the system answers such queries compared to existing tools. The experiments discussed in previous sections showed evidence of the latter, where performance gains can be achieved for facettted search by indexing and compressing genomic data as done in GEMINI and MedSavant.

To test the usability of MedSavant, a small user study was designed whereby recruited subjects would be asked to complete a realistic scenario: to download a genetic variant dataset from the internet (made publicly available by Illumina) and to iteratively filter for high quality variants. For comparison, the tasks were to be performed using both GATK and MedSavant. Handwritten observations in addition to screen capture recording were performed to track the subject’s interactions with each tool. A scripted instruction set outlining the following tasks, along with relevant URLs, was provided:

**Getting Started**


**Using GATK**

2. Download and install GATK.
3. Determine the number of variants in the VCF file.
4. Filter variants by quality $\geq 20$. How many are there?
5. Filter variants by quality $\geq 30$. How many are there?

**Using MedSavant**

7. Create a MedSavant project.
8. Upload variant file to the MedSavant project you just created.
9. Determine the number of variants in the VCF file.
10. Filter variants by quality $\geq 20$. How many are there?
11. Filter variants by quality $\geq 30$. How many are there?

The test subject enrolled in the study was a male PhD. computer science student having 11 years of computer programming experience and 10 years in the field of computational biology. Prior to the study, the subject indicated a preference for doing bioinformatics analysis using scripting and command-line tools but had limited experience using either GATK or MedSavant.

The subject was able to complete some, but not all, of the tasks using GATK and command-line alternatives. Seven distinct errors were encountered performing tasks 3-5, which generally related to GATK being stringent on the formatting of its inputs and having missing documentation on the language for specifying filter expressions. As examples of the former, GATK’s VariantFiltration tool requires an indexed reference file (e.g. human_genome19.fa, along with associated .dict and .fai files) and does not accept variant files whose entries are not karyotypically sorted (i.e. ordered chromosome 1, 2, ..., 22, X, Y). As an example of the latter, the subject was not confident if the string for filtering against quality values was “QUAL” or “QUALITY”. GATK does not provide a mechanism for determining which fields are filterable and so the subject sought and gained clarification from searching GATK’s community forums.

The user alternated frequently between using GATK, the web browser, and other tools accessible from the terminal. Ultimately, the subject gave up trying to perform tasks 3-5 using GATK and opted to perform the tasks using custom combinations of terminal utilities. The utilities used to perform these tasks included: ls, less, cd, gunzip, wget, ssh, scp, grep, cut, uniq, wc, awk, head, and tr. After approximately 20 minutes though, the user elected to move on to the tasks to be performed using MedSavant.

The subject was able to complete all of the tasks, 6-11, using MedSavant in about 16 minutes and 15 seconds. The user encountered only one error, where he attempted to create a project with non-alphanumeric characters (e.g. “="), and was prompted by the system to report difficulties here. MedSavant was able to import and annotate the input file without any issues. At no time in this portion of the study did the user reference command-line utilities. The only other tool used was a web browser, to download MedSavant and to obtain assistance creating a MedSavant project. His search for “medsavant create database” directed him to a YouTube video that demonstrated this workflow. The majority of time spent performing these tasks was in
setting up the MedSavant instance: creating a container project and loading it with data. Once
the data was loaded, though, the variant analysis tasks themselves, 9-11, took about 2 minutes
and 15 seconds. A summary of time spent using various tools is summarized in Figure 3.35 and
a comparison of key metrics measured in this study for GATK and MedSavant is shown in
Figure 3.36.

The abovedescribed study revealed a number of important usability issues in performing
bioinformatic analysis with GATK that make the process inefficient and the experience difficult
and frustrating. Friction points were identified in formatting data, specifying parameters for
command-line tools, and managing their outputs. The study suggests that these friction points
can be ameliorated through applications that account for these usability concerns to more
elegantly handle user input, list available functionalities, and present results in clear and
interpretable ways. MedSavant was evidenced to be more usable in these areas than GATK for
the given tasks. Given the results of this study, it is unlikely that a user without informatics
background would be capable of efficiently completing the specified tasks using GATK, though
additional usability studies should be performed to assess how such users perform the tasks using MedSavant.

**SUPPORT & COMMUNITY**

Support for MedSavant is provided in three ways. First, if the tool encounters an error, a user is prompted to send details which are sent to the development team by email. Second, users are encouraged to send issues and other forms of feedback via email to support@genomesavant.com. Finally, there is an online community where users can ask questions and provide responses publicly. Since its inception in 2010 the community has grown to include over 140 members.

### 3.3 Summary

The potential to gain new insights into genetic disease is currently encumbered by significant challenges in the data analysis methodologies that can be used to easily, efficiently, and intelligently identify causative genetic mutations from the large number of variants discovered through sequencing. Existing approaches either rely on complicated computational pipelines that include manual inspection only as an endpoint, or are desktop-based solutions that do not scale well for medium-to-large sequencing experiments. MedSavant is a genomic variant search engine built upon the client-server paradigm designed to meet the demands of even large population sequencing studies. It unifies the storage, annotation, filtration, prioritization, and visual inspection of variants into a powerful yet simple-to-use graphical interface that is designed for users with all levels of computational expertise, with demonstrated utility in both clinical and research settings. It is one of the only freely available systems that performs these tasks comprehensively, and as such represents an important contribution to the genomics community.

<table>
<thead>
<tr>
<th></th>
<th>GATK</th>
<th>MedSavant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Changes between applications</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>Total Time</td>
<td>20m 30s</td>
<td>16m 15s</td>
</tr>
<tr>
<td>Tasks Accomplished</td>
<td>2,3*</td>
<td>6,7,8,9,10,11</td>
</tr>
<tr>
<td>Tasks Accomplished (%)</td>
<td>50%</td>
<td>100%</td>
</tr>
</tbody>
</table>

*Figure 3.36 Summary of number of errors, changes between applications, total time spent, and number of tasks accomplished with each tool.

* This task was performed using a custom combination of command-line tools, not GATK.*
4 Conclusions and Future Work
4.1 Summary

Genomics is undergoing a revolution sparked by higher throughput and cost effective sequencing technologies. Sequencing has become a ubiquitous tool with varied inputs (e.g. DNA, RNA), scopes (e.g. whole genome sequencing, whole exome sequencing), techniques (e.g. shotgun sequencing, paired sequencing), technologies (e.g. single base sequencing, dibase sequencing), and purposes (e.g. genome assembly, gene expression analysis, genotyping).

There is potential for the development of software systems that assist in translating raw sequence data into actionable information that helps improve disease assessment (e.g. prenatal screening, risk profiling, preventative therapy), detection (e.g. diagnosis, classification, prognosis), and treatment (e.g. drug prescription, response) so that individuals who are or may be affected by genetic conditions are treated with an unprecedented level of precision and predictiveness. This new kind of medicine informed by personal genomic interpretation promises to have immense medical and economic benefits, including the replacement of multiple expensive tests with a single low cost genome sequence, reducing disease morbidity in patients with high genetic risk through early detection and treatment, as well as reducing costly adverse drug reactions.

Despite the capacity of new sequencing technologies to generate huge volumes of raw sequence data, it remains a substantial informatics challenge to efficiently analyze it. HTS technologies produce data at a rate that exceeds Moore’s Law, creating enormous technical (e.g. storage, processing) and usability issues (e.g. visualization, data complexity).

Freely available tools that are both powerful enough to make analysis interactive and user-friendly enough to be used by genomic researchers without informatics expertise are scarce. Notwithstanding a few exceptions, users are often forced to choose between powerful, specialized software that needs to be run on the command-line or alternatives that are less specialized but are graphical and user-friendly. In the absence of an end-to-end system for genetic data analysis that is both powerful and easy to use, a multidisciplinary team is required where there is constant communication between the people configuring and running the computational tools (e.g. bioinformaticians) and those interpreting their results (e.g. technicians and geneticists). Coordination between various people in the process, each with unique expertise, can be a significant bottleneck.
This thesis presents two software platforms that combine techniques from various domains of computer science, most notably data structures, databases, algorithm design, data visualization, user interface design, and user experience design, that together form a highly integrated system for interpretation of personal genomes that is powerful and easy to use.

The Savant Genome Browser is a tool for visualizing datasets in the context of a linearized genome. It is one of the first and most popular genome browsers for visualization of HTS datasets, as it contains a number of unique visualization modes for visualizing different types of genetic variation.

MedSavant, which embeds the Savant Genome browser, is a system that unifies the storage, annotation, filtration, prioritization, and visual inspection of variants into a powerful yet simple-to-use graphical interface that is designed for users with all levels of computational expertise. The system introduces novel data storage and query methodologies optimized for faceted search of genomic variants and their annotations that are demonstrably more powerful than state-of-the-art tools. These server-side advances enable interactive exploration of genomic datasets at scale — supporting fast ad-hoc querying of millions of genomic variants — while being easy to use — manageable entirely through graphical applications that have evidenced utility in both clinical and research settings. MedSavant demonstrates the utility of a powerful, integrated, interactive, user-friendly, freely available, and extensible platform for the interpretation of personal genomes.

4.2 Limitations & Future Work

The outcome of this thesis includes a number of conceptual contributions with respect to visualization and interpretation of personal genomes, and two independent software systems that embody these. The values of these software systems are assessed on an ongoing basis through real use cases; these evaluations have identified limitations that ought to be addressed in future work.

4.2.1 Use in Secure Environments

Many potential users of the system work in hospital environments that have strict networking policies to prevent unauthorized access to and within private networks. MedSavant requires communication over two dedicated ports that are not typically open on these protected
networks and so administrators that wish to install the system must therefore coordinate with network operators at their institution to ensure that access is not blocked.

It is possible to avoid network restrictions by tunneling traffic over standard protocols like SSH that communicate over ports that are usually open. While tunnelling is usually difficult to set up for casual computer users, which contradicts the ethos of the platform, it is possible to perform SSH tunnelling using Java RMI [107] and have credentials configurable through the graphical interface to facilitate this. This functionality should be considered for inclusion in future versions of the system.

4.2.2 Protecting Personal Health Information

It is an expectation, and often policy, that software deployed within such environments make efforts to protect Personal Health Information (PHI) of patients whose data is stored and processed through the system. PHI is defined in Ontario’s Personal Health Information Protection Act, 2004 [108] as identifying information about an individual in oral or recorded form. This includes the physical or mental health of an individual, the provision of health care for that individual, and their payment for or eligibility to receive health care. Since genetic information is indeed identifying information, it must be sufficiently protected as PHI.

A compliance standard for the United States that is recognized by institutions in Canada and also worldwide is specified by the Health Insurance Portability and Accountability Act of 1996 [38], or HIPAA. The Act sets out rules to protect the privacy and security of PHI; systems that are certified HIPAA compliant must, among other things:

- Implement a mechanism to encrypt and decrypt electronic protected health information.
- Establish and implement procedures to create and maintain retrievable exact copies of electronic protected health information.
- Apply appropriate sanctions against workforce members who fail to comply with the security policies and procedures of the covered entity.
- Have procedures for guarding against, detecting, and reporting malicious software.
- Implement policies and procedures to limit physical access to its electronic information systems and the facility or facilities in which they are housed, while ensuring that properly authorized access is allowed.
Because MedSavant is distributed software that can be installed in any environment, much of ensuring compliance with HIPAA is on the onus of the system administrator. However, there are additional precautions needed to be engineered into the software for it to be possible to be made compliant. For example, in the current system, third-party applications may access and use PHI data in a manner that is not compliant with HIPAA (e.g. uploading data to an external server for visualization or analysis). Restrictions need be placed on all components of the system so that they adhere to a strict and common set of privacy and security policies; adherence to HIPAA technical requirements and having a certified compliant installation may have significantly accelerated adoption of MedSavant by institutions whose data includes PHI that is not consented for public use.

4.2.3 Installation Support

Due to potential risks inherent in allowing access to sensitive data over the internet, most administrators prefer to install the MedSavant system as an in-house solution so that access is restricted to within hospital firewalls. Documentation and video tutorials for installing the MedSavant server are provided at https://www.genomesavant.com/p/medsavant/tutorial_admin/, however the provision of support for this process in restricted systems is difficult, as full access to the system is impossible without being physically on premises of the installation. Maintenance of a preconfigured Virtual Machine (equipped with the MedSavant server, the Infobright database, and network diagnostic tools) that can be downloaded and run internally may help avoid installation issues that tend to dissuade potential users from trying the solution.

Unlike Savant, which can be very easily downloaded and installed without the need for a resource-intensive server, the MedSavant server is distributed for in-house installation and configuration. The distribution of the server — which involves the installation of multiple components — in this way is a significant trade-off which was initially intended to favour security and flexibility of installation over the convenience of connecting to a hosted solution that is accessible by connecting to a remote machine running the MedSavant server. This model also avoids the need to host a centralized server, which would require computational resources and consideration of how they should be shared (e.g. storage provisioning, load balancing, etc). However, this significantly limits adoption, as MedSavant cannot be tested with one's own data.
without overcoming installation hurdles mentioned above. In the short term, a demo server that hosts 1000 Genomes Trios data has been made available for users to try.

Cloud-based computational environments that automate resource provisioning are becoming increasingly available and some cloud providers even make guarantees to secure data according to privacy policies like HIPAA. Deployment of the MedSavant system within a cloud environment is an intriguing direction for future work, as it may simultaneously increase its scalability and resource efficiency while removing an important barrier to adoption.

### 4.2.4 Compatibility and Reproducibility

The scientific process demands the highest standards of quality, ethics, and rigour. To this end, the ability for scientific outcomes to be reproduced is critically important. Yet, care to ensure that reproducibility of research is rarely practiced [109]. The MedSavant system has multiple connected components (a server, clients, and apps) each having a specific version, and while there are safeguards in place to ensure compatibility between these components and to update them when applicable, the onus remains that of the user to record versions used so that analyses performed by the system may be reproducible. Future implementations of MedSavant should make recording of versions into results straightforward, identify when updates to the system alter previous analyses, and automate reproduction of results.

### 4.2.5 Scalability

The resolution and statistical power to make conclusions using MedSavant increase as more data is entered into the system: for example, mutational burden analysis using Mendel would likely yield less spuriously implicated genes (fewer false positives) given a larger number of controls. The high-performance database and query engine are able to execute most queries required by MedSavant within a few seconds on datasets containing up to 100 million variants on a single machine running CentOS 6.5 having 8 Quad-Core Intel Xeon (E5472) Processors and 16 GB RAM. Beyond this threshold, data exploration becomes non-interactive.

It may be possible to utilize the paid, multi-threaded version of Infobright to improve performance, or alternatively “scale-up” the hardware on which the server is running. There are several problems with the scale-up approach however:

- Scaled-up machines are expensive; in most cases cost increases at a higher rate than performance.
- Scaled-up machines are single points of failure that are very expensive to replace or upgrade.
- Scale-up has limits; there is a limit to the number of processors and amount of RAM that can fit into a single machine.

An alternative strategy that has been adopted by technology companies like Google and Amazon is to “scale-out” resources [110, 111], i.e. distributing computation across many machines. The scale-out approach is less expensive: commodity hardware can be used, failures are less catastrophic, and it can be dynamically expanded through the addition of hardware.

The scale-out approach can be used to make database queries on huge datasets significantly more efficient: by partitioning, or sharding, the data to be searched across multiple machines and aggregating the results. The sharding approach works well for database operations that can be accomplished naturally by divide and conquer, i.e. the final result can be computed simply from the result of executing the query on its parts. SQL keywords like SUM, COUNT, MIN, and MAX lend well to sharding but those like LIMIT, AVG, and ROWNUM introduce complexities in aggregating the results.

As part of Google Summer of Code 2013, Miroslav Cupak created a derivative of MedSavant that sharded the database across a preconfigured cluster of computers running ICE. This work, as described in his Master’s thesis entitled “Parallelization of Query Processing in MedSavant”, significantly improves performance for large datasets [112]. A similar approach for variant storage and analysis is being developed by and in collaboration with Google Genomics [102]. Engineering the system to be able to adaptively scale to meet growing demands of the genomics community is an exciting future direction for this work.
Bibliography


41. ACMG, ACMG Updates Recommendation on "Opt Out" for Genome Sequencing Return of Results. 2014.


89. Cingolani, P., et al., A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff: SNPs in the genome of Drosophila melanogaster strain w1118; iso-2; iso-3. Fly (Austin), 2012. 6(2): p. 80-92.


