Abstractions for Genomics: Or, which way to the Genomic Information Age?

Vineet Bafna∗ Alin Deutsch∗ Andrew Heiberg∗ Christos Kozanitis∗
Lucila Ohno-Machado† George Varghese∗

November 14, 2011

Abstract

With DNA full sequencing costs dropping below $1,000, two major application drivers are: finding the genetic basis for disease (via Genome Wide Association Studies), and personalized medicine (tailoring treatments to genetic makeup as in cancer genomics). While hardware costs are falling, software costs are a major obstacle to both discovery and personalized medicine. In this paper, we propose attacking this obstacle via the use of a hierarchical set of abstractions (akin to Internet layers) for genome processing. While interface formats exist for mapped files (BAM) and variant formats (VCF), we propose a new separation of variant calling into a deterministic evidence layer that is queried by a probabilistic inference layer via a proposed Genome Query Language. GQL allows inference engines to efficiently sift only the relevant evidence (e.g., the READs that flank a SNP site) from raw genomic data.

We describe a set of exemplar queries ranging from SNP calling to phasing based on input from sequencing companies and show how evidence for these queries can be concisely captured using GQL. Discovery is modeled as finding the variants common to a given phenotype, and personalized medicine as the dual problem of finding the phenotypes corresponding to individual variations, from a database of complete individual genome sequences. We suggest that the standard implementation of GQL using relational database tables with billions of rows for each genomic location will not scale to a single human; instead, we introduce efficient indices based on compressed strength vectors to scale to large populations. We outline several benefits to separating evidence from inference including the ability to reuse genomic data across studies, the ability to logically assemble case-control cohorts, and the ability to rapidly change queries without customized programming. Finally, GQL allows us to move away from deep analysis of individual genomes towards group inference on large populations. Our paper is written for computer scientists: it starts with a model of genetic processing assuming no prior knowledge, and ends with a set of challenges in computer systems, security, information retrieval, inference, and databases.

1 Introduction: Universal sequencing: opportunities and problems

We are a product of “nature and nurture”. By this, we mean that our phenotype (the composite of all outward, measurable, characteristics including our health parameters) is a function of two things: our genotype (the DNA program inside all cells), and the environment (all the inputs to a human including food and medicine). For a computer scientist, a useful analogy can be drawn from how the output of a program (e.g., an Internet search engine) is a function of both the program and the input (keywords typed by user). Specifically, using the same input with a different program (e.g. Google search vs. Bing) is likely to result in a different output.

In this analogy, the role of the medical professional is to provide information that is either “diagnostic” (i.e. is there a bug in the program based on observed output?), “prognostic” (e.g., can we predict the output/outcome, given specific inputs such as diet?), or “therapeutic” (e.g., can any specific input such as a drug lead to the desired output). Also, the Electronic Medical Record (EMR) of a patient can be described as an archive of previously acquired inputs and outputs. While this is a useful analogy, there are subtle differences. Unlike computers, the human hardware (the body) itself is the output of programmed instructions and previously received input; thus, each program runs on unique and constantly
changing hardware. Additionally, the program itself changes over the course of the lifetime of the individual due to acquired mutations in the DNA, but we will not discuss this here.

Unlike computers, the human “program” was largely hidden away with only the hardware visible. As a consequence, medicine has traditionally been “not-personalized” with doctors providing treatments based on comparisons of the patient’s phenotype (symptoms) against empirical observations of outputs from a large number of individuals. However, there is some customization based on observation of phenotypes from ‘similar’ individuals based coarse classes such as ‘race’. All of this changed with the sequencing of the human genome in early 2000, and the subsequent drop in costs from hundreds of millions of dollars down to $1000 on small desktop sequencing machines. The ability to cheaply read the program of each human underlies the great promise of personalized medicine— treatments based on symptoms and the patient’s distinctive program (i.e., DNA).

We frame this point with a classic example. The blood-thinner drug, Warfarin, has seen myriad previous uses, including as rat-poison during World War I [26], to treat Eisenhower [14], and allegedly to poison Stalin [4]. It is now a widely prescribed anti-coagulant therapy against blood clot formation. The dosage is critical: too high, and the patient can bleed to death; too low and the drug may not prevent life-threatening blood-clots. Often, the right dosage is established through multiple visits to the clinic, and regular testing. However, recent reports [22] suggest that knowledge of the patient’s genetic program can help establish the right dosage. The approach to discover the genetic association of drug response is standard in the field, and outlined below.

- Collect a sample of affected, and ‘genetically matched’ control individuals.
- Sample the DNA and catalog variations in the populations.
- Identify and report variations that co-segregate (i.e., correlate) with the affected/control status of the individual.
- Follow up on the genetic basis of the correlation through expensive follow-up studies and experiments in animal models and clinical trials. Transfer knowledge to the clinic.

We dub this sequence the discovery workflow. Much of the discovery workflow is carried out by trained researchers, and subsequently adapted for clinical use. Even with its successes, the discovery approach has some problems. First, such studies are resource intensive and rely on the researcher’s ability to identify and sequence large cohorts of individuals with and without a disease. Second, it is unclear how to apply the study results to a specific individual, especially one that is genetically different from the investigated cohort. Finally, a specific study often looks at only a small subset of variants, and the entire data is then dumped into archives. Significant computational burden must be expended to dig out data from a previous study, and a lot of care needs to be exercised in using the data (See, for example, “$100 Genome with a 100,000 dollar analysis”—Stanford Medicine, 2010) and to make inferences across datasets from different labs.

We contrast the discovery workflow with personalized medicine. Here, a physician treating individual A may query a database either for: treatments suitable for patients who have genetic variations similar to those of A; or query for patients genetically similar to A for treatments and dosages that worked well for these patients.

To enable both discovery and personalized medicine, we suggest a shift in perspective. Instead of sequencing small cohorts on a need-to basis, we assume that each individual is sequenced at birth, and her personal genome is an integral part of her EMR (medical record), and available to be queried by trained researchers and medical personnel. This scenario is realistic considering the rapid drop in sequencing costs.

In the example of choosing Warfarin dosage for a patient, the medical team might do the following: (a) identify a collection of individuals genetically similar to the patient and on a warfarin regimen; (b) query their genomes and EMRs for genetic variation in candidate genes and warfarin dosage, respectively; and, (c) choose the appropriate dosage based on the patient’s specific genetic program. The ability to logically select from a very large database of individuals using the phenotype as a key removes the first problem described with the discovery workflow. Using genetic variations specific to an individual A as a key to return treatments (that work well for such variations) addresses the second problem. Finally, if the EMRs and genomic information is maintained in a uniform manner, and the accompanying software system has good abstractions for querying across various experiments, then the third problem (computational burden to reuse data) is also greatly eased.

Thus our major thesis is that the simple shift in perspective to assuming that the genome sequence of all individuals is available to be queried (see Figure [1] helps 1) decrease bias for discovery and 2) enables personalized medicine. Realizing this requires the ability to provide suitable abstractions that are flexible enough for stakeholders such as experiments and physicians, and yet easy to use. Our paper focuses on proposing new abstractions for genomics along with efficient implementations. In computer science, key abstractions such as virtual memory and relational data models have vastly improved
productivity and reduced software costs to keep pace with cheap VLSI hardware: our central thesis is that the same is required for genomics.

Figure 1: Genome Sequencing and personalized medicine. We consider the scenario where every individual is sequenced at birth. The Discovery workflow starts with logical selection of a subset of individuals with a specific phenotype (e.g., disease) and another without the phenotype, and then identifying genetic determinants for the phenotype. By contrast, Personalized medicine can be seen as the task of retrieving the medical records and genotypes of all patients genetically similar to a sick patient S.

Further, computer systems designers often benefit from systems thinking: making tradeoffs at the component level for overall benefits, and leveraging technology trends. We will argue that system thinking is badly required in genomics because genomic processing was invented in an era of scarcity, where genomes were scarce and sequencing was expensive; the time has come to rethink these assumptions. This article is written so that computer scientists — not just bioinformaticians who have already contributed so much, but computer scientists of every ilk and especially computer systems researchers — can engage with biologists and doctors in an essential endeavor to improve the health of the planet.

To begin the conversation, we start by describing basic genetics using programming metaphors and introduce some of the terminology (Section 2). Section 3 describes recent developments in sequencing, and continuing trends for the near future. With these preliminaries, we offer some advanced examples in Section 4 describing the state-of-the-art of genome querying, and motivating the abstractions for genome queries. Our vision for a vast genomic database built in layers is described in Section 5 — the key idea here is the separation of ‘evidence’ and ‘inference’. In Section 6, we propose an algebra for specifying genome queries, extending ideas from Databases, String matching algorithms, and Computer Systems. We end in Section 8 by outlining research directions for other areas of computer science to further this agenda.

2 Genetics for Computer Scientists

The following is a quick introduction to the relevant biology and terminology used in the sequel for computer scientists. The reader is urged to consult a standard reference for details (1).

All living creatures consist of cells; each cell can be considered to be a computer running a program which is its DNA. Humans are diploid — there are two programs controlling each cell, one inherited from the father and one from the mother. Further, each program is broken up into 23 “modules” called chromosomes and within each chromosome are sparsely scattered small functional blocks called genes. The module pairs from the father and mother are called homologous chromosomes and each human has a pair of genes from each parent. Each program uses a 4-character alphabet of bases (A,C,G,T) and is around 3 billion bases long.

Much of the ‘hardware’ of the computer consists of cellular organelles and proteins — the cellular machinery. Proteins are folded sequence of amino acids, and perform specific cellular functions such as catalyzing metabolic reactions, transducing signals, and so on. The gene contains the ‘code’ for manufacturing proteins. Each gene executes in one of many ‘ribosomes’ (analogous to a CPU). Information travels from the nucleus (where the packaged DNA resides) to the ribosome via a ‘messenger’ molecule (mRNA) that is essentially a copy of the coding DNA. The ribosome ‘reads’ the code 3 bases (one codon) at a time. Each codon is analogous to an OpCode instructing the ribosome to attach a specific amino acid to
the protein sequence being constructed. Thus, the DNA program can provide instructions for making the hardware which in turn performs all cellular functions.

![Diagram of DNA sequence assembly]

Figure 2: Repetitive nature of genomic sequence and its assembly. 40% of the human genomic sequence is made up of repetitive sequence. In this illustration, region B appears in 3 copies while regions A,C,D,E are unique. Assembly is confounded if the length of repetitive regions exceeds the length of reads. In this case the overlaps in short reads numbered 1 through 6 are equally consistent with two possible architectures. However, a single paired-end read from a clone that spans a repeat sequence sampling unique regions A and C, is consistent with the true sequence, but not with the alternative sequence (the two ends are mapped far apart).

A change (mutation) in the DNA can change the amino-acid, and correspondingly, the cellular machinery resulting in a different phenotype (output). In the Mendelian paradigm, each of the two homologous copies of the gene controls one phenotypic trait (e.g., ability to clot blood). A mutation in one gene might impact the phenotype strongly (dominant), not at all (recessive mutation) or somewhere in between. In fact, most phenotypes are complex—controlled by the paired copies of multiple genes. Nevertheless, DNA controls traits so even the simplest queries on DNA are useful — e.g., Compared to a ‘normal’ individual, are there parts of the patient’s DNA program that have mutated.

Advancements in sequencing has made it possible to cheaply scan an individual’s genetic program for mutations (or, variations). First, a physical process is used to randomly shear genomic DNA into small inserts of size 500-10000 bp. The sequencing machines deciphers the DNA from small fragments, or reads (length $L \simeq 100$ bp) at one or both ends of the inserts. Thus, the genomic information is presented as a collection of small strings over A,C,G,T, sampled from a random location on a (maternal or paternal) chromosome. Rapidly changing technologies change the throughput of sequence, insert-lengths, and read lengths, and we will discuss these parameters in the next section.

It is natural to assume that these fragments will be assembled like a giant jigsaw puzzle. However, this is complex and expensive because of the large amounts of repetitive portions in human genomes (Figure 2).

Mapping, and Variation: An alternative to assembly is to align, or map the fragments of a sampled genome (the donor/patient genome) to a reference human genome, such as a current version of a reference human genome assembly [29, 7]. Note that the assembly is a single (haploid) copy of each chromosome sampled from multiple individuals. Mapping involves finding a location on the reference where the genomic substring matches the query fragment up to a small number of errors. The errors might be sequencing errors or true variations in the query, relative to the reference. In case of multiple matches, the ‘best’ match is returned, sometimes with some alternatives. Mapping new patient genomes works because string search is cheaper than assembly and any pair of human genomes is identical to 1 in 1000 locations.

A true deviation in the sequence of the donor, relative to the reference is called a variation. The simplest variation is a SNV (single nucleotide variation). As the donor genome consists of two copies, we can be more precise in cataloging variations. A homozygous variation occurs in both copies; if it occurs in only one copy, it is said to be heterozygous. The specific value of the variant is called an allele. For example, suppose that the DNA at homologous chromosomes of
individual $A$ compared to the reference is as follows

\[ \cdots \text{ATG} \cdots \text{GAGTA} \cdots \text{Reference Assembly} \]
\[ \cdots \text{ACG} \cdots \text{GAGTA} \cdots \text{Maternal chromosome 1} \]
\[ \cdots \text{ATG} \cdots \text{GAGCA} \cdots \text{Paternal chromosome} \]

Then, individual $A$ is bi-allelic, or heterozygous at the 2 SNV sites, has the genotype $\cdots C/T \cdots C/T \cdots$, and the genotypes are resolved into two haplotypes $\cdots C \cdots T \cdots, \cdots T \cdots C \cdots$.

Sites containing SNVs that are prevalent in a population demarcate chromosomal positions as varying, or polymorphic. Consequently, these locations are called SNPs (Single Nucleotide Polymorphisms). In discovery workflows, we test populations to see if the occurrence of variation correlates or associates with the phenotype status of the individual.

All of this refers to simple variation involving one, or a small number of changes at a location. By contrast, we also have structural variations in which large (one kbp to several million bases) genomic fragments are deleted, inserted, translocated, duplicated, or inverted, relative to the reference \[27\]. The impact of these changes on the phenotype are only now beginning to be appreciated \[24\].

3 Trends in genome sequence technology

The following trends are relevant to the system design of genomic software:

- **Costs are reducing:** The Human genome project required over 22500 hours of sequencing, and cost $\sim$ $100M. Human re-sequencing costs for a redundant (15\times) coverage are now under $5K, and projected to go below $1K very soon. This implies that universal sequencing may soon be realizable, and that complex inference techniques on small populations may be replaceable by simpler inference techniques on large populations. Further, it implies that archival and analysis, not sequencing, will dominate cost.

- **Read lengths will continue to be small:** The Sanger sequencing technology used for the human genome project has long Read lengths and could achieve 100\times multiplexing on capillaries with sequence lengths between 700 - 1000, to get 100kbp in 3 hours per machine. However, capillary sequencing has largely ceded ground to new sequencing technologies that sacrifice length and sequence quality to massive parallelism and high throughput (achieving close to 600Gbp per machine per 11 day run) \[11\]. While newer, ‘single-molecule’ sequencing technologies promise to deliver longer read lengths at the expense of quality and throughput \[19\], it seems likely that in the near future, individuals will be sampled using relatively short read fragments (< 10000bp). This implies that useful abstractions must consider fragments and mapping as first-class entities.

- **Assembly continues to be hard and paired end sequencing will be useful:** Repetitive regions cover $\sim$ 40\% of the human genome. If the read-length is shorter than the length of repetitive sequence, the genome cannot be assembled uniquely. Longer reads or reads sequenced from ends of long clones (paired end reads) are necessary to resolve repeats and assemble sequences de novo. Therefore, current emphasis is on re-sequencing applications in which the sequenced reads are mapped to a standard human reference to identify variations, and correlate these variations against phenotypes. Even with improving technologies as well as assembly algorithms, it is likely that scientists will want to retain, and query the short read data used to generate the assembled sequence. Once again, this implies that fragments and reads (and paired-end reads) must be first class entities.

- **Computer System Costs will soon dominate:** Some studies have shown that the costs of disk storage for genomes is now greater than the costs of sequencing and the cost of sequencing is dropping faster than the cost of disk storage. Disk storage is only a small portion of overall system costs. This implies the need for good abstractions that can reduce system costs, potentially sacrificing information quality for modularity and making up for this loss by marshalling large quantities of data.

Therefore, we start with the premise that the genome sequence of an individual will continue to be in the form of fragments (possibly with an assembly) in the near future. We start by presenting some biologically inspired queries. The main point to note here is that there are many methods to make biological discoveries from sequence data, and no consensus on the best method. Abstractions must be flexible to handle the variety of methods some of which are surveyed in the next section.
4 Variation Calling Today

The key to efficiency in genomics is the premise that an individual’s genetic record can be summarized succinctly by a much smaller list of individual genetic variations. While we develop this premise further in our layering proposal (Figure 3)), we use this section to provide the reader with some intuition in how variants are called today. The expert should skip this section and proceed to our layering proposal. We start with querying for Single Nucleotide Variations (SNVs) in the donor genome, the simplest form of variation.

4.1 Calling SNVs

Figure 3 shows how a mutation may be called. Consider the reference allele C. We see two copies of the donor genome with a G allele, and some copies with C, indicating a heterozygous SNV. If the variation is homzygous, we expect all overlapping READs to have a G in that position. Even this simple call can be confounded. First, some of the reads may have been mapped to the wrong place on the reference; such as the top donor read in the figure. The G/T mutation may not be correct and the alignment of the incorrectly mapped read might present many variations. Even if the READ is mapped correctly, sequencing errors can present themselves as heterozygous mutations.

Thus, mutation callers use various statistical methods informed by mapping quality (e.g., the number of potential places in the genome a read can map to) of the read, the quality score of a base call, and the distribution of bases or alleles in the READs for that location. Some mutation callers even use evidence based on the surrounding locations (e.g., is there an excess of insertion/deletion events nearby suggesting problems in alignment). The decision itself could be based on frequentist, Bayesian inference, or other machine learning techniques. While SNP callers use various inference techniques, they all use the same evidence: the set of reads overlapping the location of the SNP in question.

4.2 Calling Structural Variations– Deletions, Inversions etc

In addition to small nucleotide changes, larger structural variations involving insertion, deletion, and translocation of large genomic regions also form an important source of genomic variation. Detection and characterization of large Structural Variations (SVs) is a challenging research problem. We consider the example of deletions in the donor genome (a large segment of DNA is deleted, relative to the reference). If both copies of the donor genome are deleted, the deletion is said to be homozygous; otherwise, it is a heterozygous deletion. Deletions can be detected using a number of techniques:

**Paired-end mapping:** In Paired-end sequencing, both ends of large genomic fragments (sampled randomly from the donor genome) are sequenced. These genomic fragments have been size-selected to be tightly distributed around an insert length \( L \). In a normal situation, when the paired-ends are mapped back to the reference (paired-end mapping), the two ends map around \( L \) base-pairs apart. If however, they end up mapping much further apart than expected (length-discordance), we can infer a deletion in the donor relative to the reference. For example, \( L \approx 500 \text{bp} \) is a typical insert length. If the two ends map about 700bp apart, that may be consistent with natural variation in insert size. However, if they map 10kbp apart, we can infer a deletion of 9.5kbp. If the deletion is heterozygous, we should see a mix of concordant and discordant reads at the breakpoints of the deletion.

**Depth of coverage:** Here, “depth” at a position refers to the count of reads mapped to the position. As the fragmentation is considered random, the genome is uniformly covered by read fragments. Deleted regions of the donor chromosome will have reduced coverage (roughly half for heterozygous deletions and zero for homozygous ones). Thus the depth of coverage is a clue to identifying deletions.

**Loss of heterozygosity:** Consider the SNV locations on the donor genome. At any specific polymorphic location with major allele frequency of \( p \) in a population, the Hardy-Weinberg principle suggests that a diploid individual (with 2 copies) is homozygous with probability \( p^2 + (1-p)^2 \), and heterozygous otherwise[5]. Clearly, while sampling multiple polymorphic sites, we expect a mix of heterozygous sites. When a deletion occurs, the single chromosome being sampled displays a loss of heterozygosity.
Evidence
1. Paired-end mapping
2. Depth of coverage
3. Loss of heterozygosity
4. Split reads

**Figure 3: Evidence for variation in the donor.** (a) The evidence for SNVs is provided by aligning donor reads against the reference sequence. The G/T variation might be a sequencing error as the variant reads map with too many errors. However, the G/C variation appears to be a true SNV. (b) Paired-end sequencing and mapping provides evidence for deletion in the genome. The dotted rectangle demarcates the region in the reference deleted in exactly one of the two donor chromosomes. Read ‘a’ samples the region around the deletion, and maps ‘discordantly’ in the reference; read ‘b’ maps concordantly, but with coverage about half of neighboring regions; read ‘c’ is sampled from the breakpoint and can map only at one end.

**Single end mapped, and split-reads:** When a read maps to the breakpoint of the deletion on the donor, it cannot be mapped back to the reference. In the case of a ‘clean’ deletion, the prefix and suffix of the fragment could be mapped separately, and such split-reads are indicative of deletion events. Often, the occurrence of breakpoints in repeat regions, and the presence of non-templated insertions makes this harder, but split reads are still useful clues for small deletion events.

Even, within these four categories, a number of design decisions are made by individual tools to improve the sensitivity, and specificity of detection. For example, the counts in a “depth of coverage” calculations must be sensitive to repeats and other natural variation in read coverage; the natural length variation in insert size must be accounted for in paired-end mapping; conflicting evidence from these orthogonal clues must be reconciled, as also read quality, and mapping quality must be accounted for. Thus, the prediction of deletion and other SVs remains a challenging research problem, much like the prediction of SNVs. In these scenarios, the problem of high throughput querying of genomes remains challenging.

We also note that genomics today is dominated by what we call *scarcity thinking*. Three examples of scarcity thinking follow. First, a human genome is stored using 250 Gbytes of storage which contrasts poorly with the roughly 6 Gbits required to encode 3 billion nucleotides. Even with the best compression, we find that storage is dominated by a 8-bit quality score per nucleotide. While this makes sense in a world of 1 or 2 genomes where every sequenced nucleotide is precious, it makes no sense in a world of cheap genomes. Second, a number of variant callers specialize their calls based on the type of instrument used to collect the sequence. While this makes sense in a world of a few genomes, the resulting loss of modularity has large software costs. Third, the relative lack of evidence (coverage of 10 is rare) for any variation, complicates inference today. However, in the future thousands of genomes may attest to a correlation between disease and variation, enabling new inference methods.

### 5 Software layers and interfaces for genomics

Our vision is inspired by analogy with systems and networks, where software layering has solved similar problems. For example, the Internet has successfully dealt with a wide variety of new link technologies (from dialup to wireless) and applications (from email to social networks) via the “hourglass” model using the key abstractions of TCP and IP (Figure 4a).

In the same way, we propose that Genomic Processing software be layered into an instrument layer, a compression layer, an evidence layer, an inference layer, and a variation layer that can insulate genomic applications from sequencing
technology. Note that the only way to achieve such modularity is to forgo some possible efficiencies that could be gained by leaking information across layers. For example, biological inferences can be sharpened by considering which sequencing technology is being used (e.g., Illumina versus Life Technologies) but we suggest that modularity is paramount.

Some of the initial interfaces are already in vogue with standard formats for exchange. Many instruments now produce sequence data as ‘fastq’ format; The output of mapping reads is often represented as SAM/BAM format \cite{12}, although other compressed formats are coming into vogue \cite{13}. At the highest level, there exists standard such as VCF \cite{28} to describe variants in a standardized way. The existing formats are shown bolded in Figure 4a.)

Here, we propose additional layering between the mapped tools and applications. Specifically, we separate the collection of evidence required to support a query (deterministic, large data movement, standardized) from the inference (probabilistic, comparatively smaller data movement, little agreement on techniques).

We show that the Evidence Layer (EL) can be implemented on a compute cloud while the more volatile Inference Layer can be implemented on the desktop. We assert that while Inference methods vary considerably, the Evidence for inferences is fairly standard and hence propose a Genome Query Language (GQL) that permits efficient implementation, and presents a flexible mechanism for gathering evidence for Inference.

Note that while we focus on the Evidence-Inference layer in this paper, a careful specification of a variation layer (Figure 4a)) is also important. While the data format of a variation is standardized using, for example, VCF \cite{28}, the interface functions are not. An application like personalized medicine or discovery will query the variation layer and join the variation information with phenotype information gleaned from medical records (EMRs).

5.1 The case for an evidence layer

Consider a complex biological query: “identify all deletions that disrupt genes in a certain biological network, and the frequency of those deletions in a natural population”. The bioinformatician struggles to transfer this question into a problem of statistical inference with bounds on false-positive and false-negative errors. However, the first part of any such query would be the gathering of the evidence. Here, the evidence would consist of all reads, and their mappings that satisfy certain properties. For example, the reads must overlap regions encoding genes in the given biological network, and must fall in one of three categories: (a) Concordantly mapping reads as these reads suggest heterozygosity of the deletion; (b) discordant reads: all reads where the read and its pair map to locations much further apart than could be explained by natural variation in insert length, indicating a deletion in the donor; (c) reads in which only one of the two ends are mapped, and which are proximal to discordant reads, corresponding to reads that map at the boundary of the deletion breakpoint. The development of an EL to support such queries provides several advantages

• The separation allows Inference Layer designers to start thinking of alternate forms of evidence to improve the confidence of their queries. For example, we might consider looking for "partially mapped" READs. If one pair of READs overlaps the boundary of a deletion, then that READ will match a prefix or suffix of the reference and the other may match perfectly. If the mapping shows that one maps correctly and the second does not, it provides some evidence as to the location of the boundary; further a small amount of effort can be used to decide if the non-mapped pair has a partial match with the reference.

• The EL often poses a ‘data bottleneck’ as it involves sifting through large sets of genomic reads. By contrast, the inference layer may be compute intensive, but typically works on smaller amounts of data (filtered by the evidence layer). We can implement EL on the cloud while the Inference Layer can be implemented either on the cloud or on client workstations. The evidence can easily be transported across the network interactively (Mbytes versus Gbytes). We have already seen moves by commercial cloud operators like Amazon to host the 1000 genome data sets on their infrastructure. The cloud allows rented computation on demand without the cost of maintaining a permanent cluster by every genomics researcher.

• The standardization of the EL will allow vendors time to creating a fast and scalable EL implementation. It is hard to do with the Inference Layer today as it is a moving target.

In the following, we develop this intuitive idea further by describing a Genome Query Language to support the Evidence Layer.
### 6 GQA: a relational algebra for genome queries

We show that we can express most genomic queries in a relational algebra (the Genome Query Algebra-GQA, as also an SQL-like language (the Genome Query Language-GQL). While our syntax is SQL-like, we need some special operators and relations that may not fit with available Relational Database Management Systems (RDBMS). In parallel, we are developing a database system to archive genomes and support genomic queries.

Our database has some key relations. The first is $R$ (Reads) that describe the set of read fragments output by the sequencer, and all of their properties. For example, for paired-end sequencing, $R$ could contain identifiers for each read $r$, and its paired-end. A second relation encodes a set of intervals on the genome, where each is specified by the triple “(chromosome, start-position, end-position)”. A specific, and important, case of the interval relation is the relation $G$ (the genome) where each entry is a point interval corresponding to a specific location ($\textit{locus}$) on the genome. As $G$ is large ($3 \cdot 10^9$ loci), we do not materialize $G$ explicitly, using it instead as a virtual (conceptual) relation to facilitate the relational expression of queries. A final relation $P$ describes the set of all individuals in a population, and will be discussed in a subsequent section.
As shown below, the key benefit of our proposed relational modeling is to enable manipulation via algebraic operators (the standard relational algebra operators, as well as domain-specific operators defined shortly). This in turn creates the opportunity to leverage proven techniques developed by the data management community for efficiently querying large collections of data. Regardless of the fact that some of these techniques need to be adapted or extended to our specific domain, while others can be applied directly, the overarching contribution here is to unlock the potential of large-scale data processing in the context of a huge data scale challenge.

**Standard relational algebra operators and language constructs** Standard algebraic operators we use in the examples below include:

- The projection operator \( \Pi \) where \( \Pi_X(E) \) returns, from the relation \( E \) evaluates to, the restriction of the tuples to the attributes named in \( X \). In GQL, we express this operation using

\[
\text{SELECT} \; X \; \text{FROM} \; E
\]

- The selection operator \( \sigma \), where \( \sigma_\theta(E) \) selects the subset of precisely those tuples in the result of expression \( E \) that satisfy filter predicate \( \theta \). Correspondingly, in GQL:

\[
\text{SELECT} \; * \; \text{FROM} \; E \; \text{WHERE} \; \theta
\]

- The Cartesian product operator \( \times \), where \( E \times F \) refers to all tuples of the form \((e, f)\) where \( e \in E, f \in F \). In GQL:

\[
\text{SELECT} \; * \; \text{FROM} \; E,F
\]

- The renaming operator \( \rho_N(E) \), which sets to \( N \) the name of the relation returned by \( E \).

- The group-by operator \( \Gamma \), where \( \Gamma_{G_1,G_2,...,G_k}F_{N_1:F_1(A_1),...,N_l:F_l(A_l)}(E) \), \( E \) is an algebraic expression that evaluates to a relation, \( G_1,\ldots,G_k \) are group-by attributes, each \( F_i \) is an aggregate function and each \( N_i, A_i \) are attribute names. The meaning of the group-by operator is the standard one: all tuples in the result of \( E \) are partitioned into groups, such that two tuples are in the same group if and only if they agree on the values of all group-by attributes. Thus, the groups can be identified by the value of the group-by attributes. For each group \((g_1,\ldots,g_k)\), the result contains a tuple of attributes \( G_1,\ldots,G_k, N_1,\ldots,N_l \) and values \((g_1,\ldots,g_k,n_3,\ldots,n_l)\), respectively. Here, each \( n_i \) is computed by applying function \( F_i \) to the collection of attribute \( A_i \)'s values found in the group. In GQL:

\[
\text{SELECT} \; G_1,\ldots,G_k, N_1 = F_1(A_1),\ldots,N_l = F_l(A_l) \; \text{FROM} \; E \\
\text{GROUP BY} \; G_1,\ldots,G_k
\]

By default, this collection is viewed as a multiset, i.e. duplicate values are not removed. To apply \( F_i \) after eliminating duplicates, we use the syntax \( N_i : F_i(\text{distinct} \; A_i) \). The syntax of the count aggregation is an exception, in that no \( A_i \) attribute need be specified.

**The Map join and the Project Interval operator:** As reads are sequenced, they are also mapped to the reference genome using various mapping algorithms. Some mapping algorithms map each read to a unique location, or not at all; others may choose to identify all intervals where the underlying genomic string is similar (in terms of edit distance) to the read. For now, assume that each read is mapped to its best location by a mapping algorithm \( M \), and the relation \( R \) includes the interval of the best map.

For an arbitrary relation of genomic intervals, \( A \), define a map-join operation \( R \bowtie A \) by a pair of tuples \((r,a) \in R \times A\) such that \((r.chr, r.start, r.end)\) and \((a.chr, a.start, a.end)\) ‘intersect’. Recall that we defined the relation \( G \) as the set of all genomic loci. Therefore, \( R \bowtie G \), denotes the set of all tuples \( r,g \) where \( g \) denotes a genomic coordinate that \( r \) maps to. In GQL:

\[
\text{SELECT} \; * \; \text{FROM MAPJOIN} \; R,G
\]

In many practical instances, the relation is quite large. If 33% of the genome was mapped on the average by 10 reads, \( R \bowtie G \) would contain \( \frac{1}{3} \cdot 10^9 \cdot 10 = 10^{10} \) tuples. To reduce output size, we define a special Project-interval operator \( \Pi' \). \( \Pi'_{A,ID}(R \bowtie A) \) outputs all genomic intervals after ‘fusing’ adjacent and overlapping intervals. Thus,

\[
\Pi'_{G,ID}(R \bowtie G)
\]
would result in the set of all disjoint genomic intervals with some read mapping to them. Correspondingly, in GQL:

\[
\text{SELECT MERGED G.ID FROM MAPJOIN R, G}
\]

We show, mostly by example, that a large corpus of biological queries can be supported by these abstractions.

6.1 Sample Queries

We have gathered many queries from practicing biologists to understand the power of GQA. Here, we show two examples of expressive power of the algebra. In a companion technical paper, we show that GQA’s expressive power captures the language of First Order Logic over the three relations above, as well as a signature of aggregation functions.

1. What is the genotype at a specific position (e.g., SNV)?

**Query:** Let \( A \) denote a relation containing the single point interval \( \langle \text{chr}, i, i \rangle \). The evidence for the genotype at the position is provided by alignments of READS that map to the location, and we can either query for the mapped reads, or for the alignments themselves, which are often stored as a mapped read attribute (e.g., \( R.A\_\text{ALIGNSTR} \)). Thus:

GQA: \( \Pi_{R.ID, R.A\_\text{ALIGNSTR}}(R \triangledown A) \).

GQL: SELECT R.ID, R.A\_ALIGNSTR FROM MAPJOIN R,A.

2. What are the diploid haplotypes (phased genotypes; see Section 2) across a set of linked loci in a dataset? **Query:** This is a harder query. To assemble haplotypes we need a collection of reads each of which (perhaps along with their paired-end reads) connect at least two polymorphic sites. Let attribute \( R.\text{CloneId} \) denote the clone identifier so that the paired-end reads \( r_1, r_2 \) derived from the same clone satisfy \( r_1.\text{CloneId} = r_2.\text{CloneId} \). Also, let relation \( S \) denote the collection of point intervals, one for each variant locus.

(a) Find a subset of READS mapping to the loci, and the count of sites the reads or their paired-ends map to (call this count \( c \)):

GQA: \( RC \leftarrow R.\text{CloneId} \Gamma_{c} : \text{count}(R \triangledown S) \)

GQL: \( RC = \text{SELECT R.CloneId, c = count(\ast) FROM MAPJOIN R, S} \)

GROUPBY R.CloneID

(b) Return IDs of only those reads whose count is at least 2:

GQA: \( \Pi_{ID} (\sigma_{c \geq 2}(RC)) \)

GQL: SELECT R.ID FROM R,RC

WHERE R.CloneID = RC.CloneID

AND (RC.c \geq 2)

3. What genomic loci are affected by Copy Number Variations (CNVs)? If the number of donor reads mapping to a region exceeds some threshold \( T \) then the inference might be that the region has been duplicated in the donor genome. Such CNVs have been implicated as an important variation for many disease phenotypes. To gather evidence, we would like all of the intervals where the number of mapped reads exceeds threshold \( t \) (for example). Let \( G.\text{loc} \) denote a specific chromosome and location.

(a) Compute for each location, the number of reads that map to the location:

GQA: \( V \leftarrow G.\text{loc} \Gamma_{c} : \text{count}(R \triangledown G) \)

GQL: \( V = \text{SELECT G.loc, c = count(\ast) FROM MAPJOIN R, G} \)

GROUPBY G.loc

(b) Return all ‘merged regions’ where the read count exceeds threshold \( t \).

GQA: \( \Pi_{G.\text{loc} \{\sigma_{c \geq t}(V)\}} \)

GQL: SELECT MERGED RS.loc FROM V

WHERE V.c > t

4. Identify all regions in the donor genome with large deletions. As discussed earlier, the evidence for deletion comes from a variety of sources. We use discrepant paired-end mapping. Paired-end reads from clones of length 500 (for example) should map \( \approx 500 \text{bp} \) apart on the reference genome. If instead, the ends happen to map discrepantly far (e.g., \( \ell \) apart for some \( \ell \gg 500 \), like \( \ell \approx 10000 \)), they support the case for a deletion in the donor genome. Thus, our goal is to identify all regions with at least \( t \) discrepant paired-end reads.
(a) Use a join in which each record contains
the mapping locations of the read as well as its paired-end.

\[ H_1 \leftarrow (\sigma_{R.CloneID=P.CloneID}(R \times (\rho_P(R) \bowtie G))) \]

GQL: READS already contains this join.

(b) Select records containing discrepant reads.

\[ H_2 \leftarrow \Pi_{R.*}(\sigma_{|R.loc-P.loc|>10000}(H_1)) \]

GQL: \( H_2 = \text{SELECT } * \text{ FROM READS} \)
\[ \text{WHERE } \text{abs(loc - mateloc) > 10000} \]

(c) Select intervals containing at least \( t \) discrepant reads.

\[ \Pi^t_{G.loc} \sigma_{c>\text{t}}(\Gamma_{c=count}(H_2 \bowtie G)) \]

GQL: \( \text{SELECT MERGED G.loc FROM } H_2 \)
\[ \text{GROUPBY G.loc, c=count(*)} \]
\[ \text{WHERE } c > \text{t} \]

### 6.2 Population based queries

The true power of querying genomes comes from the ability to query populations. Indeed, existing tools (Samtools) allow for the ability to extract reads from multiple individuals at specific locations corresponding to polymorphic sites. We want to extend the full power of genomic queries to interrogate populations. Recalling the warfarin example, we would like to query for warfarin dosage and genetic variation in candidate genes (genes identified through a discovery work flow) among individuals on the warfarin regimen. Specifically, we could be interested in copy number changes.

**Colloquial:** Report warfarin dosage and genomic intervals in individuals s.t. (i) the individuals are on the warfarin regimen, and (ii) the genomic intervals intersect with some gene in a candidate set \( E \), and (iii) the copy number of mapped reads is at least twice the expected coverage in the interval.

Consider a population relation \( P \) of individuals that have been sequenced. Individuals in the population have ‘phenotype’ attributes. The attributes could be, for example, diseases, treatment regimen (“P.W” is set for individuals taking warfarin), and could be categorical, or continuous valued (“PWD” for warfarin dosage). The natural join \( P \bowtie R \) describes relations \((p, r)\) in which attributes of read \( r \) are linked to the attributes of individual \( p \). To answer the query, first we identify all reads that map to gene intervals \( E \), and are sampled from individuals on warfarin regimen.

\[ W \leftarrow \sigma_{P.W}(P \bowtie R \bowtie E) \]

Next, count the number of reads at each location

\[ W' \leftarrow \Gamma_{c=count(R.ID)}(W) \]

Finally, we project the locations and reads where the coverage is higher than threshold \( T \).

\[ \Pi_{G.loc,R.ID} \sigma_{c>T}(W') \]

A similar idea applies to a personalized workflow, where we ask: report warfarin dosage in individuals with copy numbers matching the query individual \( p \) at the candidate set \( E \).

To gather evidence, we collect the set of reads and their copy numbers mapping to \( E \) in all individuals on a warfarin regimen. Thus, we have

\[ W \leftarrow \Gamma_{c=count(R.ID)}(W)\sigma_{P.W}(P \bowtie R \bowtie E) \]

From \( W \), we select the individuals and READS whose copy counts ‘match up’ with \( p \)

\[ \sigma_{p.c\equiv p.c}W \]

**Group Inference without Accurate Individual Inference:** The ability to query populations has an important benefit. For individual genomes, calling SNVs accurately is often challenging. The Inference layer assigns quality values to individual SNV calls (e.g., SNV at Chr 10:10500 with probability 0.8), and these confidence values are used for association with phenotypes in a segregating population. By contrast, population queries allow us to ask for evidence about arbitrary subsets of large groups of users based on various characteristics of these users and doing statistical inference directly on
these large groups. They allow us to bypass the determination of individual SNVs. Often, individual SNVs may have very little evidence for low-coverage sequencing. However, if a large number of affected individuals in a population (e.g., 800 out of 1000) all show the same SNV, while controls do not, we can reliably predict an association even with unreliable calls for individuals. While more work is required to demonstrate the benefits of group inference, the important point is GQL provides query support for group inference.

7 A prototype implementation

We have developed a prototype implementation, genomequery of a subset of GQL (Figure 5), with visualization provided by the open-source tool jbrowse [23]. The uploaded genome (in BAM format) can be queried using a simple text interface that allows the user to write a GQL query. The query is compiled and executed, and the output returned as a (smaller) BAM file that can be visualized using jbrowse, or downloaded to the client for further analysis.

Figure 5: A prototype implementation of GQL, with visualization using the open source tool jbrowse [23]. Discordant paired-end reads supporting deletions can be seen.

The implementation of genomequery has a customized parser that converts GQL to an intermediate representation resembling GQA. Thus, there are customized procedures for each of the algebraic operations, with some concessions for efficiency (mainly w.r.t memory). Specifically, we use interval trees to implement $\triangledown$, and customized indices (including Strength Vectors; see below) for efficient querying. Details on the implementation will be provided elsewhere.

8 The way forward: challenges for computer scientists

At this point, we have made the case for a set of genomic layers, including an Evidence Layer where the genomic evidence is retrieved using a Genome Query Language. Successful implementation of this vision will depend upon some new ideas from computer science: the specific areas are marked in the subsection titles

8.1 Scope of querying – Database Theory

On the language design front, we ask if the Genome Query Language (calculus) GQL and the Genome Query Algebra GQA are sufficiently powerful to address all Evidence Layer queries needed in practice to truly support the inference layer. The goal is to push into the Evidence Layer as much of the data-intensive computation as possible while preserving performance (note that, absent this desideratum, any query language that can return a copy of the relations provides trivial support by relegating the entire computation to the inference layer). Our design has been guided so far by the relational gold standard.
That is, GQL and GQA have identical expressive power, which coincides with that of first order logic over the schema of the three relations $R, G, P$, a signature of aggregation functions, and a group-by operator. As an immediate consequence, this allows the usage of GQL as a highly declarative language for application developers, and the deployment of GQA as the internal representation used to optimize and evaluate query plans. Relational experience suggests that special-purpose indexes and optimization will play a crucial role in pursuing this goal.

Our preliminary study of high-frequency analysis tasks has yielded a query exemplar that is covered by the expressive power built into the first-cut language draft. Further user feedback may require extensions. In implementing them, care must be taken to balance expressive power with efficient evaluation.

8.2 Efficient querying — Database Implementation

As a first step assume that queries on populations will automatically be decomposed into queries on individuals. Consider queries of the general form $\Pi_a \sigma_b (R \bowtie G)$. The two steps are

1. Select for relations that satisfy constraints $b$.
2. Project (while removing duplicates) on to attributes $a$.

We present some straightforward indices that help speed up computations. Consider an index $\text{LOCATION TO READS}$, where $\text{LOCATION TO READS}(\ell)$ is a pointer to first read $r$ that maps to the point-interval $l$. For each individual, we keep a compressed index of the mapped reads in memory. The index can be used for Select operations based on specific locations (e.g., reads that map to specific genes).

However many queries involve scanning the entire genome for maximal intervals. For example, find all maximal regions where there is a disproportionate increase in the number of mapped reads (High copy number). The corresponding GQA is given by

$$\Pi_{G,\text{loc}} \sigma_{c \geq t_{G,\text{loc}}} \Gamma_{c,\text{count}(R, ID)}(R \bowtie G)$$

Generalizing, we may have other constraints encoded by $\theta$, and consider queries of the form

$$\Pi_{G,\text{loc}} \sigma_{c \geq t_{G,\text{loc}}} \Gamma_{c,\text{count}(R, ID)}(R \bowtie G)$$

For efficient implementation of these queries, we construct special indices that allow filtering for READS according to a user-defined constraint. Define a strength vectors $S_\theta$ for a constraint $\theta$ as a vector of length $G$ (the entire genome)

$$S_\theta \leftarrow \Pi_{G,\text{loc},c,\text{count}}(G, \text{loc}) \Gamma_{c,\text{count}(R, ID)}(R \bowtie G)$$

Thus for any location $\ell \in G$, $S_\theta[\ell]$ gives the strength of the evidence at that location, and can be precomputed for common constraints $\theta$. Clearly, the strength-vectors allows us to efficiently search for intervals with high strength. Strength vectors are memory intensive. To get around that, we choose a minimum cut-off such that only intervals above the cut-off are interesting. Define a compressed strength vector $C_{\theta, t}$ as a sorted sequence of intervals $i_1, i_2, \ldots$ such that each $i_j$ is a maximal interval satisfying $S_\theta[\ell] \geq t$ for all $\ell \in i_j$. In other words, we only store the subintervals of the genome where the strength for the predicate is sufficiently high. Of course, if the user changes the threshold $t$ to a lower value, the strength vectors have to recomputed using $\text{LOCATION TO READS}$.

If the compressed strength vector is $100 \times$ smaller than the strength vector, it will be correspondingly faster to read it off disk. For large populations, where the compressed strength vectors must be loaded off the disk iteratively, this offers a great speed advantage. The trick is to choose thresholds appropriately so that the inference layer can get a superset of what it needs, but that is still smaller than the set of all READS.

8.3 Interface with Electronic Medical Records – Information Retrieval

We have so far glossed over the effort required to record phenotypes associated with each sequenced individual: a list of diagnoses (e.g., diabetes Type 1), clinical findings (e.g., elevated HbA1C), and therapies (e.g., insulin), as well as demographics (e.g., age and gender). Such information is already in patient medical records. Initial results from the eMERGE network indicate that, for a limited set of diseases, it is possible to utilize the EMR for phenotype characterization in genome
wide association studies (GWAS) within a reasonable margin of error. It is anticipated that most healthcare institutions will be utilizing EMRs by 2014, given incentives provided by the HITECH act, and it is also expected that more genome information will become part of the EMR, well beyond the cursory data on select SNPs that is available from GWAS. This will require significant modification of EMR systems so that the personal genome information can be utilized in real-time, at the point of care. Given the increasing adherence to standards for interoperability in clinical documents and important advances in biomedical natural language processing, a large portion of the EMR is now structured in a way that all data from a given patient can be retrieved efficiently. However, there is essentially no effective integration of genotype and phenotype data, hence there is significant work remaining that is related to software engineering. Furthermore, several issues related to querying across a group of patients need to be addressed. Although high level abstractions of the information obtained from genome sequences are already part of the EMR as laboratory tests (e.g., specific SNPs in BRCA1 or 2), systems that allow whole genome sequences to be queried in real-time to assist in clinical decision making are yet to be developed. Furthermore, GQL should be useful both for interrogating a single genome or interrogating several genomes across groups of individuals. Most EMR systems in the market today are based on hierarchical databases optimized to retrieve all data from a single patient. Significant transformations of transaction-based EMRs are needed to allow efficient queries on multiple patient records. For example, relational databases are implemented to allow efficient queries for populations, constituting the basis for clinical data warehouse systems. GQL should integrate with these systems so that EMR data could be queried together with genome data.

8.4 Privacy concerns – Computer Security

The genome is the ultimate unique identifier. All privacy is lost once we have access to the genome of an individual, but the current regulation, the Health Information Portability and Accountability Act (HIPAA) is silent about this identifier. Although the Genetic Information Nondiscriminating Act (GINA) addresses accountability for use of genetic information, privacy laws will need to change to ensure that sensitive information is available only to the appropriate person. Regardless of what the privacy laws are/will be, they all hinge on some definition of which information about an individual is “leaked” when some study results are disclosed. Checking that a given study satisfies a specific privacy definition requires formal reasoning about the data manipulations that generated the disclosed data, which is impossible without a declarative specification of such manipulations, with rigorous semantics. This is exactly what GQA queries offer. Of course, an equally rigorous semantics for privacy, adopted legally, is required to complete the picture. Current efforts involve the investigation of differential privacy, as well as multiple types of de-identification procedures. While this is not within the scope of this article, GQA at least enables a design process to start. Typically, such a design process for privacy measures tries to maximize the class of allowable study outputs.

8.5 Provenance – Software Engineering

GQL is an ideal vehicle for recording provenance of study conclusions, by recording the set of queries launched for a specific study. The declarative nature of such queries renders them orders of magnitude more readable by humans and programs than the scripts used currently to code the data manipulations. Such scripts are usually not publishable, because they consist of code that is often too ad-hoc for human readability, and because they span various programming languages, which are often too rich and low-level for automatic analysis. Publishing the set of queries along with their results would significantly enhance the clarity of the study’s claims, as well as its repeatability.

A further benefit of storing study result provenance as queries is to pave the way for tools that react automatically when the input data to the study changes (an all too common event). This happens when the input data producer (possibly a set of autonomous third-party labs), adds new data entries, or retracts or corrects old ones. By analyzing the query expressions, changes to the study’s input data can be automatically propagated to the output in an incremental fashion, without re-running the entire expensive query from scratch. The relational analogy to this scenario is known in the database literature as ‘incremental view updating’, and is being used successfully in data warehousing applications.

Finally, the practice of disclosing provenance queries along with each study result would enable scientists to reuse the data of previously published computing-intensive studies. Instead of running her expensive queries directly on the original input databases, the investigator would launch an automatic search for previously published studies whose provenance queries correspond to (parts of) the computation needed by her own queries. The results of the provenance queries can be directly imported and used as partial results of the new study’s queries, skipping re-computation. This scenario corresponds
in relational database practice to that of ‘rewriting queries using views’. Views are cached results of previously executed queries, each result annotated with the actual query expression. In essence, rewriting algorithms try to match query and view expressions against each other to determine reusability, thus relying crucially on the availability of the query expressions.

8.6 Inference at Scale — Learning Theory, Probabilistic Inference

We have suggested that learning the correlation between diseases and variations can be tackled differently if there are a large number of genomes. Perhaps, it is less critical to evaluate individual variations for such a discovery problem, because erroneous variations are unlikely to occur over a large group of randomly chosen individuals. More generally, what other inference techniques are there that leverage the presence of data at scale. For example, Google leverages the large data it has to find common misspellings. Note that carefully screening individual variations is still needed for personalized medicine.

8.7 Commoditizing Genomics – Computer Systems

Genomics and personalized medicine must be commoditized to be successful. Such commoditization requires computer system research. For example, Amazon has already moved a number of genomes in the 1000 genomes project where they are accessible via S3 storage. However, the amount of data is so vast that manipulation and visualization over the cloud is still expensive and time consuming. We see several avenues for improvement. First, since most genomes are read only, it makes sense to leverage new technologies such as solid-state disks (SSDs). Second, efficient decomposition between the cloud and the workstation will be key to reducing data traffic in and out of the cloud, a major bottleneck today. Our layering suggests that the Evidence Layer be deployed in the cloud and be accessible via GQL, and the results be visualized or inferred in a workstation. Much clever decompositions are clearly possible. Third, while genomics has been dominated historically by expensive parallel computers it will be important to tailor processing software to cheap multicore CPUs to improve locality of reference.

9 Conclusion

Genomics is moving from an era of scarcity (a few genomes with imperfect coverage) to abundance (universal sequencing with high coverage and cheap resequencing when needed). This shift requires rethinking genomic processing from ad hoc tools and pipelines that support a few scientists to commodity software that can support a world of medicine. The history of computer systems teaches us that as systems move from scarcity to abundance, modularity is paramount; ad hoc software is replaced by a set of layers with well defined interfaces as in Operating Systems (application, kernel, drivers) and networks (TCP, IP, Link Layers). That these trends have been recognized by industry can be seen by the shift from machine specific formats such as Illumina to standards such as BAM, and from vendor specific variant formats to VCF. The 1000 genomes project has gained momentum and a large number of sequences are already accessible. However, much of the progress today has been to define data formats without powerful interface functionality; to use an Internet analogy this is as if TCP packet formats were defined without the socket interface.

Our paper proposes going beyond the layering implicit in current industry standards, and suggest that such layering can enable personalized medicine and discovery. Specifically, we proposed the separation of evidence from inference, and the separation of individual variation from variation across groups (Figure 4)). More concretely, we propose a specific interface between the EL and the IL via a Genome Query Language (GQL). While GQL is based on a relational model using a virtual interval relation, we have found that effort is required beyond standard relational optimization to allow GQL to scale to large genomes and large populations.

We described several benefits to separating evidence from inference. A vast genome repository accessible by GQL offers the ability to reuse genomic data across studies, the ability to logically assemble case-control cohorts, and the ability to rapidly change queries without time consuming ad-hoc programming. GQL also offers the ability to reduce the quality of inference on an individual basis when doing group inference on large populations.

We described some simple ideas to scale GQL to populations using compressed strength indices and doing EL processing in the cloud. However, much work remains to be done to complete this vision including better large scale inference, system optimizations to increase efficiency, information retrieval to make medical records computer readable, and security mechanisms. While the work is vast, so is the opportunity. It is not often that one gets the chance to work on a large system such as the Internet or UNIX that can change the world.
10 Acknowledgment

The authors acknowledge support from the NIH sponsored iDASH project (Grant U54 HL108460), NIH 5R01-HG004962, and a CSRO scholarship to CK. Some of us would also like to thank Rajesh Gupta, Vish Krishnan, Ramesh Rao, and Larry Smarr for useful discussions and support.

References


