Efficient genotype compression and analysis of large genetic-variation data sets

Ryan M Layer[1], Neil Kindlon[1], Konrad J Karczewski[2], Exome Aggregation Consortium[3] & Aaron R Quinlan[1,4,5]

Genotype Query Tools (GQT) is an indexing strategy that expedites analyses of genome-variation data sets in Variant Call Format based on sample genotypes, phenotypes and relationships. GQT's compressed genotype index minimizes decompression for analysis, and its performance relative to that of existing methods improves with cohort size. We show substantial (up to 443-fold) gains in performance over existing methods and demonstrate GQT's utility for exploring massive data sets involving thousands to millions of genomes. GQT can be accessed at https://github.com/ryanlayer/gqt.

For the majority of common human diseases, only a small fraction of the heritability can be attributed to known genetic variation. A widely held hypothesis is that the remaining heritability can be explained in part by rare, and thus largely unknown, genetic variation in the human population[1]. On the basis of current and forthcoming efforts, it is predicted that more than 1 million human genomes will be sequenced in the near term[2]. Integrated analyses and community sharing of such population data sets will clearly be critical for future discovery. In aggregate, the resulting data sets will include trillions of genotypes at hundreds of millions of polymorphic loci. Therefore, the development of more effective data exploration and compression strategies is crucial for broad use and future discovery.

The Variant Call Format[3] (VCF) defines a common framework for representing variants, sample genotypes and variant annotations from DNA-sequencing studies (**Fig. 1a**), and it has become the standard for genome-variation research. However, because VCF is intentionally organized from the perspective of chromosomal loci to support 'variant-centric' analyses, it is ill suited to queries focused on specific genotype, phenotype or inheritance combinations, such as which variants are homozygous exclusively in affected women (**Fig. 1b**). GQT introduces a complementary 'individual-centric' strategy for indexing and mining very large genetic-variation data sets. GQT creates an index of a VCF file by transposing the genotype data to facilitate queries based on the genotypes, phenotypes and relationships of one or more of the individuals in the study (**Fig. 1c**). Once transposed, variant columns are sorted by allele frequency to take advantage of the fact that the majority of genetic variation is extremely rare in the population[4,5] (**Fig. 1d**). Reordering by allele frequency greatly improves data compression, as it yields much longer runs of identical genotypes than transposition alone (**Supplementary Fig. 1**). GQT also utilizes an efficient data-compression strategy based on Word-Aligned Hybrid (WAH) compressed[6–8] bitmap indices of the sample genotype information (Online Methods and **Supplementary Figs. 2–4**). The combination of genotype transposition and WAH compression maximizes query performance by allowing direct inspection of genotype data without decompression.

In addition to indexing genotypes, GQT can also create a simple database from a pedigree (PED) file describing the names, relationships, multiple phenotypes and other attributes of the samples in an associated VCF file (**Fig. 1e**). The sample database complements the GQT index of the original VCF or binary VCF (BCF) file and allows GQT to quickly identify the compressed sample genotype bitmaps that are germane to the query. For example, a query could search for variants where all affected individuals ("phenotype==2") are heterozygous ("HET") (**Fig. 1e**). GQT uses the sample database to find compressed genotype bitmaps of the affected individuals. Once these individuals have been identified, the relevant bit arrays for heterozygous genotypes are AND'ed (**Supplementary Fig. 2b**) to return the single VCF record in which all affected individuals are heterozygous. With this structure, queries can exploit any attribute that is defined in the PED file, and one can combine multiple criteria. This functionality enables, for example, searches for *de novo* mutations in a multigenerational pedigree (**Supplementary Fig. 5**) and variants having markedly different minor allele frequencies in different world subpopulations or case-versus-control samples. Query results are ordered by genome coordinate and returned in VCF format, which supports sophisticated analyses combining GQT queries with other, variant-centric tools such as BEDTOOLS[9], VCFTOOLS[3] and BCFTOOLS.

GQT's genotype index is a complement to the existing variant-centric indexing strategies[3,10] available for data sets in VCF (or BCF). GQT creates two additional indices (BIM and VID; **Supplementary Fig. 6**) that allow variants satisfying a query to be quickly returned in VCF. The storage overhead of the GQT indices is minimal with respect to the size of the underlying data set. Moreover, index size relative to the size of the VCF

[1]Department of Human Genetics, University of Utah, Salt Lake City, Utah, USA. [2]Analytical and Translational Genetics Unit, Harvard Medical School, Boston, Massachusetts, USA. [3]A list of members and affiliations appears in **Supplementary Note 1**. [4]Department of Biomedical Informatics, University of Utah, Salt Lake City, Utah, USA. [5]USTAR Center for Genetic Discovery, University of Utah, Salt Lake City, Utah, USA. Correspondence should be addressed to R.M.L. (ryan.layer@utah.edu) or A.R.Q. (aquinlan@genetics.utah.edu).

RECEIVED 5 JUNE; ACCEPTED 7 OCTOBER; PUBLISHED ONLINE 9 NOVEMBER 2015; DOI:10.1038/NMETH.3654

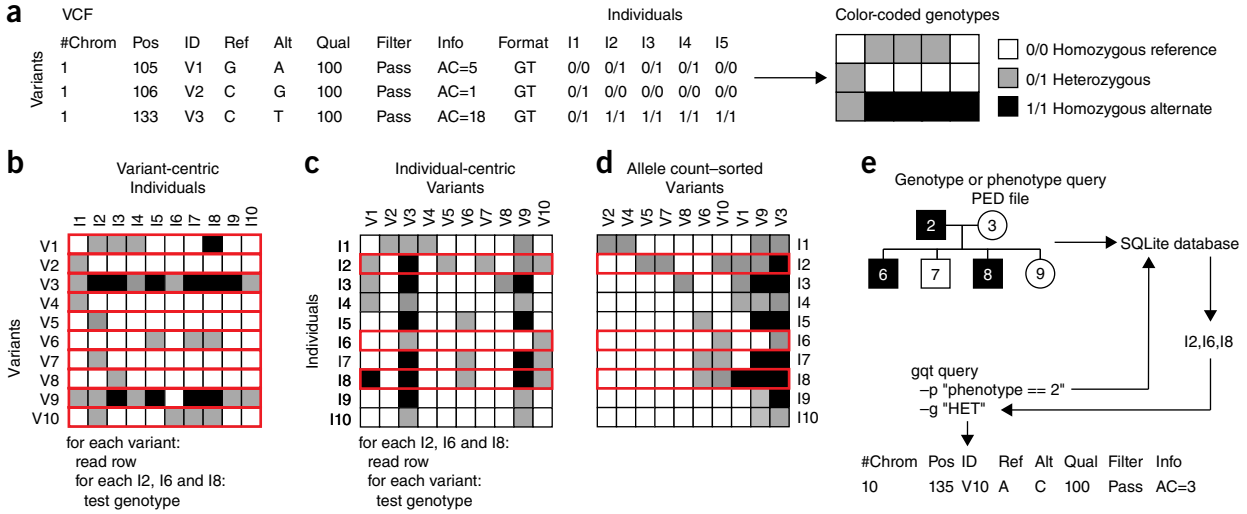

Figure 1 | Creation and data exploration of an individual-centric genotype index. (**a**) The variant-centric VCF standard is a genotype matrix in which rows correspond to variants and columns correspond to individuals. (**b**) Each variant row must be inspected to test all of the genotypes of specific individuals. (**c**) GQT transposes the matrix such that rows (data records) represent the full set of genotypes for each individual, which better aligns the data to individual-centric questions and algorithms. (**d**) Sorting the columns of an individual-centric matrix by alternate allele count improves compressibility. All genotypes for each sample are converted to WAH compressed bitmaps (**Supplementary Note 1**). (**e**) GQT will create an SQLite database of a PED file describing the familial relationships, gender, ancestry, and custom, user-defined sample descriptions. The resulting database allows GQT to quickly extract the specific compressed bitmap records in the genotype index corresponding to a query. Once the compressed bitmaps for the relevant samples have been extracted, they are compared so the subset of variants that meet the genotype requirements can be quickly identified (in this example, all individuals must be heterozygous, as in variant V10). Chrom, chromosome; Pos, position; Ref, reference; Alt, alternate allele; Qual, quality score.

file continues to decrease as the cohort size grows, as most new variation discovered from additional samples is very rare[4,5]. Comparisons to PLINK and BCF based on the 1000 Genomes Project (phase 3) VCF files demonstrated that the GQT compression strategy was on par with the compression scheme used in BCF (**Supplementary Fig. 7** and **Supplementary Note 1**). In particular, the GQT genotype index alone (i.e., excluding the ancillary BIM and VID indices) represented merely an additional ~10% (12.08 GB) storage burden beyond the requirement of the phase 3 1000 Genomes data set in BCF.

A typical tradeoff for data-compression algorithms is the cost of decompressing data before analysis[11]. We designed the GQT indexing strategy precisely to avoid this tradeoff and achieve efficient queries of cohorts involving thousands to millions of individuals. To demonstrate this, we compared the query performance of GQT to that of both BCFTOOLS and a comprehensive update to PLINK (v1.90). First, we considered the time required to compute the alternate allele frequency among a target set of 10% of individuals from the 1000 Genomes VCF data set (**Fig. 2a**). Compared with BCFTOOLS, which required 1,517.5 s, both GQT and PLINK were substantially faster, requiring 58.4 s (26.0-fold increase in computation speed) and 156.3 s (9.6-fold increase), respectively. We emphasize that GQT's performance advantage improved as the number of individuals increased, whereas PLINK's performance remained relatively flat. Moreover, matching variants were identified almost instantly, and thus the majority of GQT's runtime was spent returning the VCF results. For example, when the GQT "count" option was invoked to simply return the number of matching variants, the runtime dropped to 4.2 s. We also compared the time required to identify rare (alternate allele frequency of <1%) variants among a subset of 10% of the individuals (**Fig. 2b**). In this case, GQT was up to 45.8 times faster than BCFTOOLS (51.5 s versus 2,360.5 s).

When we considered the Exome Aggregation Consortium (ExAC; version 0.3) variant data set (9.36 million exonic variants among 60,706 human exomes), we found that the GQT index was only 0.2% the size of the VCF file (28 GB versus 14.1 TB), reflecting a storage requirement of merely 0.38 bits per genotype. Rare variants were found in only 2.1 min (9.98 s when excluding the time required to report the variants), reflecting a 443-fold improvement over BCFTOOLS (931.4 min) (**Supplementary Table 1**). On the basis of simulated data sets involving 100–100,000 individuals on a 100-Mb genome, it was clear that GQT's relative data compression and query performance continued to improve with larger cohorts. Although simulating variants from 1 million or more individuals was computationally intractable for this study, extrapolation suggested that GQT's query performance for a cohort of 1 million genomes was at least 218-fold faster than that of BCFTOOLS (**Fig. 2c**,**d**).

Because the GQT indexing strategy is fundamentally optimized for questions that involve comparisons of sample genotypes among many variant loci, it is also well suited to many common statistical and population-genetics measures. For example, as the basis for principal-component analysis of the 2,504 individuals in the 1000 Genomes data set, GQT produced a similarity matrix using the number of shared, nonreference variants in 207 min (**Fig. 2e**). A similarity matrix for the 347 admixed American individuals required merely 3 min. GQT also computed the Weir and Cockerham[12] F_{ST} statistic for all 84.7 million variants in the phase 3 1000 Genomes data set in 74 s, reflecting a 146-fold increase in computation speed over VCFTOOLS (**Supplementary Fig. 8**). These examples demonstrate how GQT indices could empower

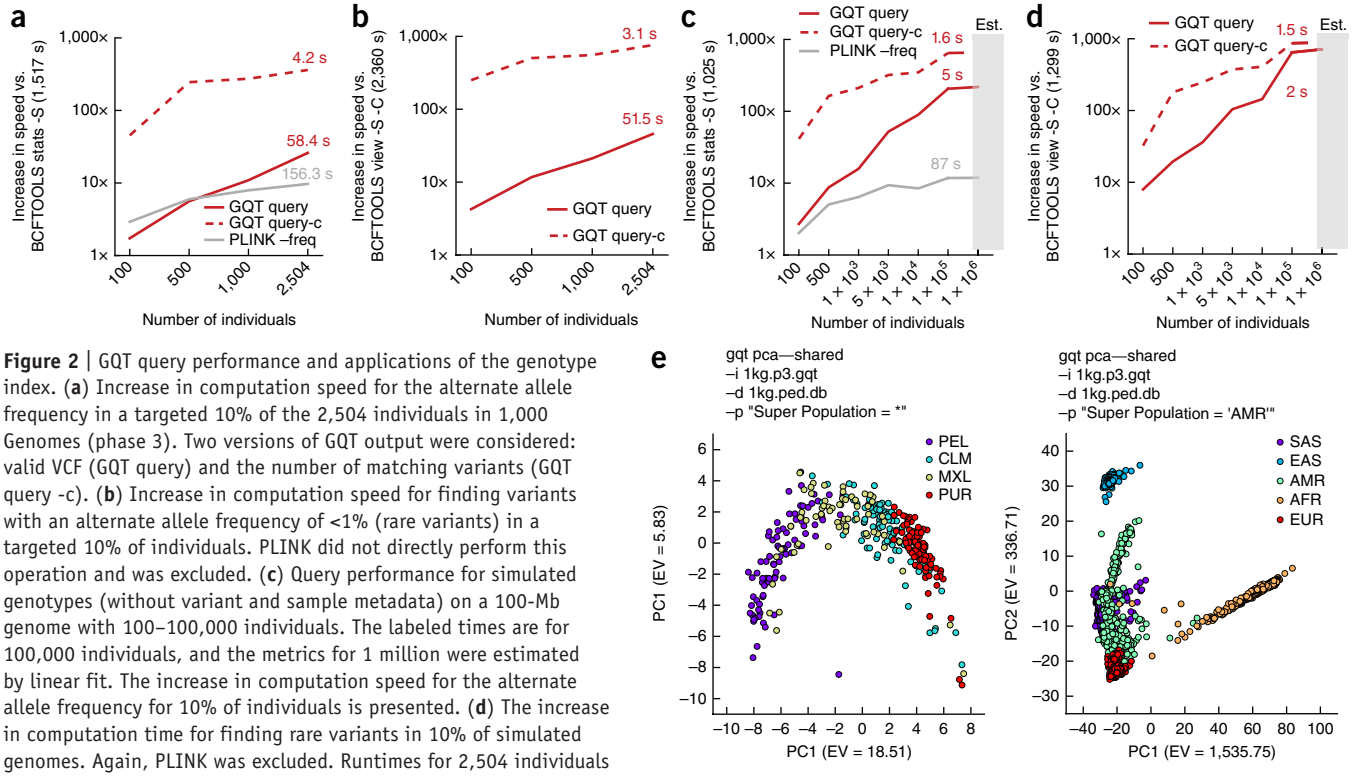

Figure 2 | GQT query performance and applications of the genotype index. (**a**) Increase in computation speed for the alternate allele frequency in a targeted 10% of the 2,504 individuals in 1,000 Genomes (phase 3). Two versions of GQT output were considered: valid VCF (GQT query) and the number of matching variants (GQT query -c). (**b**) Increase in computation speed for finding variants with an alternate allele frequency of <1% (rare variants) in a targeted 10% of individuals. PLINK did not directly perform this operation and was excluded. (**c**) Query performance for simulated genotypes (without variant and sample metadata) on a 100-Mb genome with 100–100,000 individuals. The labeled times are for 100,000 individuals, and the metrics for 1 million were estimated by linear fit. The increase in computation speed for the alternate allele frequency for 10% of individuals is presented. (**d**) The increase in computation time for finding rare variants in 10% of simulated genomes. Again, PLINK was excluded. Runtimes for 2,504 individuals from 1,000 Genomes and for the simulation were similar because the total numbers of genotypes were nearly identical. (**e**) A principal-component analysis (PCA) of all variants from 1,000 Genomes (phase 3) took 207 min for 2,504 individuals and 3 min for 347 AMR individuals. PEL, Peruvians from Lima, Peru; CLM, Colombians from Medellin, Colombia; MXL, individuals of Mexican ancestry from Los Angeles, California; PUR, Puerto Ricans from Puerto Rico; SAS, South Asian; EAS, East Asian; AFR, African; EUR, European; EV, eigenvalue.

other statistical genetics software and serve as a framework for future method development.

Although GQT greatly expedites individual-centric analysis, it is currently incapable of searching for variants in specific chromosomal regions, unless the results of GQT queries are filtered to specific regions by variant-centric tools such as Tabix[10] and BCFTOOLS. However, we have shown that it is a natural complement to variant-centric indexing strategies that struggle with individual-centric queries. Considering the strengths and trade-offs of each approach, we envision a general querying interface that integrates both indexing strategies and supports genomic data–sharing efforts such as the Global Alliance for Genomics and Health. Given the efficiency and inherent flexibility of this genotype-indexing strategy, we expect GQT to be a potent analysis tool and to enhance existing methods for the exploration of massive data sets involving millions of genomes[2].

METHODS

Methods and any associated references are available in the online version of the paper.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

We are grateful to C. Chiang for conceptual discussions, I. Levicki for helpful advice on AVX2 operations, S. McCarthy and P. Danecek for their guidance with htslib, and Z. Kronenberg for advice on population genetics measures. We also thank the Exome Aggregation Consortium and the groups that provided exome variant data for comparison. A full list of contributing groups can be found at http://exac.broadinstitute.org/about. This research was supported by a US National Human Genome Research Institute award to A.R.Q. (NIH R01HG006693).

AUTHOR CONTRIBUTIONS

R.M.L. designed and wrote GQT and analyzed the data. N.K. wrote a fast output method. K.J.K. analyzed ExAC data. A.R.Q. conceived and designed the study. R.M.L. and A.R.Q. wrote the manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Zuk, O. *et al. Proc. Natl. Acad. Sci. USA* **111**, E455–E464 (2014).
2. Stephens, Z.D. *et al. PLoS Biol.* **13**, e1002195 (2015).
3. Danecek, P. *et al. Bioinformatics* **27**, 2156–2158 (2011).
4. Keinan, A. & Clark, A.G. *Science* **336**, 740–743 (2012).
5. 1000 Genomes Project Consortium. *et al. Nature* **491**, 56–65 (2012).
6. Wu, K., Otoo, E.J. & Shoshani, A. In *Proc. 14th International Conference on Scientific and Statistical Database Management* (Ed. Kennedy, J.) 99–108 (IEEE, 2002).
7. Siqueira, T.L.L., Ciferri, C.D.deA., Times, V.C., de Oliveira, A.G. & Ciferri, R.R. *J. Braz. Comput. Soc.* **15**, 19–34 (2009).
8. Liu, Y.-B. *et al. J. Korean Astron. Soc.* **47**, 115–122 (2014).
9. Quinlan, A.R. & Hall, I.M. *Bioinformatics* **26**, 841–842 (2010).
10. Li, H. *Bioinformatics* **27**, 718–719 (2011).
11. Ziv, J. & Lempel, A. *IEEE Trans. Inf. Theory* **23**, 337–343 (1977).
12. Weir, B.S. & Cockerham, C.C. *Evolution* **38**, 1358 (1984).

ONLINE METHODS

Overview of the GQT genotype-indexing strategy. Once the genotypes in a VCF file have been transposed from a variant-centric form (**Fig. 1b**) to an individual-centric form (**Fig. 1c**), the variant columns are sorted by allele frequency (**Fig. 1d**). Each individual's genotypes are then encoded into a bitmap index comprising four distinct bit arrays corresponding to each of the four (including 'unknown') possible diploid genotypes (**Supplementary Fig. 2**). Bitmaps allow for efficient comparisons of many genotypes in a single operation by means of bitwise logical operations, thereby enabling rapid comparisons of sample genotypes among many variants in the original VCF file (**Supplementary Fig. 3**). Lastly, the bitmap indices are compressed using WAH encoding[6], which achieves near-optimal compression while still allowing bitwise operations directly on the compressed data (**Supplementary Fig. 4**). This encoding minimizes the disk-storage requirements of the index and, by eliminating the need for data inflation, improves query speed, as runtime is a function of the compressed input size.

Sorting by allele frequency to improve compression. Long runs of identical genotypes are easily compressed. We have chosen an alternative individual-centric data-organization strategy that, although it facilitates queries based on individual genotypes, destroys the inherent compression of the genotype runs in the variant-centric approach. This loss of compression is the direct consequence of the fact that records in the individual-centric approach reflect the genotypes for a given individual at each site of variation in the genome. Runs of identical genotypes are far shorter on average than those obtained with the variant-centric approach, and therefore the individual-centric strategy yields poor compression. The question then becomes how to leverage the query efficiency of individual-centric data organization while also retaining the opportunity for data compression. GQT solves this problem by sorting the variant columns of the transposed matrix in order of allele frequency. This results in fewer, longer runs of identical genotypes in each individual's row of genotypes (**Supplementary Fig. 1**). For example, we compared the effect of sorting variants on genotype runs using chromosome 20 from phase 3 of the 1000 Genomes Project. As expected, sorting variants by allele frequency caused both a dramatic increase in the mean length (10.7 versus 23.2) of identical genotype runs and a decrease in the median number of runs per individual (158,993.5 versus 70,718.5). Fewer, longer identical genotype runs allow for greater compression of each individual's (reordered) variant genotypes.

Representing sample genotypes with bitmap indices. The fundamental advantage of individual-centric data organization is the fact that all of an individual's genotypes can be accessed at once. This enables algorithms to quickly compare all variant genotypes from multiple samples in search of variants that meet specific inheritance patterns, allele frequencies or enrichments among subsets of individuals. Despite the improved data alignment, comparing sample genotypes can still require substantial computation. For VCF, which encodes diploid genotypes as 0/0 for homozygotes of the reference allele, 0/1 for heterozygotes, 1/1 for homozygotes of the alternate allele and ./. for unknown genotypes (**Supplementary Fig. 2a**), comparing the genotypes of two or more individuals requires iterative tests of each genotype for each individual.

Recognizing this inefficiency, we encode each individual's vector of genotypes with a bitmap index. A bitmap index (bitmap) is an efficient strategy for indexing attributes with discrete values that uses a separate bit array for each possible attribute value. In the case of an individual's genotypes, a bitmap comprises four distinct bit arrays corresponding to each of the four (including 'unknown') possible diploid genotypes. The bits in each bit array are set to true (1) if the individual's genotype at a given variant matches the genotype the array encodes (**Supplementary Fig. 2a**). Otherwise, the element is set to false (0). In turn, bitmap encoding facilitates the rapid comparison of individuals' genotypes with highly optimized bitwise logical operations. As an example, a bitmap search for variants where all individuals are heterozygous involves a series of pairwise AND operations on the entire heterozygote bit array from each individual. The intermediate result of each pairwise AND operation is subsequently compared with that for the next individual, until the final bit array reflects the variants where all individuals are heterozygous (**Supplementary Fig. 2b**). Such queries are expedited owing to the ability of modern CPUs to simultaneously test multiple bits (i.e., genotypes) with a single bitwise logical operation.

Efficient comparisons using bitmaps. By using bitwise logical operations, we can compare many genotypes in a single operation, instead of comparing each individual value. At a low level, bitwise logical operations are performed on words, which are the fixed-size unit of bits used by the CPU. Modern processors typically use either 32- or 64-bit words. When a bitwise logical operation is performed between two bit arrays (each of which corresponds to the genotypes of two individuals), the processor completes this operation on one pair of words at a time. For example, if the word size is 32, then computing the bitwise AND of two bit arrays that are each 320 bits long would require only ten ANDs. This optimization is equivalent to a 32-way parallel operation with zero overhead. Consider a search for the loci at which three individuals are heterozygous among eight variants (**Supplementary Fig. 3**). When genotypes are encoded in ASCII (as they are in VCF), the algorithm must loop over every individual and every variant to find the common sites. In total, this requires 24 iterations (**Supplementary Fig. 3a**). In contrast, encoding genotypes with a bitmap allows the same computation to be completed with only three bitwise AND operations (**Supplementary Fig. 3b**). In effect, bitwise logical operations compare all eight genotypes in parallel in a single step. For brevity, an 8-bit word is used, and only the heterozygous bit arrays are shown, but the same principles hold for the larger word sizes used by GQT.

Using WAH to directly query compressed data. Although bitwise logical operations can drastically improve query runtime performance, bitmap indices require double the amount of space over the minimum 2 bits per genotype. To address this issue, we can look toward compressing that data. Although genotype data can be compressed with standard run-length encoding (RLE), bitwise logical operations require that the bits associated with variants be aligned, which is difficult to ensure with RLE. Instead we used the WAH encoding strategy, which represents run length in words rather than bits. RLE encodes stretches of identical values ('runs') as a new value in which the first bit indicates the run value and the remaining bits give the number of bits in the run

doi:10.1038/nmeth.3654

(**Supplementary Fig. 4a**). WAH is similar to RLE, except that it uses two different types of values. The 'fill' type encodes runs of identical values, and the 'literal' type encodes uncompressed binary (**Supplementary Fig. 4a**). This hybrid approach addresses an inefficiency in RLE in which short runs map to larger encoded values. The first bit in a WAH value indicates whether it is fill (1) or literal (0). For a fill value, the second bit gives the run value and the remaining bits give the run length in words (not bits, like in RLE). For a literal value, the remaining bits directly encode the uncompressed input. As each WAH-encoded value represents some number of words, and as bitwise logical operations are performed between words, these operations can be performed directly on compressed values.

The algorithm that performs bitwise logical operations is straightforward (**Supplementary Fig. 4b**). To operate on two uncompressed bit arrays, one simply moves in unison between the two arrays from the first to the last word and finds the result for each pair of words. Because each WAH value encodes one or more words, one must move across each WAH array independently. At each step the number of words that have been considered in the current value is tracked, and the next word is considered only once the words in the current value have been exhausted. Bitwise logical operations improve the performance of most queries by computing many genotype comparisons in parallel, but some higher-level queries cannot be resolved with these operations alone.

Finding the allele frequency among a set of individuals is one such query. In this case, each bit corresponds to the genotype of a particular individual at a particular genomic position, and the allele frequency for that position is the summation of the corresponding bits across all individuals. As 32 bits are packed into a single word, this process can be reduced to a series of bitwise sums between words, which, unfortunately, is not a standard operation. Although no architecture provides single-instruction support for bitwise summation, the operation does exhibit a high degree of parallelism. The sum of each position is independent of all other positions, which allows (in principle) the sum of all positions to be found concurrently. This classic Single Instruction Multiple Data (SIMD) scenario can be exploited through the use of the vector processor registers and instructions that are supported by the most recent Intel CPUs (Haswell and beyond). These special registers are designed to perform instructions on a list of values in parallel, and by combining several instructions (logical shift, AND, sum) from the AVX2 instruction set one can obtain the bitwise sum of eight words in parallel. Although the eight-way parallelism lags behind what is possible for other operations, it still represents a meaningful increase in speed for an operation that is expected to be part of many queries.

The index described above has the ability to identify variants that meet a complex set of conditions among millions of individuals and billions of genotypes in seconds.

32-bit WAH word size. A fundamental choice for WAH-encoding bit arrays is the word size. Modern processors support up to 64 bits, but smaller words of 32, 16 and 8 bits are also possible, and the choice affects both the compression ratio and the query runtime. As WAH uses one bit of each word to indicate the type of word (fill or literal), it would seem that larger words would be more efficient. An 8-bit word will have 7 useful bits to every 1 overhead bit, whereas a 64-bit word will have a 63:1 ratio. However, there is a large amount of waste in fill words. Considering that the first 2 bits of a fill word indicate the word type and run value and the remaining bits give the length of the run in words, a 64-bit fill word can encode a run that is 1.4×10^{20} bits long. That is enough bits to encode 46.1 billion human genomes. In fact, we need only 27 bits to cover the full genome, meaning that every 64-bit fill word will have at least 35 wasted bits. This would seem to indicate that smaller words are more efficient, but as the word size decreases, the increase in the speed of the bitwise logical operations also decreases. A single operation between two 64-bit words compares eight times more bits (and their associated genotypes) than an operation between two 8-bit words. Taken together, our tests show that 32 bits gives the best balance between size and speed.

Contents of BCF file, PLINK index and GQT index. Direct compression comparisons must account for the fact that each tool compresses different subsets of the variant and sample genotype sections of a VCF file (**Supplementary Fig. 6**). By default, BCF encodes all of the data and metadata in both sections as binary values and then compresses those values using blocked LZ77 encoding. PLINK ignores both variant and sample genotype metadata, does not compress the variant data and simply encodes each genotype with 2 bits without compression. GQT uses a hybrid strategy for compressing VCF files. It retains all of the variant data and only the genotype values (no metadata) in the genotype section, and it stores those data in a BIM file. The variant data are compressed with LZ77 encoding. In addition, an index of individual genotypes is created by transposition, sorting and WAH compression. Because the variants in the BIM file are stored in the same order as in the original VCF and the genotype index columns are ordered by allele frequency, mapping between these two orderings must be maintained to retain the ability to print results in the same order as in the original VCF. This mapping is stored in a VID file.

Comparison of GQT index sizes to those of other file formats. For file-size comparisons we used an uncompressed VCF file as a baseline; BCFTOOLS used a compressed BCF to store both variant and sample data, PLINK used the binary plink format (BED) to store sample data and a BIM file to store variant data, and GQT used a GQT index file to store WAH-encoded sample genotype data and a BIM file to store LZ77-compressed variant data.

Performance comparisons. We compared GQT v0.0.1 to PLINK v1.90p and BCFTOOLS 1.1 (https://github.com/samtools/bcftools) in terms of index file size and query runtime against four large-scale cohorts and simulated data sets. The cohorts included 2,504 human genomes from the phase 3 1000 Genomes Project, 28 mouse genomes from the Mouse Genomes Project, 205 fly genomes from the *Drosophila* Genetic Reference Panel, and 60,706 human exomes from ExAC. Query comparisons included time to compute the alternate allele frequency for a target 10% of the population and time to find rare (details below) variants among a target 10% of the population. VCF maintains an ordered list of samples, and both target sets comprised the last 10% of individuals in that list. For all runtime comparisons, BCFTOOLS considered a BCF file, PLINK considered a BED and a BIM file, and GQT considered a GQT index and a BIM file (**Supplementary Note 1**).

Runtimes for GQT considered two different modes: the default mode that reports all matching variants in full VCF format, and the 'count' mode (specified by the "-c" option) that reports only the number of matching variants. The count mode is a useful operation in practice, and it also demonstrates speed without input-output overhead.

Alternate allele count. The baseline runtime for finding the alternate allele count was the BCFTOOLS "stats" command with the "-S" option to select the subset of individuals; the PLINK command was "--freq" with the "--keep" option to select individuals, and the GQT command was "query" (with and without the "-c" option) with the "-g "count(HET HOM_ALT)" " option to specify the allele count function and the "-p "BCF_ID >= N" " option to select the subset (where N was the ID of the range that was considered).

Identifying rare variants. The baseline runtime for selecting the variants was the BCFTOOLS "view" command with the "-S" option to select the subset of individuals and the "-C" option to limit the frequency of the variant, and the GQT command was "query" (with and without the "-c" option) with the "-g "count(HET HOM_ALT)<=F" " option to specify the allele count filter (where F was the maximum occurrence of the variant) and the "-p "BCF_ID >= N" " option to select the subset (where N was the ID of the range that was considered). In both cases the limit was set to either 1% of the subset size or 1, whichever was greater. PLINK was omitted from this comparison because third-party tools are required to complete this operation with that software, and in our opinion it is not fair to assign the runtime of those tools to PLINK.

Principal-component analysis. Using the "pca-shared" command, GQT computed a score for each pair of individuals in the target population that reflected the number of nonreference loci shared between the pair. This score was calculated in two stages. First, an intermediate OR operation of the HET and HOM_ALT bitmaps in each individual produced two bitmaps (one for each member of the pair) that marked nonreference loci. Then an AND of these two bitmaps produced a final bitmap that marked the sites where both individuals were nonreference. GQT then counted the number of bits that were set in this bitmap and reported the final score. The "pca-shared" command also takes the target population as a parameter. Here two cases are considered (**Fig. 2e**): all 2,504 individuals, which includes the South Asian, East Asian, Admixed American (AMR), African and European 'super populations', and only the 347 individuals in the AMR super population, which included Peruvians from Lima, Peru, Colombians from Medellin, Colombia, individuals of Mexican ancestry from Los Angeles, California, and Puerto Ricans from Puerto Rico.

This analysis considered only the autosomes in the 1000 Genomes phase 3 variants. The result of the GQT "pca-shared" command is the upper half of a square symmetrical matrix. The Python script "fill_m.py" fills out a full matrix by reflecting those values, and another Python script, "pca_light.py," calculates the Eigen vectors and values using the Numpy scientific computing library and plots the results.

The resulting GQT commands were as follows:

```
gqt pca-shared \
-i ALL.phase3.autosome.vcf.gz.gqt \
-d integrated_call_samples.20130502.ALL.
spop.ped.db \
-p "*" \
-f "Population" \
-l ALL.phase3.autosome.vcf.gz.gqt.pops \
> ALL.phase3.autosome.vcf.gz.gqt.o
gqt pca-shared \
-i ALL.phase3.autosome.vcf.gz.gqt \
-d integrated_call_samples.20130502.ALL.
spop.ped.db \
-p "Super_Population = 'AMR'" \
-f "Population" \
-l ALL.phase3.autosome.vcf.gz.gqt.AMR.pops \
> ALL.phase3.autosome.vcf.gz.gqt.AMR.o
```

Fixation index. F_{ST} is a widely used measurement of the genetic difference between populations, and here we focused on the method proposed by Weir and Cockerham[12]. Although this metric has many parameters, it is fundamentally based on the frequency of an allele and the proportion of individuals that are heterozygous for that allele in each population. GQT can quickly calculate both metrics for various populations across the whole genome. The algorithm calculates allele frequency by considering the bits marked in both HET and HOM_ALT bitmaps. Because each bit in a bitmap corresponds to a specific variant, calculating the allele frequency of all variants involves incrementing the associated counter for each set bit. WAH encoding makes this process more efficient by collapsing large stretches of reference alleles (which are represented by zeros in the HET and HOM_ALT bitmaps) into a small number of words that can be quickly skipped. We further accelerate the processing of each word by using the AVX2 vector-processing instruction set. Vector-processing instructions are a set of special CPU instructions and registers that exploit data-level parallelism by operating on a vector of values with a single operation. These instructions allow us to consider 8 bits in parallel, and therefore compute the resulting sum of each word in 4 (as opposed to 32) operations. The proportion of individuals who are heterozygous for an allele is computed in a similar manner, except that only the HET bitmaps are considered.

This analysis considered the 1000 Genomes phase 3 variants that were biallelic and had a minimum alternate allele frequency of 1%. The GQT "fst" command can consider two or more populations with the "-p" option. Here we considered two cases (**Supplementary Fig. 8**): Utah residents of Northern and Western European ancestry (CEU) versus Han Chinese in Beijing, China (CHB) ("-p "Population = 'CHB' " -p "Population = 'CEU' " "), and CEU versus Yoruba in Ibadan, Nigeria (YRI) ("-p "Population = 'CHB' " -p "Population = 'YRI' " "). In both cases values were smoothed using the mean F_{ST} value over a 10-kb window with a 5-kb step. The comparison to VCFTOOLS considered version 0.1.12 and the "--weir-fst-pop" options for the CHB and CEU populations.

The resulting GQT commands were as follows:

```
gqt fst \
-i  \  ALL.phase3_shapeit2_mvncall_inte-
grated_v5a.20130502.genotypes.vcf.gz.gqt \
-d 1kg.phase3.ped.db \
-p "Population = 'CHB'" \
```


doi:10.1038/nmeth.3654

```
-p "Population = 'CEU'" \
> CHB_vs_CEU.gqt.fst.vcf
gqt fst \
-i  \  ALL.phase3_shapeit2_mvncall_inte-
grated_v5a.20130502.genotypes.vcf.gz.gqt \
-d 1kg.phase3.ped.db \
-p "Population = 'YRI'" \
-p "Population = 'CEU'" \
> YRI_vs_CEU.gqt.fst.vcf
```

Experimental data sets.

- **1000 Genomes phase 3.** Individual chromosome VCF files were retrieved from ref. 2 (last accessed December 10, 2014) and combined into a single file using the BCFTOOLS "concat" command. To understand how each tool scaled as the number of samples and variants increased, we subsampled the full data set (which included 2,504 individuals) to create new sets with 100, 500 and 1,000 individuals. To create each data-set size, we randomly selected the target number of samples and then used the BCFTOOLS "view" command with the "-s" option to return just the genotypes of the target samples. We then recomputed the allele frequency of each variant with the BCFTOOLS "fill-AN-AC" plugin and filtered all nonvariable sites with the BCFTOOLS "view" command and the "-c 1" option.

- **ExAC.** Version 3 of the ExAC data set was analyzed, and runtimes were measured on the computing infrastructure at the Broad Institute.

- **Mouse Genomes Project.** Data were retrieved in VCF format from ftp://ftp-mouse.sanger.ac.uk/current_snps/mgp.v4.snps.dbSNP.vcf.gz, last accessed November 25, 2014.

- ***Drosophila* Genetic Reference Panel.** Data were retrieved in VCF format from http://dgrp2.gnets.ncsu.edu/data/website/dgrp2.vcf, last accessed November 25, 2014.

- **CEPH 1473 pedigree.** A VCF file of variants in the CEPH 1473 pedigree that was sequenced as part of the Illumina Platinum Genomes Project was downloaded from ftp://ftp-trace.ncbi.nih.gov/giab/ftp/data/NA12878/analysis/RTG_small_variants_01132014/cohort-illumina-wgs.vcf.gz.

Simulated data sets. Genotypes were simulated using the MaCS[13] simulator version 0.5d with the mutation rate and recombination rate per site per $4N$ generations set to 0.001 and the region size set to 100 Mb. Because our simulation considered between 100 and 100,000 diploid samples and MaCS only simulates haplotypes, we simulated $2x$ haplotypes for each case and combined two haplotypes to create a single diploid genome. It was computationally prohibitive to produce a data set for 1 million individuals (the 100,000-sample simulation ran for more than 4 weeks), so we used a simple linear fit to estimate the file size and runtimes for 1 million individuals.

Computing environment. GQT is a tool written in C, which uses htslib (https://github.com/samtools/htslib) to interact with VCF and BCF files and zlib (http://www.zlib.net/) to compress and inflate variant metadata. All experiments were run on Ubuntu Linux v3.13.0-43, with gcc v4.9.2, 4 Intel Core i7-4790K 4.00 GHz CPUs with the Haswell microarchitecture, and a 550 MB/s read-write solid-state hard drive.

Code availability. All source code for the GQT toolkit is available at https://github.com/ryanlayer/gqt. Furthermore, all commands used for the experiments conducted in this study are available at https://github.com/ryanlayer/gqt_paper.

13. Chen, G.K., Marjoram, P. & Wall, J.D. *Genome Res.* **19**, 136–142 (2009).