SUPPLEMENTARY NOTE

GQT index scalability. GQT creates two additional indices (BIM and VID) that allow variants satisfying an analysis to be quickly returned in VCF format (**Supplementary Fig. 6**). The storage cost of the GQT indices is marginal with respect to the size of the underlying VCF file and its size continues to diminish as the cohort size grows since most new variation discovered will be very rare. To demonstrate this effect, we compare the GQT index and PLINK (cite) BED encoding of the Phase 3 of the 1000 Genomes project (2,504 individuals and 84,739,846 variants) to the size of the uncompressed VCF for the full dataset (**Fig. 2a**). We also include the size of a BCF compressed version of the 1000 Genomes VCF as a point of comparison for the size of the GQT index.

Both the BCF and PLINK encodings exhibited constant compression across population sizes with a 9.6× improvement (138.4 GB) and 24.1× reduction (55.1 GB), respectively, for the full 2,504 individuals. In contrast, the reduction in the relative size of the GQT indices steadily improved as the number of individuals and variants increased, yielding a 92.7× (14.3 GB) reduction for 2,504 individuals and requiring, on average, only 0.54 bits per genotype. The BCF encoding of the 1000 Genomes dataset included extensive metadata such as genotype likelihoods and allelic read depths for each individual and variant. It is encouraging that when we removed this metadata (BCF* in **Supplementary Fig. 7a**) in order to provide a direct comparison to the storage requirements for all of the GQT indices, the compression rates were nearly identical. This demonstrates that for genotypes from large human cohorts, the GQT compression strategy is on par with the LZ77 algorithm and it is not sacrificing additional compression.

Query performance. The typical tradeoff for high data compression is the time spent decompressing compressed data prior to analysis[1]. We designed our 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 BCFTOOLS and PLINK. First, we compared the time required to compute the alternate allele frequency among a target set of 10% of individuals from the 1000 Genomes VCF (**Fig. 2a**). Whereas BCFTOOLS required 1517.5 seconds, both GQT and PLINK were substantially faster, requiring 58.4 seconds (26.0 fold speedup) and 156.3 seconds (9.6 fold speedup). Importantly, GQT's performance improved as the number of individuals increased, whereas PLINK's performance was relatively flat. Moreover, matching variants are identified almost instantly and thus, the majority of GQT's runtime is spent emitting the VCF results of the query. For example, when the GQT "count" option is invoked to simply return the count of variants matching the query (i.e., without returning the full variant records themselves), the runtime dropped to 4.2 seconds. We also compared the time required to identify rare (AAF < 1%) variants among a subset of 10% of the individuals (**Fig. 2b**). In this case, GQT was up to 45.8× faster than BCFTOOLS (51.5 seconds v. 2360.5 seconds). PLINK was not included in the rare variant search comparison because it does not support that function.

GQT's query performance relative to existing methods continues to improve for even larger datasets. When considering the Exome Aggregation Consortium (ExAC) variant dataset (9.36 million exonic among 60,706 human exomes), the GQT index was only 0.2% the size of the VCF (28GB v. 14.1 TB), reflecting a storage requirement of merely 0.38 bits per genotype. Moreover, rare variants were found in only 2.1 minutes (9.98 seconds when excluding the time required to report the variants), reflecting a 443.5 fold improvement over BCFTOOLS (931.4 minutes). Furthermore, based on simulated datasets involving 100 to 100,000 genomes (Methods), it is clear that GQT's data compression and query performance continues to improve dramatically with larger cohorts. While simulating variants from one million or more individuals is computationally intractable for this study, extrapolation suggests that GQT indices involving a cohort of a million genomes will yield query performance that is at least 218 fold faster than BCFTOOLS (**Fig. 2c,d**).

Pedigree analyses. Using the GQT query interface, we screened for high confidence de novo mutations in the CEPH 1473 pedigree sequenced as part of the Illumina Platinum Genomes project (**Supplementary Fig. 5a**). To accomplish this, we employ a combination of genotype and phenotype query pairs. For example, "`-p "sample_id = 'NA12878'" -g "HET"`" will select only those variants where NA12878 is heterozygous. Any subsequent genotype/phenotype query pair will further restrict the final set of variants to the subset meeting the extra conditions. For example, the query "`-p "sample_id = 'NA12878'" -g "HET" -p "maternal_id = 'NA12878'" -g "pct(HET HOMO_ALT)>=0.3"`" includes two pairs and will return only those variants where NA12878 is heterozygous and where at least 30% of her children are also non-reference.

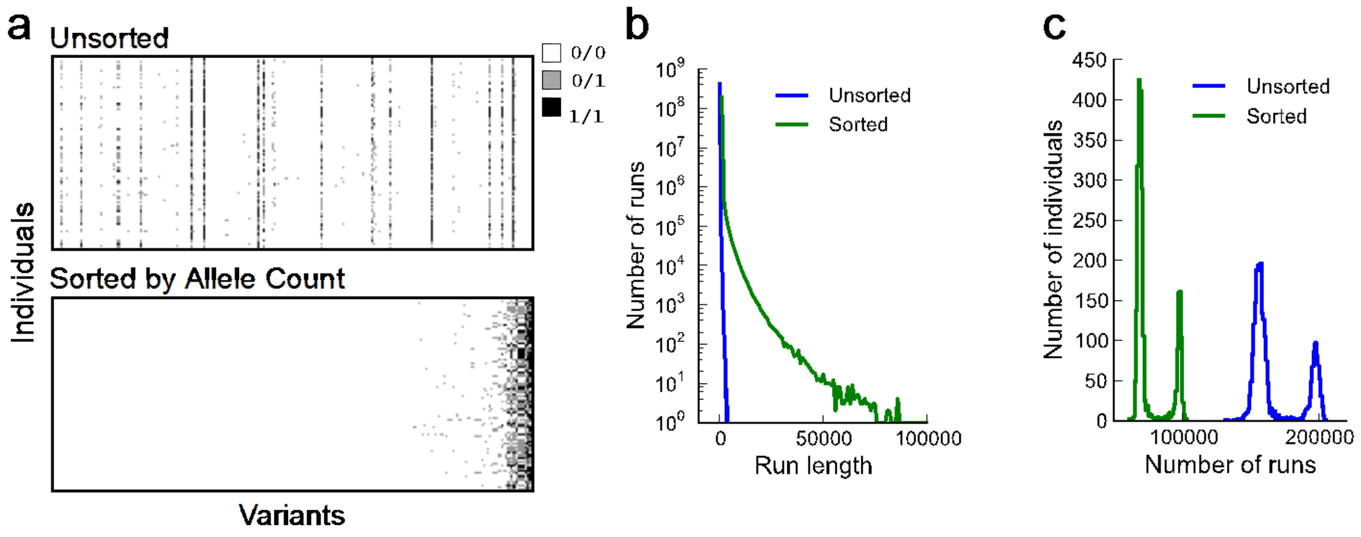
A preliminary search for candidate de novo mutations in the extensively studied NA12878 daughter involves screening for variants that are homozygous for the reference allele in NA12878's parents, yet are heterozygous in NA12878. GQT identifies 11,172 such candidates from more than 8 million total variants in 0.04 seconds (**Supplementary Fig. 5b**). A more sophisticated GQT query recognizes that true de novo mutations in the germline of NA12878 should be inherited by her offspring, reducing the set of candidates to 3,002 (**Supplementary Fig. 5c**). Excluding suspicious variants that lie in low-complexity regions[2] reduces the set of de novo mutation candidates by another 12% (N = 2,659). Since GQT reports results in valid VCF format, we were able to use other existing software to filter the results. In particular, we used BEDTOOLS to filter out any variants fell within a low complexity region as identified by Li[2] (**Supplementary Fig. 5d**). While this yields many more candidates than would be predicted by the 1.2×10^{-8} per generation base pair mutation rate observed in the CEU pedigree[3], the 1000 Genomes Project employed additional filters based on genotype likelihoods, proximity other variants, and other properties of the sequence alignments[3]. Moreover, the intent of this analysis is to demonstrate GQT's analytical power in the context of both large studies of unrelated individuals as well as family-based studies of disease.

The resulting GQT command was:

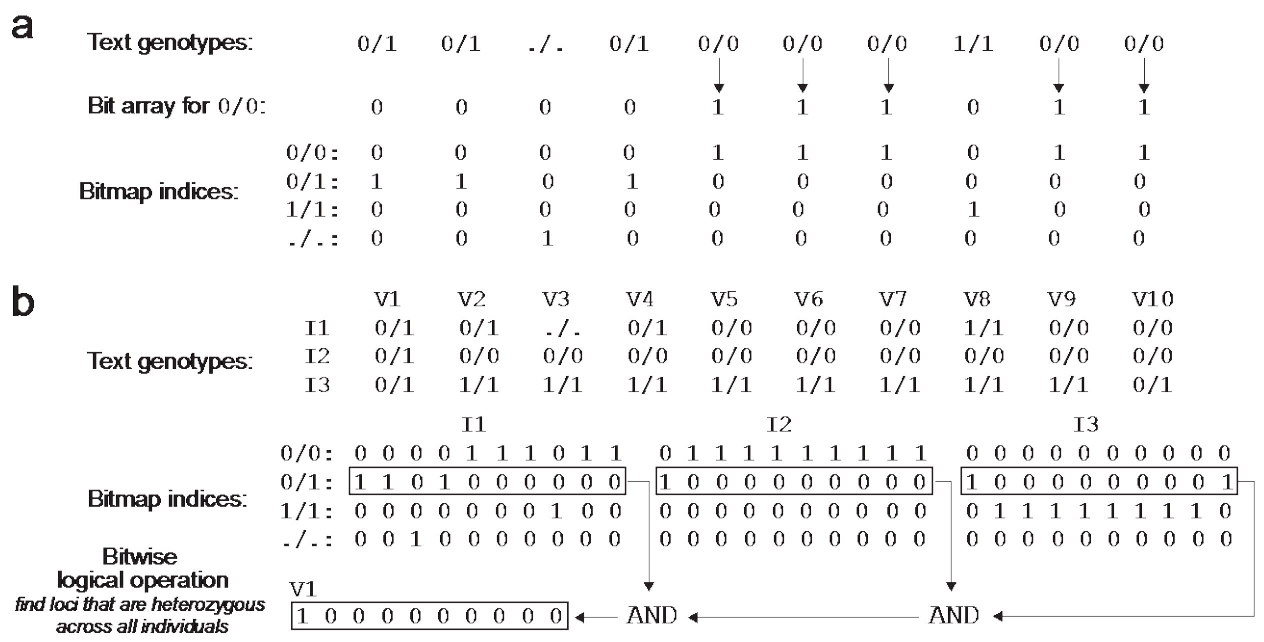
```
gqt query -i cohort-illumina-wgs.vcf.gz.gqt -d ceph1463.ped.db \
    -p "sample_id in ('NA12891','NA12892','NA12877')" \
    -g "HOMO_REF" \
    -p "sample_id = 'NA12878'" \
    -g "HET" \
    -p "maternal_id = 'NA12878'" \
    -g "pct(HET HOMO_ALT)>=0.3" \
| bedtools intersect -v \
    -a stdin \
    -b btu356_LCR-hs37d5.bed/btu356_LCR-hs37d5.bed \
> NA12878_de_novo.vcf
```

Burden tests. Although the GQT index is structured based on individuals rather than variant loci, it is nonetheless capable of efficiently computing the fundamental measures that underlie many locus-centric statistics such as burden tests. For example, the C-alpha burden test[4] measures the difference in the number of times an allele is observed in cases and controls versus a binomial expectation of the allele counts in the population as a whole. We have implemented the C-alpha statistic in GQT; once case and control populations are defined via the GQT query interface, a standard summary statistic (in this case minor allele frequency) is computed for both subpopulations at each variant site. The statistics for each variant are then aggregated across a given locus and normalized by the variance to produce the final statistic for each gene or locus. We emphasize that C-alpha is merely a representative example of the several burden tests that follow this computational pattern and are therefore candidates for future integration into the GQT toolset.

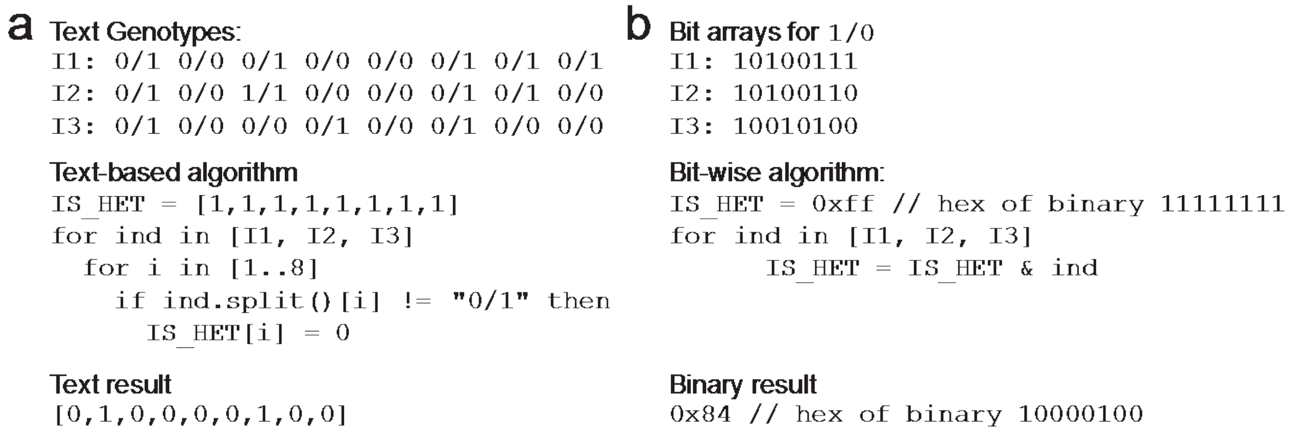
SUPPLEMENTARY FIGURES AND TABLES



Supplementary Figure 1. Sorting variants by allele frequency improves compression. (**a**) A graphical comparison of the genotype distribution of individuals (rows) and variants (columns) before and after sorting. These data represent genotypes from the 1000 Genomes Project, phase 3, for a portion chromosome 20. (**b**) The run length distribution for unsorted and sorted genotypes. (**c**) The distribution of the number of runs for sorted and unsorted data. In both cases the second peak is composed predominantly of individuals from African decent (AFR); 604/661 AFR are in the second peak in the sorted case, and 640/661 AFR in the unsorted case.



Supplementary Figure 2. Bitmaps enable rapid genotype comparisons en masse. (**a**) A bit array marks the existence of one genotype state (for example, homozygous reference; “0/0”) for all variants. Similarly, a bitmap index is composed of a distinct bit array for each possible genotype state. (**b**) Example genotypes in VCF format are presented for three individuals (I1, I2, I3) at 10 variant sites (V1-V10). A bitwise AND of the bit arrays corresponding to the heterozygous genotype yields the variant that is heterozygous in all individuals.



a Text Genotypes:

```
I1: 0/1 0/0 0/1 0/0 0/0 0/1 0/1 0/1
I2: 0/1 0/0 1/1 0/0 0/0 0/1 0/1 0/0
I3: 0/1 0/0 0/0 0/1 0/0 0/1 0/0 0/0
```

Text-based algorithm

```
IS_HET = [1,1,1,1,1,1,1,1]
for ind in [I1, I2, I3]
  for i in [1..8]
    if ind.split()[i] != "0/1" then
      IS_HET[i] = 0
```

Text result

```
[0,1,0,0,0,0,1,0,0]
```

b Bit arrays for `1/0`

```
I1: 10100111
I2: 10100110
I3: 10010100
```

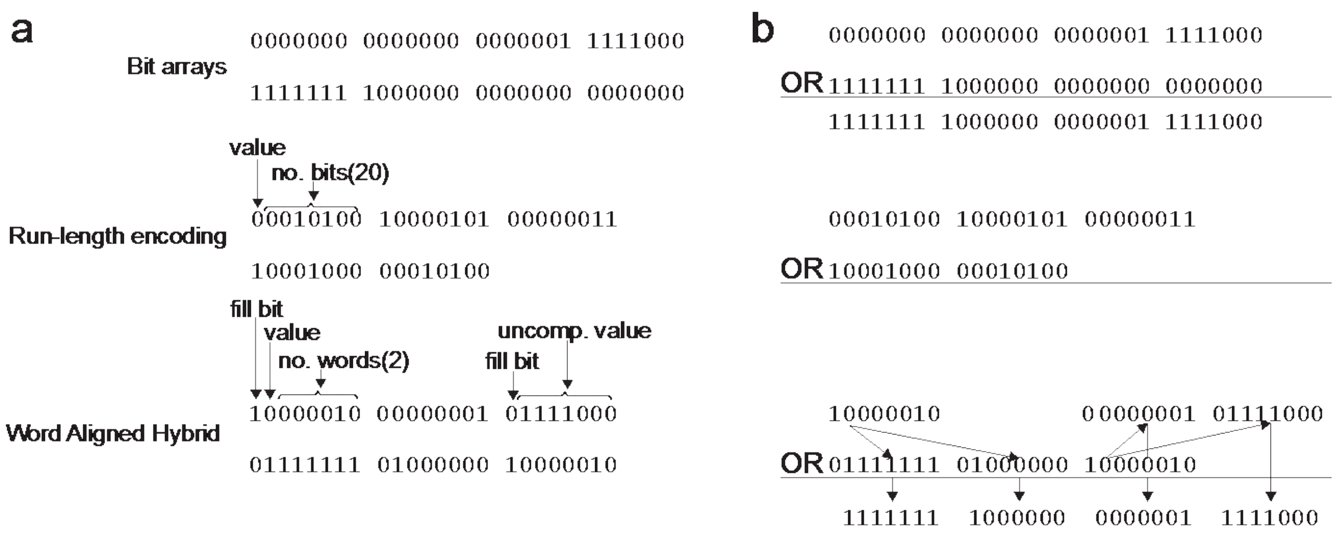
Bit-wise algorithm:

```
IS_HET = 0xff // hex of binary 11111111
for ind in [I1, I2, I3]
      IS_HET = IS_HET & ind
```

Binary result

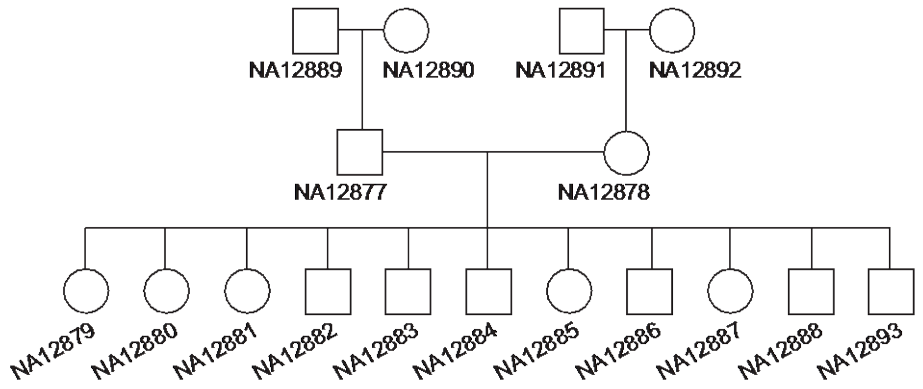
```
0x84 // hex of binary 10000100
```

Supplementary Figure 3. Efficiency improvements for genotype comparisons when using a bitmap index. (**a**) When considering genotypes in ASCII format (e.g., VCF), an algorithm searching for the set of variants that are heterozygous in all individuals must operate on every genotype for each for every individual separately. (**b**) In contrast, when genotypes are represented with a bitmap index, where a set of genotypes are encoded into a single CPU word (for brevity, only the bit arrays associated with the heterozygous state are shown), bitwise logical operations can be used to operate on all of the genotypes in the word with a single operation. This example assumes a word size of 8, but modern CPU support up to a 64-bit word and the speedup of bitmaps will be linear to the word size. For the 24 genotypes given here (3 individuals, 8 genotypes each), the ASCII-base algorithm executes the "if" statement 24 times, while the bit-wise algorithm executes the logical AND ("&") only three times, with both algorithms producing equivalent results.



Supplementary Figure 4. A comparison of binary encodings and associated bit-wise logical operations. (**a**) Two bit arrays are given in three different binary encodings: the uncompressed bit array (in 7 bit words) on top, followed by the run-length encoding (RLE) and word-aligned hybrid encoding (WAH), which each add an additional bit as part of the encoding to make 8 bit words. RLE maps each set of consecutive bits to a new value that uses one bit for the run value, and the remaining bits for the number of bits in the run. WAH maps bits to one of two types of values: those that include runs and those that encode the raw binary. The first bit in a WAH value indicates if the value encodes a run (the fill bit). If that bit is set then the second bit gives the run value and the remaining bits give the run length in number of ***words*** (i.e., not number of ***bits*** as in RLE). If the fill bit is not set then the remaining bits are the uncompressed binary values. (**b**) The logical OR for the three encodings is given. For the uncompressed binary, the OR follows bit for bit across both values. For RLE, the logical OR is undefined (without inflation) because the two encoded values are no longer aligned and have different lengths. For WAH, the values are aligned based on their run length, then the logical OR is performed (in this case) between the value bit and the associated uncompressed values.

a



b

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
```

↓

11,172 candidates (0.04 seconds)

c

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892','NA12877')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
  -p "maternal_id = 'NA12878'" -g "pct(HET)>=0.3"
```

↓

3,002 candidates (0.19 seconds)

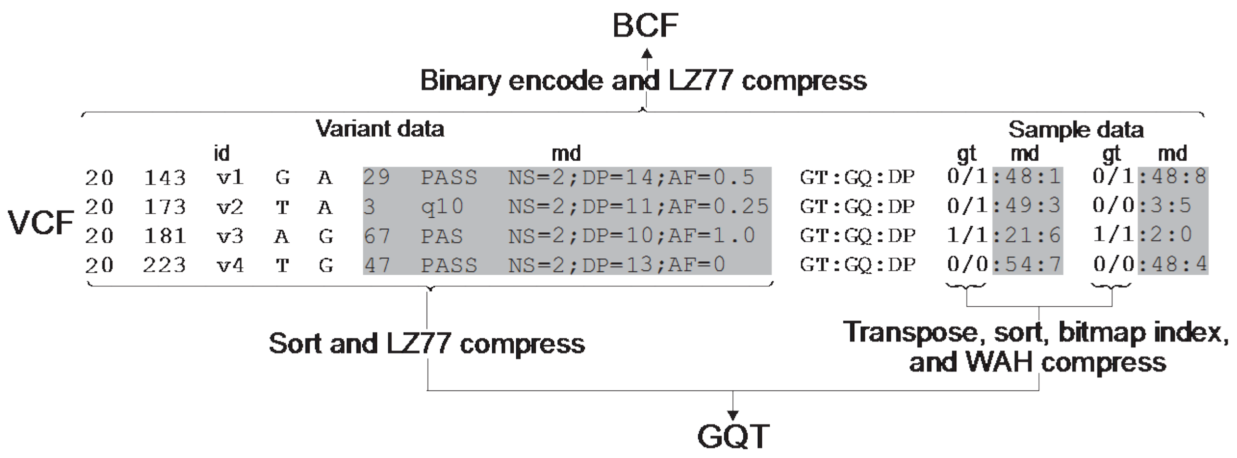
d

```
gqt query -i ceph1463.gqt -d ceph1463.ped.db
  -p "sample_id in ('NA12891','NA12892','NA12877')" -g "HOM_REF"
  -p "sample_id = 'NA12878'" -g "HET"
  -p "maternal_id = 'NA12878'" -g "pct(HET)>=0.3"
| bedtools intersect -a - -b LCR-hs37d5.bed.gz -v -sorted
```

↓

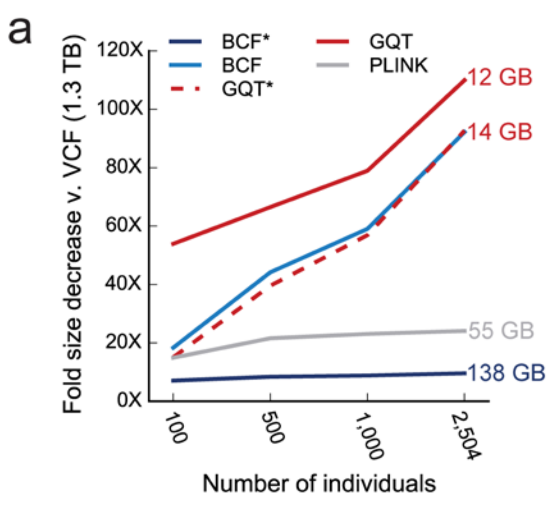
2,659 candidates (0.82 seconds)

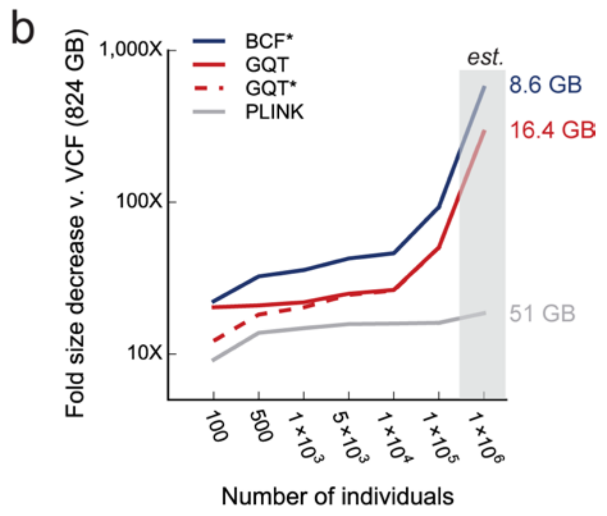
Supplementary Figure 5. De novo mutation discovery in the CEPH 1463 pedigree. (**a**) The CEPH 1463 pedigree. Our analysis is focused on the discovery of de novo mutations in NA12878, the daughter of NA12891 and NA12892. (**b**) A GQT query for de novo mutations based on the expected genotypes (homozygous for the reference allele) in NA12878's parents, as well as an expected heterozygous genotype in NA12878. (**c)** True de novo mutations in NA12878's germline should be passed on to 50% of her offspring, on average. Allowing for genotyping error and binomial expectation, we filter for more confident de novo mutation candidates by requiring that the apparent mutation allele is passed on to at least 30% of NA12878's children. (**d**) A GQT query that further filters candidate mutations by excluding those lying in low complexity regions of the genome.


BCF
Binary encode and LZ77 compress
Variant data
Sample data
id
md
gt
md
gt
md
VCF
Sort and LZ77 compress
Transpose, sort, bitmap index, and WAH compress
GQT

```
20  143  v1  G  A  29  PASS  NS=2;DP=14;AF=0.5   GT:GQ:DP  0/1:48:1  0/1:48:8
20  173  v2  T  A  3   q10   NS=2;DP=11;AF=0.25  GT:GQ:DP  0/1:49:3  0/0:3:5
20  181  v3  A  G  67  PAS   NS=2;DP=10;AF=1.0   GT:GQ:DP  1/1:21:6  1/1:2:0
20  223  v4  T  G  47  PASS  NS=2;DP=13;AF=0     GT:GQ:DP  0/0:54:7  0/0:48:4
```

Supplementary Figure 6. BCF and GQT file composition. VCF files are composed of a variant data section and a sample genotype section and each of these include both core information (e.g. variant position and alleles, and genotype, respectively) and extra metadata. BCF encodes both sections into a binary format, and then compresses the data using blocked LZ77 compression. GQT uses three files: a BIM file that contains all variant data compressed with LZ77, a GQT file with WAH-compressed genotypes (without metadata), and a VID file that stores the mapping between the allele-frequency variant order and the original source VCF variant order. PLINK (not shown) omits all metadata, storing uncompressed binary encoded for genotypes in a BED file and variant positions in a BIM file.





Supplementary Figure 7. GQT compression performance for 1000 Genomes data (Phase 3) and simulated genomes. Compression ratios describe the fold reduction in file size relative to an uncompressed VCF file. GQT represents solely total size of the GQT genotype index. GQT* reflects the total size of the GQT genotype index plus the size of the BIM and VID indices. BCF* represents the size of the full BCF file including genotype metadata, whereas BCF omits genotype metadata. Lastly, PLINK reflects the size of PLINK v1.9 BED and BIM files. (**a**) File size reduction for 1000 Genomes Phase 3, which is comprised of 2,504 individuals and over 84 million variants. (**b**) A comparison of BCFTOOLS, GQT, and PLINK for simulated genotypes on a 100 Mb genome with between 100 and 100,000 individuals.

SUPPLEMENTARY TABLE

Experiment	**Tool**	**MGP** *(28 samples; 68.1 million variants)*	**DGRP** *(205 samples; 6.1 million variants)*	**1KGp3** *(2,504 samples; 84.7 million variants)*	**ExAC** *(60,706 samples; 9.3 million variants)*
File size (Gb)	BCFTOOLS	14.94	2.99	138.41	2119.7
	PLINK	**2.37**	0.47	55.10	142.29
	GQT	2.97	**0.43**	**14.33**	**27.96**
Alternate allele count time (min.)	BCFTOOLS	2.79	0.43	14.30	544.30
	PLINK	**0.41**	0.05	2.61	20.10
	GQT	1.07	**0.04**	**0.97**	**1.50**
Rare variant search time (min.)	BCFTOOLS	4.44	0.64	39.34	931.40
	PLINK	NA	NA	NA	NA
	GQT	**0.75**	**0.03**	**0.86**	**2.10**

Supplementary Table 1. The performance of BCFTOOLS, PLINK, and GQT across four cohorts of different sizes and sample populations. This analysis included three whole-genome data sets from three different species: mouse genome project (MGP), *Drosophila* genome reference panel (DGRP), human from 1000 Genomes phase 3 (1KGp3), and a whole-exome for human from the Exome Aggregation Consortium (ExAC).

REFERENCES

1. Ziv, J. & Lempel, A. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theory* **23,** 337–343 (1977).
2. Li, H. Toward better understanding of artifacts in variant calling from high-coverage samples. *Bioinformatics* **30,** 2843–2851 (2014).
3. 1000 Genomes Project Consortium *et al.* An integrated map of genetic variation from 1,092 human genomes. *Nature* **491,** 56–65 (2012).
4. Neale, B. M. *et al.* Testing for an Unusual Distribution of Rare Variants. *PLoS Genet.* **7,** e1001322 (2011).