Supplementary Figure 1. Regional assembly

Graph A represents the initial assembly of a region, which is complicated by spurious edges resulting from sequencing errors. The two red edges represent sequences where the sample and reference genomes are predicted to be identical (called stones in the text). The problem is to assemble the sequence between the two red edges. To do this, we first ‘prune off’ any edge (other than the two red edges) that ‘ends’, i.e. involves a source or a sink of the graph. Thus a, b, and c are pruned off. The process is iterated, so d is pruned off too. Then we let the reads ‘vote’ between any residual branches. Thus if 10 reads support edge e, but only 1 read supports edge f, we can remove edge f. Graph B is the result: it is linear, and thus unambiguously describes the sequence of the region.
Supplementary Table 1. Bacterial data sets used to test polymorphism discovery with VAAL

| sample organism           | reference sequence (Genbank accession) | validated differences, sample vs ref | genome size (Mb) | GC content (%) | total reads (M) | Q20 coverage (X) |
|---------------------------|----------------------------------------|-------------------------------------|------------------|----------------|----------------|-----------------|
| *Staphylococcus aureus* USA300 | CP000730.1 + AF167161.1                 | 2                                   | 2.90             | 32.7           | 6.7            | 52.9            |
| *Escherichia coli* K12 MG1655 | U00096.2                                 | 3                                   | 4.64             | 50.8           | 9.5            | 32.9            |
| *Mycobacterium tuberculosis* F11 | NC_009565.1                             | 3                                   | 4.42             | 65.6           | 9.0            | 18.5            |

Data about the samples (organism, closest reference sequence, genome size, and genome GC content) are shown, together with data about the sequence that was generated from them. Each line describes a lane of 36 bp Illumina reads. Validated differences: number of observed differences between the sample and the reference sequence, all of which were validated using an alternative sequencing technology (Supplementary Methods). These differences represent either base changes that have occurred in culture since the reference sequence was generated or actual errors in the reference sequence, and were incorporated into the reference sequences used in this work. Total reads: total number of reads, same as number of detected clusters. All reads were included, regardless of quality or alignability. Read length: 36 bases. For *M. tuberculosis*, the original read length was 47 bases, but the reads were trimmed to 36 bases. Q20 coverage: number of Q20 bases in reads aligning to the reference, divided by the genome size. Quality scores were inferred from "Bustard" *.prb.txt files in the generated by Illumina software.
Supplementary Table 2. Nature of called, uncalled polymorphisms of *M. tuberculosis* F11 vs H37Rv

| polymorphism type | called | indel size (bp) | number of events | Q20 coverage (X) | GC content (%) |
|-------------------|--------|-----------------|------------------|------------------|----------------|
| SNP               | +      | 736             | 18.5             | 66               |
| indel             | +      | 4-10            | 6                | 19               | 63             |
| indel             | +      | 11-30           | 4                | 31.5             | 55             |
| insertion         | +      | 37              | 1                | 30               | 54             |
| deletion          | +      | 1100            | 1                | 10               | 76             |
| composite         | +      | 18              | 15.5             | 68               |
| SNP               | -      | 21              | 5                | 80               |
| indel             | -      | 1-3             | 3                | 7                | 78             |
| indel             | -      | 4-10            | 2                | 9.5              | 74             |
| indel             | -      | 11-30           | 3                | 6                | 86             |
| deletion          | -      | 60              | 1                | 28               | 60             |
| insertion         | -      | 68              | 1                | 8                | 66             |
| insertion         | -      | 78              | 1                | 3                | 86             |
| composite         | -      | 1               | 4                | 84               |

Information about called and uncalled polymorphisms is given. For indels, the size is shown. To compute the Q20 coverage, we aligned the reads to the F11 reference, then reported coverage depending on polymorphism type: (a) for SNPs, the number of Q20 bases covering the variant position; (b) for insertions, the mean number of Q20 bases covering the inserted positions; (c) for deletions, the mean of the Q20 coverages of the two flanking bases; (d) for composites, the mean Q20 coverage across the feature. The GC content was taken over a 50-base window on the sample genome, centered at the midpoint of sequence A (step 4).
Supplementary Table 3. Effect of depth of sequence coverage on discovery of polymorphisms in *M. tuberculosis* F11 vs H37Rv

| Total reads (M) | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Q20 coverage (X) | 2.1 | 4.1 | 6.2 | 8.2 | 10.3| 12.3| 14.4| 16.4| 18.5|
| Polymorphisms called | 54  | 424 | 630 | 723 | 763 | 784 | 799 | 808 | 808 |
| Fraction of callable | 6%  | 50% | 75% | 86% | 91% | 93% | 95% | 96% | 96% |
| False positives | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Random samplings of given numbers of *M. tuberculosis* F11 reads (Supplementary Table 1) were selected, and polymorphisms were called against strain H37Rv using VAAL. Nine million reads constitutes the entire data set. We show the number of polymorphisms that were called, and also that number as a fraction of the callable polymorphisms (Table 1).
Supplementary Table 4. *Vibrio cholerae* isolates used to find rifampicin-resistance mutations

| isolate           | total reads (M) | Q20 coverage (X) |
|-------------------|-----------------|------------------|
| JA-G-001 (sensitive) | 8.3             | 19.4             |
| JA-G-002 (resistant) | 6.5             | 24.5             |
| P003 (sensitive)  | 7.7             | 7.5              |
| RIF3-1 (resistant) | 8.0             | 9.4              |
| P004 (sensitive)  | 6.9             | 11.9             |
| RIF4-1 (resistant) | 11.3            | 8.8              |
| P005 (sensitive)  | 7.5             | 10.5             |
| RIF5-1 (resistant) | 11.1            | 14.0             |
| P006 (sensitive)  | 9.0             | 4.2              |
| RIF6-1 (resistant) | 9.4             | 22.1             |

The table names ten *Vibrio cholerae* isolates, and provides statistics for the single lanes of unpaired 36 bp Illumina reads obtained from them. (See Supplementary Table 1 for definitions.) The genome size is 4.03 Mb, and the GC content is 47.5%.
Supplementary Tables 5-6. Comparison of three methods for detecting polymorphisms

| sample organism | related strain, for comparison | VAAL | MAQ | Velvet
|----------------|--------------------------------|------|-----|--------|
|                |                               | total events | incorrect events | total events | incorrect events |
| *S. aureus* USA 300 | COL NC_002951.2 | 1337 SNPs | 0 | 1116 (see text) |
| *E. coli* K12 MG1655 | DH10B CP000948.1 | 145 SNPs | 0 | 149 | 0 |
| *M. tuberculosis* F11 | H37Rv AL123456.2 | 776 SNPs | 0 | 825 | 2 |

For each of three data sets (from the first three entries of Supplementary Table 1), we used three methods to detect polymorphisms, in each case showing how many polymorphisms of each type were detected, and how many of those were incorrect. For VAAL, the results recapitulate Table 1. For Velvet, we generated a Velvet assembly, then applied steps 3 and 4 of VAAL. Supplementary Table 5 describes the comparison to MAQ, whereas Supplementary Table 6 describes the comparison to Velvet. The two parts are separated because the reporting is slightly different: MAQ does not report composite events or indels from unpaired reads, so for Supplementary Table 5 we extracted the constituent SNPs from composite events, and ignored the constituent indels.

1 Version 0.6.1. All reads were provided as input without filtering. We used a hash size of 21. We experimented with other parameters so as to improve results. Initially, we required that contigs must be at least 100 bp long and have at least 10X coverage. This defines experiment 1. We found 542 SNPs. Experiment 2: we reduced the coverage requirement to 5X. Many more SNPs (679) were found. Experiment 3: we used the initial coverage requirement, but trimmed reads for adaptor prior to input. Many more SNPs (622) were found than in experiment 1. Experiment 4: we allowed all contigs (length \( \geq \) 0 bp, coverage \( \geq \) 0X), and trimmed for adaptor. Fewer SNPs (589) were found than in either experiment 2 or 3, although more than in experiment 1. Experiment 5: we reduced the coverage requirement to 5X, as in experiment 2, and trimmed for adaptor, as in experiment 3, finding 690 SNPs, which was the largest of any conditions tried. We then applied the parameters of experiment 5 to all three genomes.

2 Version 0.6.6. We ran the command ‘maq easyrun’, taking the file cns.final.snp as output. We excluded entries for which the call’s fasta code was an ambiguous base, and which would thus correspond to a ‘heterozygous’ base (not meaningful for our haploid samples). We also excluded calls for which the reference base was reported as ambiguous. So as to optimize performance, prior to running MAQ, we calibrated the Illumina quality scores (Supplementary Table 1) in the following fashion. We aligned the reads to the reference sequence for the related strain, then for each reported quality score (0, 1, 2,...) we estimated the true error rate of the bases assigned that score by dividing the total mismatched bases assigned that score by the total aligned bases assigned that score. We converted this to a new ‘calibrated’ quality score, replacing the old score.

3 Count includes a single correct but ‘uncallable’ SNP, as discussed earlier.

4 Count excludes a single incorrect composite event, resulting from a region that Velvet assembled correctly but step 4 of VAAL incorrectly reported as a polymorphism.

5 In one of these two cases, an indel was incorrectly reported as a SNP.
Supplementary Table 7. Effect of depth of sequence coverage on discovery of polymorphisms in *M. tuberculosis* F11 vs H37Rv, using VAAL and Velvet + VAAL steps 3,4

| Total reads (M) | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
|----------------|----|----|----|----|----|----|----|----|----|
| Q20 coverage (X) | 2.1 | 4.1 | 6.2 | 8.2 | 10.3 | 12.3 | 14.4 | 16.4 | 18.5 |
| VAAL correct polymorphisms called | 54 | 424 | 630 | 723 | 763 | 784 | 799 | 808 | 808 |
| VAAL fraction of callable | 6% | 50% | 75% | 86% | 91% | 93% | 95% | 96% | 96% |
| VAAL false positives | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Velvet+ correct polymorphisms called | 1 | 68 | 372 | 584 | 673 | 709 | 732 | 748 | 750 |
| Velvet+ fraction of callable | 0% | 8% | 44% | 69% | 80% | 84% | 87% | 89% | 89% |
| Velvet+ false positives | 0 | 6 | 48 | 47 | 37 | 23 | 19 | 12 | 7 |

The table replicates the results of Supplementary Table 3, created using VAAL, and adds to it the result of parallel calculations obtained using Velvet + VAAL steps 3,4 (denoted here "Velvet+"), as described for Supplementary Table 6. We show the number of polymorphisms that were called, and also that number as a fraction of the callable polymorphisms (Table 1).
Supplementary Table 8. *Vibrio cholerae*: observed differences between resistant isolates and their sensitive parents, found using Velvet (+ VAAL steps 3,4)

| Differences with sensitive cultures | JA-G-02 | RIF3-1 | RIF4-1 | RIF5-1 | RIF6-1 |
|-------------------------------------|---------|--------|--------|--------|--------|
| Correct                             | 1       | 1      | 1      | 1      | 0      |
| Incorrect                           | 2       | 5      | 5      | 4      | 12     |

Five rifampicin-resistant isolates were compared to the five sensitive cultures from which they arose, as in Table 2, except that here instead of using VAAL, we use a combination of Velvet and VAAL, as described in the text. We report the presumably correct differences (of which one could be found, as in Table 2), as well as all other differences, which are presumed to be incorrect. For all, Velvet was applied as in Supplementary Table 6.
Supplementary Methods

VAAL ALGORITHM OVERVIEW

Step 1. Position reads on the related reference genome. Whereas most reads can be directly positioned on the genome by alignment, reads arising from insertions in the sample relative to the reference are unique to the sample genome and thus cannot be aligned. To determine positions for these reads, we align reads not only to the genome, but also to reads that have already been aligned to the genome, in an iterative process. We start by aligning the reads to the reference genome. Subsequently, for each read that did not align, we attempt to align to reads that did align, and if we can find an alignment, we infer by transitivity the start position for the read on the genome. This process is repeated an arbitrary number (100) times. While a more sophisticated algorithm might instead repeat until no more reads could be added, this was impractical for the existing algorithm, because it tended to add a few extraneous reads at each step.

Step 2. Assemble the sample genome with assistance from the reference. Supplementary Figure 1 describes the method for doing this: we first identify discrete windows on the reference genome where we predict that the sample and reference genome are identical—which we call stones—then ‘hop’ across the genome, from one stone to the next, successively assembling the regions between them. We assemble only the reads that are positioned on or near a given region. As implemented, step 2 is designed to yield a linear sequence representing a given region, and will thus work correctly only on haploid data sets. For diploid data, step 2 would need to be modified to return a graph for each region \(^{14,15}\). These graphs could then be glued together along the stones to yield an assembly of the entire genome, which would also be a graph.

Step 3. Define trusted bases on the assisted assembly. The reliability of the assembly varies from place to place, depending on depth of read coverage and the repetitiveness of the genome. Therefore before calling polymorphisms, we first mark the assembly to show which bases are trusted: sufficiently reliable to be used in polymorphism discovery. They are labeled as such if they have enough support from the reads (5X+), and there is little conflicting evidence. We note that a more sophisticated algorithm might instead report quality scores at each base, thereby providing a quantitative assessment of trustedness. Note also that for diploid genomes, step 3 would need to be adapted. One approach would be to mark each base on the assembly graph as trusted or not, but to carry this out, the method of step 3 would need to be changed significantly.

Step 4. Find trusted differences between the sample genome assembly and the reference genome. Polymorphisms are called at trusted differences. A trusted difference should be flanked by sequences that are identical and unique in both genomes. More specifically, let \(R_1\) denote the sample genome assembly and let \(R_2\) denote the related reference. A difference between \(R_1\) and \(R_2\) is defined by sequences \(XAY\) in \(R_1\) and \(XBY\) in \(R_2\), where \(A \neq B\), and where \(X\) and \(Y\) are \(K\)-mers, each occurring exactly once in both \(R_1\) and \(R_2\). We used \(K = 28\), a number chosen because it was slightly smaller than the read length. A difference is trusted if the rightmost base of \(X\), all bases of \(A\), and the leftmost base of \(Y\) are all trusted. A difference could be a mismatch, an insertion, a deletion, or a composite, e.g. two adjacent mismatches.

We note that while in this paper we focus on generating a list of differences between the sample genome assembly and the reference genome, the sample genome assembly itself (along with trustedness information) may provide a more satisfactory ‘final answer’. For example, the sample genome assembly may contain novel sequence that is not reflected at all in the list of differences. In addition, by using the assembly directly, one would avoid the arbitrariness intrinsic to all methods for defining the differences between two genomes.

VAAL ALGORITHM DETAILS

The algorithm takes as input reads, quality scores for those reads, a related reference sequence, and an adaptor sequence. It produces as output a list of polymorphism events.

Details for step 1: Position reads on the related reference genome.
(a) Trim the reads to remove adaptor sequence, then align each read to the related reference genome, as follows. For each 12-mer perfect match between the read and the genome, carry out a banded Smith-Waterman alignment. Alignments having more than 4 errors (mismatches plus indels) are ignored. The best alignment of the read is regarded as unique if the next best alignment has more than 2 more errors.

(b) Carry out an initial editing step: we use the alignments to identify putative differences, edit the reference genome accordingly. These initial edits define preliminary guesses for some of the differences between the sample and the reference genome. This substep was added because it improved the overall sensitivity of the algorithm.

(c) Error correct the reads. If in alignment to the reference, we observe a position on the reference where the most common base in the reads occurs at least 5 times as often as the next most common base, the next most common bases occur at most 3 times, and the coverage at the position is at most 3 times the mean coverage, then we edit the bases on the reads to change them to the most common base. Later, in step 3(a), we bring the uncorrected reads back to prevent propagation of ‘incorrect corrections’ that might be introduced by this substep.

(d) Trim the reads as follows. Form the K=20 unipath graph from the reads and the reference genome together. Shave the graph as in step 3. Trim bases off the end of each read until it aligns to the new graph. (Reads whose first K bases did not align were necessarily discarded.) Align the trimmed reads to the reference genome, keeping only those placements that are perfect and unique. We note that the true accuracy of these trimmed reads could be assessed by aligning to the sample genome. For example, for E. coli, > 99.9% of the trimmed reads were perfect.

(e) For each 20-mer that is unique in the reference genome, find its occurrences in the reads. Every such occurrence defines an inferred start position of the read on the genome. A given read is allowed to have more than one start position. A read assigned a start position is said to have been placed.

(f) If a read had a unique placement on the genome, as defined by its alignment in substep (a), but was not placed in substep (e), define its start position now. This step can produce additional placements because a 36 bp read carries more information than a 20-mer within it, and thus may be uniquely placed even though its 20-mers are not.

(g) For each 20-mer that is not in the reference, but is in a placed read, find reads that contain it and which have not already been placed. Add in their inferred start positions on the genome, by transitivity. But do not allow a read to be placed more than 10 times. This substep is iterated 100 times, but more iterations could be allowed.

Details for step 2: Assemble the sample genome with assistance from the reference.

(a) To identify the ‘stones’, windows on the reference genome where the sample and reference genome are predicted to be identical, we first find windows on the reference genome whose constituent 20-mers are unique in the reference genome, occur at least 5 times in the reads, and which are not adjacent in the reads to a kmer occurring at least 5 times in the reads but not occurring in the reference.

(b) Any such interval of size at least 60 is accepted as a stone.

(c) To keep adjacent stones reasonably close together, when two stones are separated by more than 300 bp, accept smaller windows from (a) as stones between them.

(d) To assemble the region between two adjacent stones, we first select those reads that are positioned by step 1 to be within 1000 bp of the two stones. In its present form, this condition limits the size of indels that can be detected.

(e) The graph that is created for this step is the unipath graph, and depends on a particular value of K. There are advantages associated with each choice of K, and to gain the advantage of both a ‘large’ and ‘small’ K, we built the unipath graph first for K=29, then K=20, then K=29 again, stopping at any stage where a successful assembly is achieved. Alternatively, if the assembly at a stage fails, we pass the unipaths onto the next stage as faux reads, thereby enhancing continuity.

(f) In the process of simplifying the unipath graph, when voting to determine which of two branches are correct, we require at least 3 reads to support the winning branch, and a ratio of at least 3:1 for winning:losing.

Details for step 3: Define trusted bases on the assisted assembly.

(a) Place reads on the assisted assembly as in 1(e), but here we use the uncorrected, untrimmed reads.

(b) A read base is called trusted if its quality score is at least 10.
(c) Delete bad read placements, in the following sense. For every 5-mer in every placed read, we require that there is another placed read, overlapping the same five bases and agreeing with them at every base that is trusted on the first read. Otherwise we delete the first placement.
(d) Paint the assembly: at each position we record the calls reported by reads placed over it, and separately, the trusted calls.
(e) For a given position on the assembly, let n1 be the number of trusted calls for the most-called base, and let n2 be the number of trusted calls for the second-most-called base. Similarly let N1 and N2 count all calls (trusted or not).
(f) Initially, we call a base on the assembly trusted if n1 ≥ 5, n1 ≥ 5n2, and N1 ≥ 2N2, and if the total coverage (by all bases) is at most 3 times the mean coverage. Some of these calls are reverted in (g) and (h).
(g) Find sequence in the assembly that is unsupported by reads: if a 12-mer in the assembly is not contained in a read lying over it, we mark the 12-mer and 10 flanking bases on each side as untrusted.
(h) Find ‘repeat joins’: places where the assembly is held together by repetitive sequence. To do this, for each placed read, trim bases on its end until the read matches the assembly perfectly. If after so trimming there is a juncture in the assembly that is ‘held together’ by an overlap of less than 12 bases or held together by a longer sequence that appears more than once in the assembly, we declare there to be a repeat join, and mark all the joining bases as untrusted.

Details for step 4: Find trusted differences between the sample genome assembly R₂ and the related reference R₁.
The following additional conditions are imposed to guarantee that X and Y minimally sandwich the difference between the two genomes and also to avoid duplicated K-mers:
(a) A and B are not both empty
(b) K-mers in XA and AY other than X and Y occur exactly once in R₁, and do not occur in R₂
(c) K-mers in XB and BY other than X and Y occur exactly once in R₂, and do not occur in R₁.
We note that in rare cases it is possible for step 4 to call valid differences between the sample genome assembly and the related reference, that are not present in the set of ‘callable’ polymorphisms. Because of the uniqueness requirement for K-mers in (b) and (c), this can happen under the following circumstances:
(i) the sample genome assembly is incomplete; (ii) consequently a K-mer is present multiple times in the sample genome, but only once in the sample genome assembly; (iii) a trusted difference between the sample genome assembly and the related reference is identified, involving this particular K-mer; (iv) this difference is a true difference between the sample genome and the related reference.

Isolation of mutant clones. V. cholerae N16961 ΔtolC was streaked on LB agar plates, and 5 well-isolated colonies were then picked and cultured in LB broth until stationary phase, at which time glycerol was added to each of these 5 cultures at a final concentration of 15%. These cultures were aliquoted and frozen at -80°C. To isolate drug-resistant mutants, selective LB plates were prepared containing rifampicin at 1.25 µg/ml (10X MIC). Aliquots from each of the 5 independently-derived V. cholerae clones were used to inoculate LB broth cultures that were grown overnight at 37°C. The overnight cultures were then diluted 200-fold and grown for 4 h prior to plating on selective medium. Each selective plate was seeded with 1 mL of the subculture (approximately 5-9 × 10⁹ cells) and incubated at 37°C until resistant colonies emerged (3 days). Resistant colonies derived from each of the 5 clones were expanded in LB broth media containing rifampicin (1.25 µg/ml) and genomic DNA was extracted using Invitrogen’s Easy-DNA Kit (Cat. No. K1800-01, Protocol #3).

Validation of differences between samples and reference sequences (column 3 of Table 1). For S. aureus and E. coli, we sequenced unpaired reads from the same DNA by 454 (runs 630018070817.{1,2}, 630456070820.2 for S. aureus, 630198070823.{1,2}, 630766070817.1, 631084070816.{1,2} for E. coli), then aligned the reads to the unmodified reference sequence. For each difference, there were at least 16 reads spanning the locus informatively, and of these, at least 88% supported the change. For M. tuberculosis, we sequenced PCR products from the same DNA on an ABI 3730. Again, the reads supported the change. Modified reference sequences for all three species are available at ftp://ftp.broad.mit.edu/crd/VAAL/bacterial_references.
Measuring the uncallable fraction of a genome (Supplementary Table 1). This was done by creating several thousand versions of the alternative reference sequence, each edited to introduce ~1000 artificial SNPs, with every position on the genome edited in one version, then comparing the edited alternative reference to the alternative reference, using step 4 of VAAL (with $R_1 =$ the edited alternative reference and $R_2 =$ the alternative reference), and observing which SNPs were reported. If, by this criterion, a SNP could not be detected, we regard it as part of the uncallable fraction of the genome.
Supplementary Results

PERFORMANCE OF VAAL ON COMPOSITE EVENTS

Of the 19 composite events in *M. tuberculosis*, there were 18 having two subevents (all but one of which were called) and 1 having 9 subevents (which was called). Of the 116 composite events in *S. aureus*, 104 were called. The most complicated of these consisted of 330 subevents in a 2.3 kb interval, which was called by VAAL.

WHY SOME CALLABLE POLYMORPHISMS ARE MISSED

To understand why some callable polymorphisms were not found by the VAAL algorithm, we investigated the case of *M. tuberculosis* F11 vs H37Rv in some detail (**Supplementary Table 2**). We found that missed polymorphisms generally have much lower sequence coverage than for called polymorphisms (e.g. 5X vs 18.5X for SNPs) and unusually high GC content (80% vs 66% for SNPs). The association of high GC content with low sequence coverage might result from difficulty in amplifying GC-rich regions in the sequencing process.

EFFECT OF COVERAGE ON SENSITIVITY

We next investigated the effect of depth of sequence coverage on sensitivity (fraction of polymorphisms discovered) and specificity (accuracy of polymorphisms discovered), again using *M. tuberculosis* as our test case (**Supplementary Table 3**). Using the full set of 9 million reads (18.5X coverage in Q20 bases), we called 96% of the callable polymorphisms. As sequence coverage dropped, the fraction of polymorphisms declined – slowly at first, then precipitously at around 6X coverage in Q20 bases. Regardless of the coverage, there were no false positives.

MUTATIONS OBSERVED

Two distinct mutations were observed, both changing amino acids (D516V and H526Y, **Table 2**). These substitutions were identical to previously reported mutations that confer rifampicin resistance in both *E. coli* and *Mycobacterium tuberculosis*16-18. The fact that only two distinct mutations were found (rather than five) is not surprising: only a limited spectrum of changes in rpoB provide both functionality and resistance16,19, and some of these mutations are far more common than others; mutations at H526 have been found in up to 43% of *M. tuberculosis* clinical isolates examined18. Additionally, the two observed mutations are among the mutations that have been shown to confer high-level resistance, while mutations at other positions in rpoB confer more modest levels of resistance to rifampicin16,20.

COMPARISON OF VAAL TO EXISTING ALGORITHMS

First, we tested the widely-used program MAQ13. Because MAQ does not call indels on unpaired reads, we could only test its SNP detection feature. Across the three bacterial genomes tested, VAAL called an average of 10% more SNPs. All of these were correct, with most being composite events. In duplicated regions, however, MAQ called substantially more SNPs. Some cases were correct, but other cases were problematic. The problematic cases involve regions in the reference genome that correspond to two distinct orthologous regions in the sample genome (i.e., copy number variation). In these cases, MAQ calls SNPs along the reference genome by combining some of the variants with either ortholog. VAAL employs a more conservative definition of SNPs and does not call these events as SNPs. These regions represent a small proportion of the genome.

Second, we replaced the built-in assisted assembly program used in VAAL with a published *de novo* short read assembly program (Velvet)15,21-25. The results of the two methods are almost identical at the highest levels of coverage tested. However, at lower levels of coverage, VAAL has substantially greater sensitivity and specificity. This is not surprising: the use of assisted assembly is expected to yield greater coverage than *de novo* assembly at low coverage.

In more detail, we carried out four comparisons (**Supplementary Tables 5-8**).
1. We called SNPs using the short read polymorphism detection algorithm MAQ\textsuperscript{13}. We could not make the same comparison for indels because MAQ will not call indels on unpaired reads.

2. We called polymorphisms by modifying VAAL to use the output of a de novo short read assembly program (Velvet\textsuperscript{19}), rather than the assisted assembly produced by VAAL. We note that while several programs are available for de novo short read assembly\textsuperscript{15,21-25}, these programs were designed for a different purpose (de novo assembly), and thus cannot take advantage of the additional power provided by a reference sequence, as VAAL can do. In addition, these programs do not mark assemblies to show ‘trusted’ bases, as VAAL does, and at least for Velvet, no quality scores are provided with the assembly. Therefore we used step 3 of VAAL to mark trusted bases on the de novo assemblies. Step 4 of VAAL was used to identify polymorphisms. We refer here to this hybrid method (Velvet + VAAL) as the ‘de novo approach’.

3. Similarly, we discovered polymorphisms using varying levels of coverage, as in Table 4, but now using the de novo approach.

4. Finally, we tried to find single mutations (as in Table 6) by separately assembling pairs of rifampicin sensitive and resistant \textit{V. cholerae} strains using Velvet, marking trusted bases on the assemblies using step 3 of VAAL, then comparing them to each other using step 4 of VAAL.

In all cases we attempted to optimize the results, either by calibrating input quality scores (MAQ), or by varying parameter settings (de novo approach). We described the results of these experiments.

1. First we compared the performance of VAAL and MAQ as SNP callers. We found that for some data sets, MAQ reports significantly more SNPs, while for others VAAL does. In total, for the three genomes tested, VAAL reported about 10\% more SNPs. Next we analyzed the differences. When VAAL reports more SNPs, it is usually because VAAL has found composite events that MAQ does not. Conversely, we found two reasons why MAQ reports more SNPs. First, in some cases MAQ was better at discovering SNPs at low coverage. This is probably because MAQ uses a probabilistic framework to score evidence, whereas VAAL uses hard-coded heuristic cutoffs. Second, MAQ calls SNPs in duplicated regions, whereas the uniqueness criteria in VAAL’s step 4 prevent this. Depending on the genome, the consequences of this were different. For \textit{M. tuberculosis}, with two exceptions, all SNPs called by MAQ were unambiguously correct, including ten in duplicated regions. However for \textit{S. aureus}, almost all of the 164 MAQ-only events appear in regions of the related strain having unclear orthology to the sample strain. For example, MAQ reports 34 SNPs between bases 44,800 and 47,000 on the related strain reference. This single region maps to two regions on the sample strain reference. Some of the SNPs appear to represent differences with the first such region, and others with the second such region. Moreover, from the reference sequences for the two strains, we see that the total number of substitutions with the first region is 387: thus the strains are highly diverged, only a small fraction of differences are reported by MAQ, and these differences are a mixture of differences between the related strain region and the two orthologous sample strain regions. It was not clear in what sense these reported differences could be labeled as true or false. In short, MAQ’s behavior of calling SNPs in duplicated regions can be either advantageous or problematic, depending on the specifics of the genomes.

2. For the de novo approach, sensitivity and specificity are almost identical to VAAL in the case of \textit{E. coli} (VAAL finds 1 more SNP), and comparable for \textit{S. aureus} (VAAL finds 14 more SNPs and 2 more indels, but 4 fewer composites). However, for the lower coverage and more challenging \textit{M. tuberculosis} data set, VAAL is clearly better: it finds 47 more SNPs, 5 more indels, and 7 more composites, and whereas VAAL has no false positives, the de novo approach yields 7. Of course it is not surprising that polymorphism detection via an assisted assembly works better than via a de novo assembly. Indeed what is surprising is de novo assembly works as well as it does.

3. When we reduced coverage, we found that performance deteriorated faster for the de novo approach. For example, at about 6X coverage, VAAL found 75\% of polymorphisms, but the de novo approach found only 44\%, and whereas VAAL had a 0\% false positive rate, the de novo approach had an 11\% false positive rate.
4. Finally, for the problem of calling single mutations, we compare the VAAL results to those of the *de novo* approach. We found that the *de novo* approach found the true mutation in all but one case, but also reported on average five incorrect polymorphisms, while VAAL reported none.
Supplementary References

16. Jin, D.J. and Gross, C.A. Mapping and sequencing of mutations in the *Escherichia coli* rpoB gene that lead to rifampicin resistance. *J Mol Biol.* **202**, 45-58 (1988).
17. Telenti, A. *et al.* Detection of rifampicin-resistance mutations in *Mycobacterium tuberculosis*. *Lancet* **341**, 647-650 (1993).
18. Kapur, V. *et al.* Characterization by automated DNA sequencing of mutations in the gene (rpoB) encoding the RNA polymerase beta subunit in rifampin-resistant *Mycobacterium tuberculosis* strains from New York City and Texas. *J Clin Microbiol.* **32**, 1095-1098 (1994).
19. Campbell E.A. *et al.* Structural mechanism for rifampicin inhibition of bacterial rna polymerase. *Cell* **104**, 901-912 (2001).
20. Taniguchi, H. *et al.* Rifampicin resistance and mutation of the rpoB gene in *Mycobacterium tuberculosis*. *FEBS Microbiol Lett.* **144**, 103-108 (1996).
21. Zerbino, D.R., Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research* **18**, 821-829 (2008).
22. Dohm, J.C., Lottaz, C., Borodina, T., Himmelbauer, H. SHARCXS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research* **17**, 1697-1706 (2007).
23. Warren, R.L., Sutton, G.G., Jones, S.J., Holt, R.A. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* **23**, 500-501 (2007).
24. Chaisson, M.J., Pevzner, P.A. Short read fragment assembly of bacterial genomes. *Genome Research* **18**, 324-330 (2008).
25. Hernandez, D., Francois, P., Farinelli, L., Osteras, M., Schrenzel, J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Research* **18**, 802-909 (2008).
Supplementary Resources

Software availability. The exact source code used in this paper is available at ftp://ftp.broad.mit.edu/pub/crd/VAAL, as is a manual. The code was applied identically in all cases. Revisions will be made available at http://www.broad.mit.edu/crd.

Data availability. Data described in Supplementary Tables 1, 4 will be deposited in the NCBI short read archive, and are presently available at ftp://ftp.broad.mit.edu/pub/crd/VAAL/bacterial_lanes. The lane identifiers (flowcell.lane) for the data in Supplementary Table 1 are 2039F.2, 9547.1, 2503.2, and for Supplementary Table 4: 4220.2, 4220.3, 5546.1, 5546.6, 5546.2, 5997.1, 5546.3, 5997.2, 5546.5, 5997.3. The 454 runs used for validation are available at ftp://ftp.broad.mit.edu/pub/annotation/human_microbiome/454_data.