Analyzing rare mutations in metagenomes assembled using long and accurate reads: Supplemental Material

Marcus W. Fedarko, Mikhail Kolmogorov, and Pavel A. Pevzner

A Read alignment **S2**

B Assembly graph **S6**

C Coverages and deletion-rich positions **S9**

D Demonstrating strainFlye on the ChickenGut dataset **S14**

E Applying LoFreq to the SheepGut dataset **S16**

F Growth of the number of p-mutations per megabase as p decreases **S19**

G Codon position analysis details **S19**

H Nonsynonymous, nonsense, and transversion decoy contexts **S21**

I Identifying mutations based solely on read counts **S25**

J Constructing and visualizing mutation matrices **S27**

K Diversity index details **S32**

L Hotspot genes in the three selected MAGs **S33**

M Identifying strains in the most mutated gene of BACT1 **S36**

N Plots of mutation locations **S36**

O Investigating coldspots **S39**

P Growth dynamics **S43**

Q The link graph structure for haplotype visualization **S44**

R Haplotypes of the most mutated gene in CAMP **S47**

S Smoothed haplotype assembly graphs **S52**

A Read alignment

strainFlye aligns reads to edge sequences in the assembly graph in order to identify mutations that may have been "smoothed over" in the process of metagenome assembly: the resulting alignment is a prerequisite for mutation identification, and impacts all downstream analyses in strainFlye. The `strainFlye align` command uses minimap2 (Li, 2018) to compute an alignment, after which it filters various sources of noise from the alignment; however, this command can be skipped if a user has an existing BAM file representing an alignment of reads to edges. This Supplement describes how `strainFlye align` works.

Performing alignment. strainFlye uses minimap2 (Li, 2018) to align all reads against the assembly graph's sequences. By default, we use the `asm20` preset parameter set for the alignment per recommendation from the minimap2 documentation for use with HiFi data; however, this and other minimap2 parameters are easily adjustable by the user, for example if the `map-hifi` preset (introduced in recent versions of minimap2) is preferred instead. We also use the `--secondary=no` parameter to exclude secondary alignments (The SAM/BAM Format Specification Working Group, 2022). After minimap2 creates the initial alignment, we then use SAMtools (Li *et al.*, 2009) to convert it to a sorted and indexed BAM file.

In the context of the SheepGut dataset, some edges may have no reads aligned to them, even before we filter the alignment. In our experience this can happen for edges that are shorter than the usual read length; these edges (which tend to have extremely high coverages) seem to be an assembly artifact from metaFlye, likely corresponding to repetitive sequences, and most of the analyses in this paper implicitly ignore these edges.

We note that this alignment process retains supplementary (also referred to as "chimeric") alignments, in which a read may have multiple distinct linear alignments (The SAM/BAM Format Specification Working Group, 2022); these alignments can be useful in describing structural variation, or in representing a read that spans the start and end of a single circular sequence. Although we preserve supplementary alignments, we do take pains to remove reads that have alignments overlapping with other alignments from the same read: the remainder of this Supplement provides details on the filtering steps strainFlye performs on minimap2's alignment, as well as potential extensions to these filters.

Filtering out overlapping supplementary alignments. Given a single read with two linear alignments A_1 and A_2 to a reference genome, there are two types of "overlap" we could define between A_1 and A_2. We note that these types of overlap are not necessarily mutually exclusive.

The first type of overlap is defined with respect to the reference genome. In this case, A_1 and A_2 align to regions of the reference genome that overlap with each other.

The second type of overlap is defined with respect to the read sequence itself. In this case, A_1 and A_2 originate from overlapping regions of the read sequence.

We refer to these types of overlap as *reference* and *read* overlap, respectively. The SAM specification currently states that "[for] a chimeric alignment, the linear alignments constituting the alignment are largely non-overlapping" (The SAM/BAM Format Specification Working Group, 2022); however, it is unclear whether the type of overlap described is reference or read overlap.

Neither type of overlap bodes particularly well in the context of our analyses. Here, we focus on removing reference-overlapping supplementary alignments; the consequences of these alignments on downstream analysis seem more problematic than the consequences of read-overlapping alignments.

While performing the analyses described in this paper, we noticed that a small, albeit substantial, number of supplementary alignments from individual reads exhibited reference overlap. Strangely, these overlaps sometimes spanned thousands of nucleotides in a read; these may thus be a sequencing artifact, since HiFi reads have been shown to be vulnerable to molecular chimeras (Wenger *et al.*, 2019).

Before filtering overlapping supplementary alignments and before filtering partially-mapped reads, the following supplementary alignment statistics held for the three selected MAGs.

In CAMP, 8,645 / 503,385 (1.72%) unique reads aligned to within the MAG had supplementary alignments within CAMP. On average, these 8,645 reads had 2.07 alignments within CAMP. Furthermore, 2,180 / 503,385 (0.43%) unique reads aligned to within CAMP had supplementary alignments within CAMP *and* had reference overlap between at least one pair of their alignments within CAMP. The average length of these overlaps (considering all pairs of overlapping alignments within CAMP from the same read) was 580.12 bp.

In BACT1, 6,067 / 268,075 (2.26%) unique reads aligned to within the MAG had supplementary alignments within BACT1. On average, these 6,067 reads had 2.04 alignments within BACT1. Furthermore, 470 / 268,075 (0.18%) unique reads aligned to within BACT1 had supplementary alignments within BACT1 *and* had reference overlap between at least one pair of their alignments within BACT1. The average length of these overlaps (considering all pairs of overlapping alignments within BACT1 from the same read) was 1,417.80 bp.

In BACT2, 5,497 / 745,461 (0.74%) unique reads aligned to within the MAG had supplementary alignments within BACT2. On average, these 5,497 reads had 2.04 alignments within BACT2. Furthermore, 728 / 745,461 (0.10%) unique reads aligned to within BACT2 had supplementary alignments within BACT2 *and* had reference overlap between at least one pair of their alignments within BACT2. The average length of these overlaps (considering all pairs of overlapping alignments within BACT2 from the same read) was 2,561.42 bp.

Regardless of whether these reads with overlapping supplementary alignments correspond to chimeras or not, the presence of these overlaps complicates interpretation of the alignment data—especially in analyses that take into account the sequence carried on individual reads, for example phasing and codon mutation analyses.

Although it is possible to only filter some of a read's supplementary alignments as needed to prevent reference overlap, we act on the assumption that the presence of *any* reference-overlapping alignments for a given read implies that this read is unreliable for our analyses. This will likely be overly strict—some "correct" reads may simply happen to have reference overlap in their supplementary alignments—but we contend that, in the context of identifying rare mutations, overly strict filtering is defensible. We therefore identify all reads with reference-overlapping supplementary alignments relative to any of the edges in the alignment, and completely filter all of these reads from the alignment. The impacts of this filter on sequencing coverage in the three selected MAGs are shown in Fig. S4.

Slight error in the overlapping supplementary alignment filter. We note that the version of this filter we used for the alignments throughout this paper was overly strict, due to an off-by-two error in its computation of "overlap" between linear alignments of a read (rather than subtracting 1 from each alignment's end coordinate, it mistakenly added 1). This resulted in a relatively small amount of reads being erroneously removed by our filter and thus not included in this paper's results—anecdotally, many of these removed reads represent cases where minimap2 would create two distinct linear alignments of a read directly next to each other, but not overlapping, on a single contig.

Based on analyzing initial versions of the SheepGut and ChickenGut alignments (reflecting just the output of minimap2, before applying our filters; we re-ran the alignment processes to obtain these "intermediate" alignment files), the number of reads that would have been erroneously removed by this bug is relatively small: 27,211 reads (0.12%) in SheepGut, and 1,596 reads (0.08%) in ChickenGut. In particular, the three selected MAGs in SheepGut (CAMP, BACT1, BACT2) are only subtly affected: there were 16, 9, and 3 reads erroneously filtered for these MAGs, respectively. (For reference, the statistics quoted earlier in this Supplement about overlapping supplementary alignments in the three selected genomes are correct.) We expect that this bug should have a minimal impact on the paper's conclusions; moving forward, we have also fixed this bug in version 0.2.0 of strainFlye.

Filtering out partially-mapped reads. Although we instruct minimap2 to not output secondary alignments, our preservation of supplementary alignments means that a single read may still be mapped to multiple contigs.

Reads that are mapped to multiple contigs in this way may be of good quality and worth retaining: for example, if a genome is split into multiple edges within a connected component of the assembly graph, then we might expect some reads' alignments to also be split across different edges within this component. Similarly, since most bacterial genomes are circular (Casjens, 1998), reads that happen to cover both the "start" and "end" of a circular MAG may be split into a supplementary alignment (The SAM/BAM Format Specification Working Group, 2022).

However, some reads are less desirable. For example, a read with alignments to edges located in completely different connected components of the assembly graph—with only short alignments to individual edges—is likely not useful in identifying rare mutations, and should be removed. This section details our attempt to filter out these sorts of "partially-mapped reads," with the goal of removing undesirable supplementary alignments between unrelated contigs while preserving ordinary supplementary alignments within a contig or between adjacent contigs.

An earlier method we attempted to use for this involved simply filtering out reads that contained more than a set number of soft-clipping operations, as mentioned at `https://github.com/samtools/samtools/issues/1169`. However, we noticed that this caused coverage drops near the ends of MAGs' sequences due to penalizing the aforementioned "desirable" supplementary alignments.

We therefore took a different strategy in the current implementation of this filter. Our filter considers each edge sequence, and then considers all reads aligned to this edge, counting the number of match and/or mismatch operations in the CIGAR strings of each read's alignment to this edge. If an assembly graph file is provided to `strainFlye align` (so that we know which edges are adjacent to each other), and if this edge has less than 50 adjacent edges in the graph (ignoring links between the edge and itself), the script also checks reads aligned to these adjacent edges and also counts match/mismatch operations from any reads that are aligned to both the adjacent edge and the initially-considered edge. A given read is only included in the filtered alignment for this edge if the sum of all match/mismatch operations in this read in this edge (and in adjacent edges, if applicable) is at least 90% of the read length.

We note that this method is not perfect—for example, supplementary alignments can still have overlaps relative to their read sequence (as described earlier in this Supplement), so the sum of match/mismatch operations as we currently compute them is not necessarily a perfect measure of how much of a read is mapped to a given MAG. Additionally, the reason for the coverage drops near the end of CAMP (Fig. S3; CAMP is the only of the three selected MAGs located in an assembly graph component containing other edges), in spite of our attempts to allow alignments to adjacent edges in the same graph to count towards the 90% threshold, is unclear. This may be due to the "full genome" of CAMP being split across the edges in its component, and this complicating the alignment process. In spite of these imperfections, from manual examination of its results the filter's behavior seems acceptable. The impacts of this filter on sequencing coverage in the three selected MAGs are implied in the comparison between the second plot for each MAG in Fig. S4 and the plots in Fig. S3: removing partially-mapped reads results in many of the extreme "peaks" in Fig. S4 being removed.

Potential filtering extensions. Although strainFlye's default alignment filtering methods seem suitable for our purposes, these methods could certainly be extended to perform even more stringent filtering. More stringent methods should further increase the likelihood that any observed mismatch at a position corresponds to a real mutation, rather than an error or an artifact of alignment.

For example, homonucleotide sequences have been known to pose problems for long-read sequencing technologies; although we have not explicitly accounted for this effect in our analyses, it would be possible to limit our focus in certain cases—for example, the construction of a decoy contig—to more complex genomic regions. Similarly, the filtering of contigs with many "deletion-rich" positions (Supplemental Material **"Coverages and deletion-rich positions"**) may also prove useful.

B Assembly graph

Choice of dataset and assembler. Although strainFlye was developed to work with output from the metaFlye assembler (Kolmogorov *et al.*, 2020), it can be extended to MAGs generated by other HiFi-based tools such as HiCanu (Nurk *et al.*, 2020), hifiasm-meta (Feng *et al.*, 2022), and LJA (Bankevich *et al.*, 2022). However, our reliance on long and accurate reads implies that the use of short reads, or long error-prone reads, will likely pose nontrivial challenges for strainFlye.

Running metaFlye. When assembling the SheepGut dataset using metaFlye (Kolmogorov *et al.*, 2020), we did not use metaFlye's `--keep-haplotypes` mode. We chose this course of action in order to reduce fragmentation in the assembly graph (giving us longer, coarser MAGs to use as a starting point for studying mutations within these MAGs).

Connected components of the assembly graph. 21,750 out of 78,793 edges (27.6%) within the SheepGut assembly graph produced by metaFlye are located within a single large "hairball" connected component of the graph. This is a common characteristic of metagenomic assembly graphs due to, for example, overlap between reads from highly conserved regions (such as 16S rRNA genes) or the presence of eukaryotic genomes within a metagenome (this dataset includes some eukaryotic genomes (Kolmogorov *et al.*, 2020)). The second largest component of the assembly graph contains 252 edges. 45,842 out of 45,988 connected components (99.7%) contain 10 or fewer edges and 43,045 (93.6%) consist of a single edge.

There is a large amount of variance between edge coverages by reads, as reported by metaFlye, in the assembly graph. The maximum edge coverage is 7,536,987x (for a 580bp long edge located in the "hairball" largest connected component). The average coverage across all edges (rounded to the nearest integer) is 951x, while the median coverage is 14x:

Name	Length (Mbp)	Coverage	Protein-Coding Genes (PCGs)	Intergenic Regions (IRs)	Positions in PCGs (Mbp)	Positions in IRs (Mbp)
CAMP	1.29	4,159x	1,297	788	1.19	0.10
BACT1	2.15	1,415x	1,761	1,559	1.95	0.21
BACT2	2.81	2,993x	2,567	2,196	2.29	0.51

Table S1: The three selected MAGs for mutation analyses. The names CAMP, BACT1, and BACT2 refer, respectively, to edges with IDs 6104, 1671, and 2358 in the metaFlye assembly graph. We note that CAMP's completeness is relatively low because it was not assembled into a single circular sequence. Since predicted PCGs can overlap, we "count" a position present in multiple genes only once when computing the amount of positions in PCGs.

the discrepancy between average and median is likely due to the presence of high-coverage outlier edges, such as the aforementioned maximum-coverage edge.

Summarizing all graph components by length and coverage. As shown in (Antipov *et al.*, 2022), many of the connected components of the assembly graph containing a small number of edges likely arise from low-coverage genomes, viruses, or plasmids. These are therefore not relevant to our selection of high-coverage bacterial MAGs for these analyses.

In an attempt to distinguish between these cases and components that capture bacterial genomes, we plotted a summary of edge coverage and length for each component. Fig. S1 shows a scatterplot where each component is represented as a single symbol, and is plotted by the total edge length in the component and by the sum of (coverage times length) for all edges in the component divided by the total edge length in the component (Antipov *et al.*, 2022).

This plot provides insight into which components in the graph are likely to represent bacterial genomes. We represent the thresholds used in our initial selection process for edges in the graph (length $\geq$ 1 Mbp, coverage $\geq$ 1,000x) as vertical and horizontal lines in the plot, respectively. This display clarifies that, while many components have a high "average" coverage or a high total length, few components have both. These components represent edges that serve as good candidates for demonstrating the analyses shown in this paper.

We do note that the length threshold shown here could arguably be reduced, since some bacteria have genomes smaller than 1 Mbp (Dicenzo & Finan, 2017); and that we initially used these thresholds to search for edges, not components, in the graph. However, this figure nonetheless indicates that the three selected MAGs are, relative to those available from this dataset, of good quality.

Details about the three selected MAGs. Table S1 provides basic information about CAMP, BACT1, and BACT2.

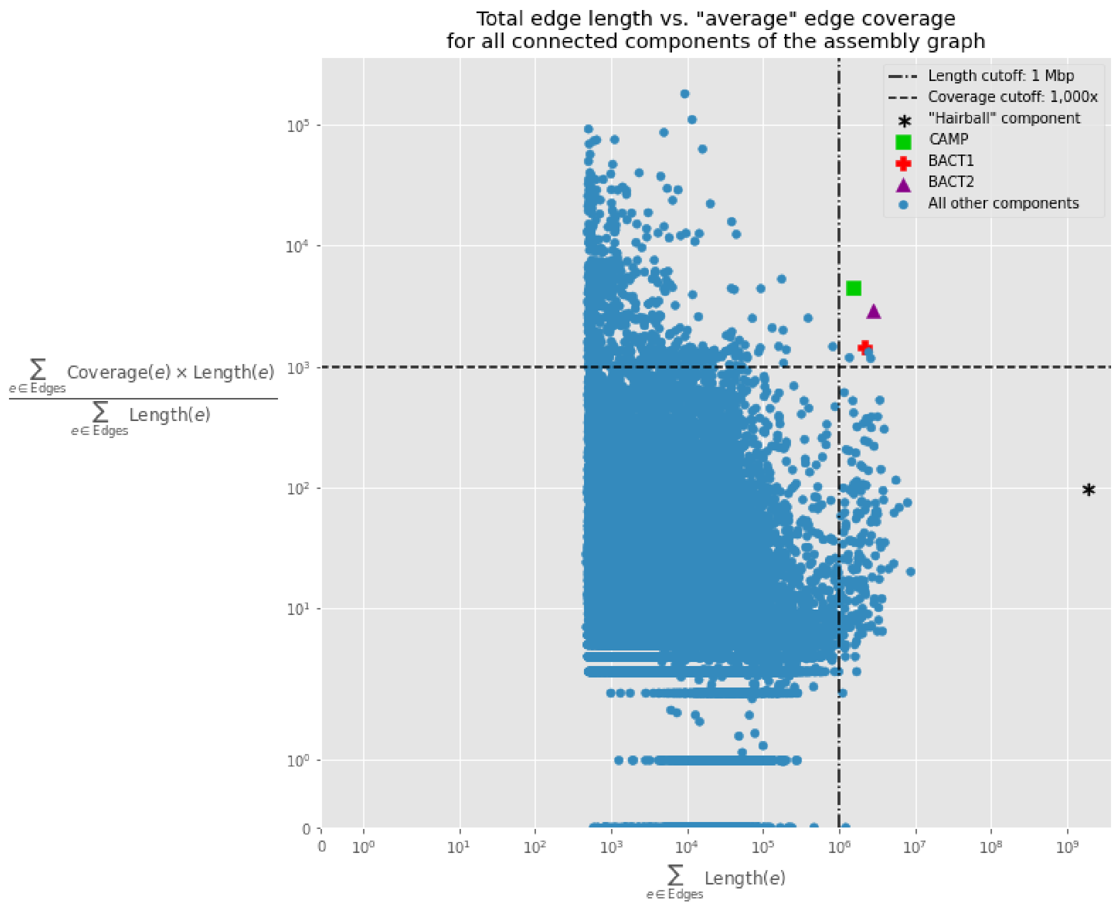


Figure S1: Summary of coverage and length across all connected components of the assembly graph. Each symbol (asterisk, square, plus sign, triangle, circle) represents a single component in the graph: the x-axis shows the total edge length for each component, and the y-axis shows an aggregate measure of coverage for each component (using coverages as defined by metaFlye and not based on the alignment-based $reads(pos)$ values computed in **"Computing mutation spectra"** and used elsewhere in this paper). In order to relate this plot to our selection criteria for MAGs (discussed in the section **"Demonstrating strainFlye"**), we plot lines showing the length and coverage minimum cutoffs we used on the plot (length $\geq$ 1 Mbp, coverage $\geq$ 1,000x). The few points located in the top-right "quadrant" defined by these lines represent MAGs that pass these cutoffs. The components containing the three MAGs we focus on throughout these analyses—CAMP, BACT1, and BACT2—are colored specially and shown as distinct symbols in the plot. It is clear that these three MAGs, and only three additional components in the graph, pass both the length and coverage cutoffs. We assign the "hairball" component of the graph a special color and symbol in order to simplify comparison of it with the other components in the graph: although this component has a high total edge length (causing it to appear as an outlier even on a log scale), its aggregate coverage is relatively low. Lastly, we note that the lack of components to the left of x = 10^2 is a reflection of the fact that all components in the graph have a total edge length greater than 100 bp. However, the aggregate measure of coverage is not similarly high: for many components, this value is 0x. We have set both the x and y axes to both have a minimum of 0 in order to clarify this.

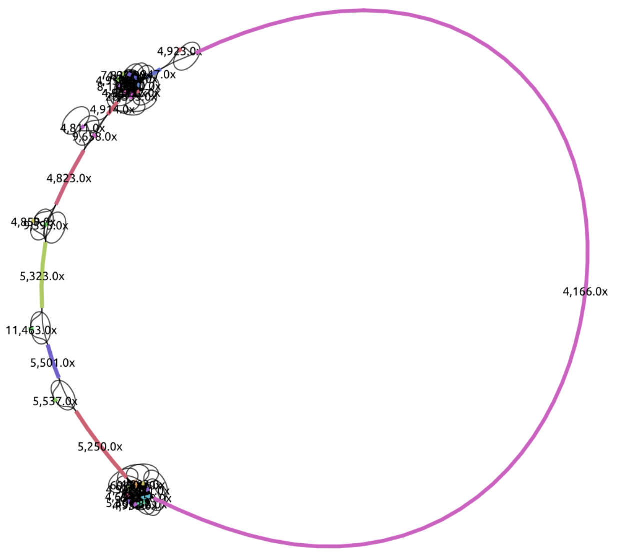


Figure S2: Visualization of the connected component containing edge 6104 (corresponding to CAMP) in the larger assembly graph, shown to clarify the component's structure. Coverages (reported by metaFlye, and slightly different from the (mis)match coverages based on our alignment that are shown in Table S1) are shown overlaid on the edges within the graph. The long magenta edge shown with coverage 4,166.0x is edge 6104. We visualized this graph using Bandage (Wick *et al.*, 2015).

BACT1 and BACT2 were assembled into edges located in isolated connected components (i.e. connected components containing just one edge) of the full dataset's assembly graph. The MAG we have named CAMP, however, is located in a component of the graph containing 31 other edges: this component is shown in Fig. S2, for reference. Although CAMP has not been completely assembled, we have nonetheless included it in the analyses throughout this paper due to its relatively high coverage. CheckM (Parks *et al.*, 2015) estimates "completeness" values of 81.4% for CAMP, 96.9% for BACT1, and 94.6% for BACT2; and "contamination" values of 0.0% for CAMP, 0.6% for BACT1, and 0.4% for BACT2. The relatively low completeness estimate for CAMP is likely a result of its incomplete assembly.

Taxonomic classifications of the three selected MAGs. We used Kaiju (Menzel *et al.*, 2016)'s web server to classify these MAGs. We used the "NCBI BLAST nr +euk" reference database and default parameters.

C Coverages and deletion-rich positions

Coverages of the three selected MAGs in SheepGut. Although many of the analyses in this paper assume that coverage is roughly uniform throughout the three selected MAGs, their coverages do vary somewhat. To illustrate this, Fig. S3 shows plots of coverage for each position in the three selected MAGs. Unlike the coverage plots shown in the Supplemental Material **"Growth dynamics"**, these coverages are not binned or normalized.

Coverage drops in SheepGut. Fig. S3 reveals a surprising result in BACT2 (and, to a much smaller extent, CAMP and BACT1): the presence of many individual positions to which many deletions are aligned, causing the coverage based purely on matches and mismatches for these positions to be relatively low. Manual inspection of some of these positions with samtools tview (Li *et al.*, 2009) confirms this.

More gradual coverage drops like those seen near the ends of CAMP, or near the middle of BACT1, could be in theory explained by real coverage variation in sequencing or noise introduced through the assembly and/or alignment processes. Fig. S4 presents coverage plots similar to those in Fig. S3, demonstrating that the two gradual coverage drops near the ends of CAMP (and, to an extent, in the middle of BACT1) indeed seem to have been introduced by our alignment filtering steps (Supplemental Material **"Read alignment"**). However, these plots also show that the high amounts of deletions in the aforemented positions in BACT2 were not introduced by these filters, indicating that further investigation is needed as to the origin of this phenomenon.

Deletion-rich positions in SheepGut and ChickenGut. We define a position in a contig as *deletion-rich* if, for some integer d, there are d or more deletions aligned to this position. For example, most of the outlier positions in BACT2 are deletion-rich for $d = 500$.

This definition enables us to ask two immediate questions. First, considering various values of d, do there exist many deletion-rich positions in other contigs in SheepGut besides the three selected MAGs? And, if so: does this pattern also hold in other contigs in other HiFi metagenomic datasets besides SheepGut?

Fig. S5 answers both of these questions affirmatively. First, many contigs in SheepGut (besides the three selected MAGs) contain many deletion-rich positions even for high values of d; second, this pattern also holds for many contigs in ChickenGut (Supplemental Material **"Demonstrating strainFlye on the ChickenGut dataset"**). Since the contigs in ChickenGut were assembled using hifiasm-meta (Feng *et al.*, 2022), while the contigs in SheepGut were assembled using metaFlye (Kolmogorov *et al.*, 2020), the persistence of many deletion-rich positions across many contigs in both datasets implies that these positions are not introduced by the assembly process.

These deletion-rich positions may happen to correspond to real variation in these MAGs; they could also, in theory, be introduced by minimap2 (Li, 2018), which we used to align reads to contigs for both SheepGut and ChickenGut. However, we believe it is much more likely that these positions are an artifact of HiFi sequencing that—for some reason—introduces many deletions at certain positions, e.g., positions corresponding to homonucleotide and dinucleotide runs that often trigger errors in HiFi reads (Nurk *et al.*, 2020). Other analyses

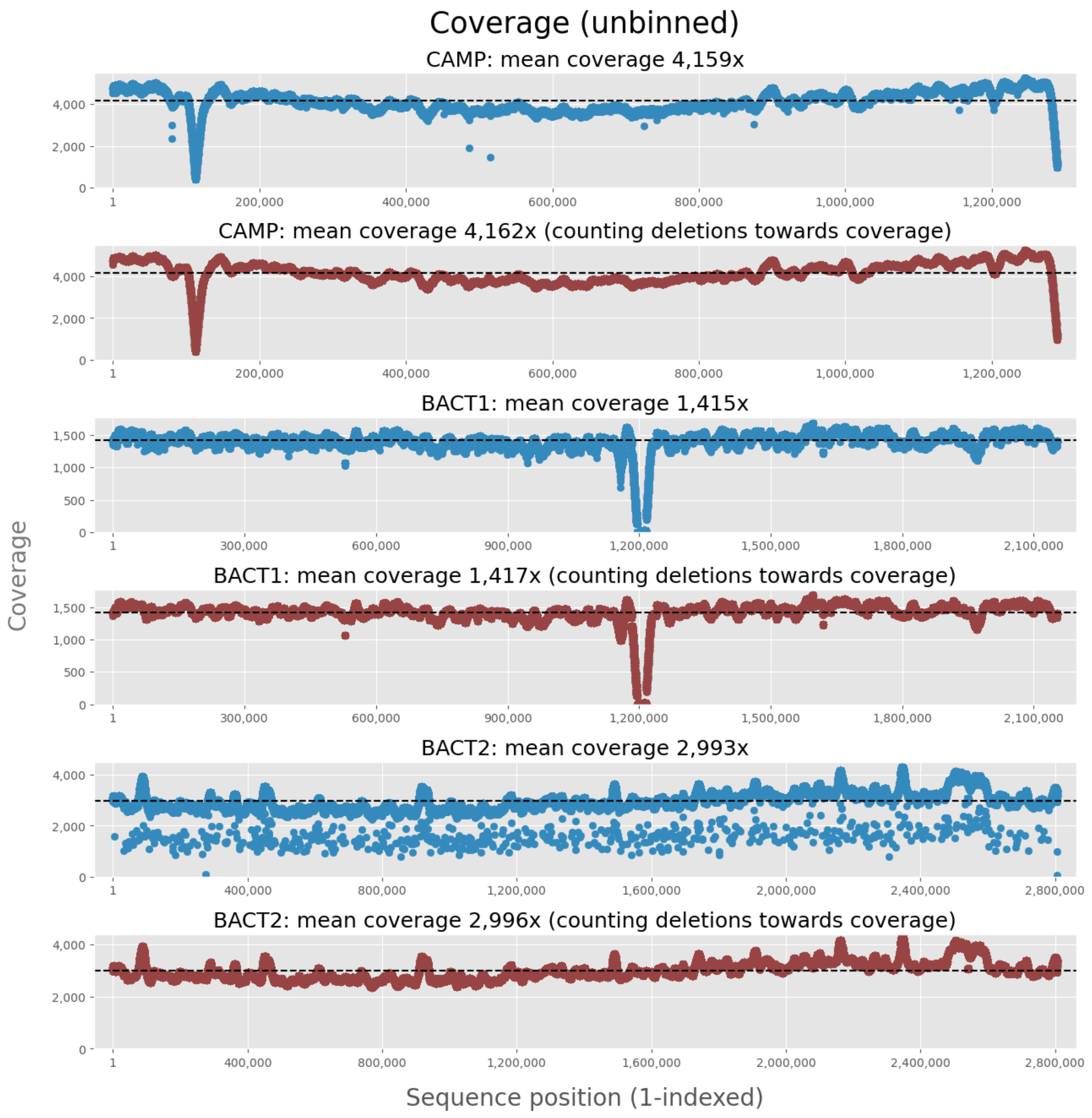


Figure S3: Coverage throughout the three selected MAGs, with and without deletions. Each dot represents a single position in a MAG. For the first plot for each MAG (using blue points), the coverage at a given position is defined as the total number of aligned reads to this position that have a match or mismatch operation, ignoring insertions and deletions; this is also how we computed the average MAG coverages in Table S1. For the second plot for each MAG (using red points), each position's coverage now includes the number of deletions aligned to it. The average coverage for each plot is shown as a horizontal dashed line. Although most parts of these MAGs have somewhat uniform coverage, especially in the "including-deletions" plots, both plots for CAMP and for BACT1 have noticeable gradual drop(s) in coverage. Furthermore, BACT2 has a relatively large amount of "outlier" positions with many deletions: these positions have relatively low coverages for the first plot, and these low coverages vanish when counting deletions towards coverage. These "deletion-rich" positions are investigated in this Supplement.

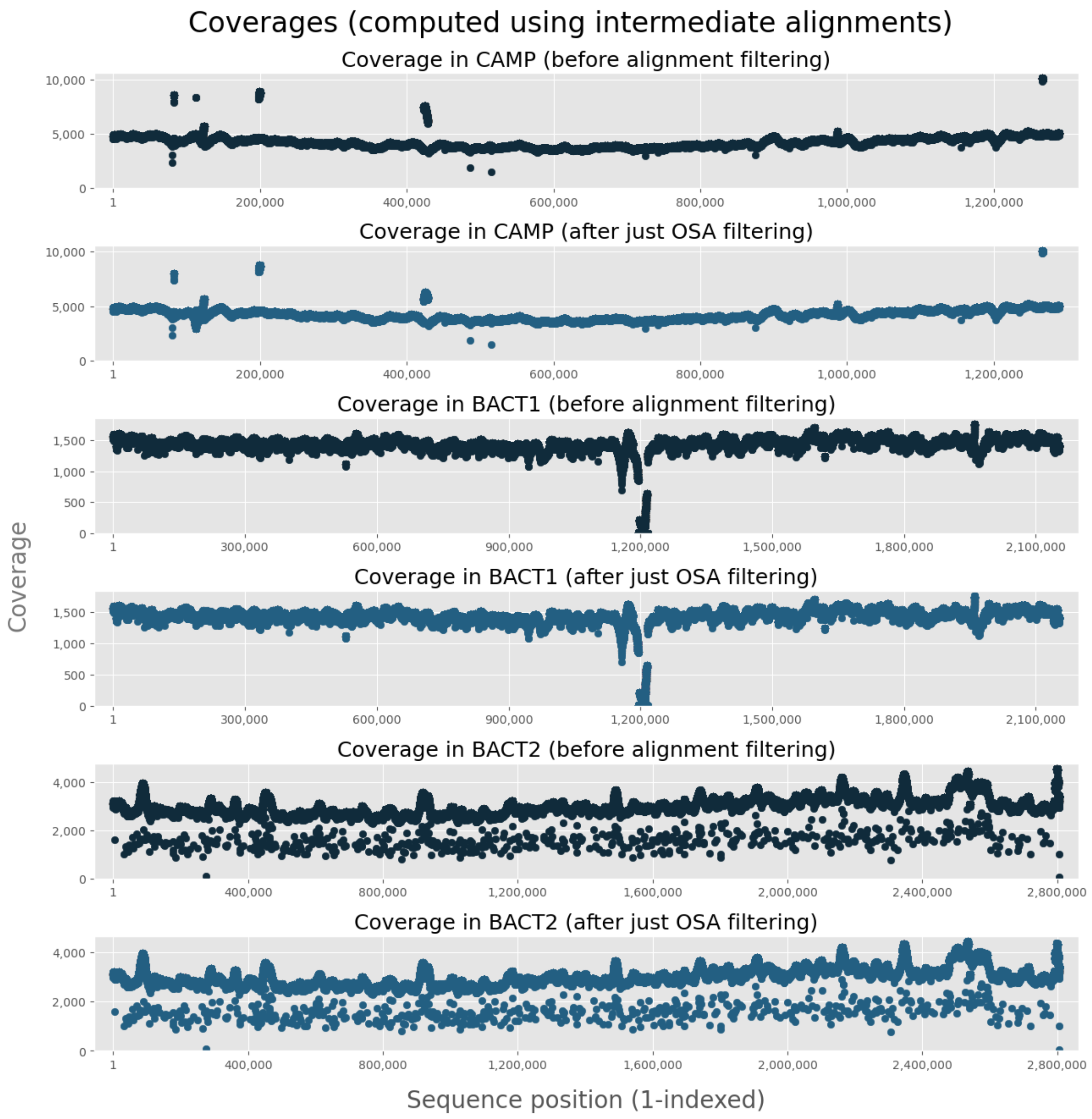


Figure S4: Coverage throughout the three selected MAGs, before each of the two alignment filtering steps. The first row for each MAG plots the coverage after performing alignment using minimap2 (Li, 2018), but before any additional filtering. The second row for each MAG plots the coverage after using just the overlapping-supplementary-alignment filter (Supplemental Material **"Read alignment"**; this filter is also impacted by the off-by-two bug discussed there). Finally, Fig. S3 demonstrates the coverages for each MAG after both the overlapping-supplementary-alignment filter and the partially-mapped read filter (Supplemental Material **"Read alignment"**). Primarily, this figure demonstrates to us that the many deletion-rich positions in BACT2 were not introduced, somehow, by these filters. It does, however, indicate that the drops in coverage near the ends in CAMP were introduced by the partially-mapped read filter, since these drops are not observed in this figure but are observed in Fig. S3. Parts of the coverage drop in the middle of BACT1 seem also to have been introduced by the partially-mapped read filter. We note that we needed to re-run the full alignment process for SheepGut to produce this figure, since we did not initially preserve intermediate alignment files in order to save storage space. These intermediate alignments may thus not be a perfect match with the intermediate alignments upstream of the one used elsewhere in this paper.

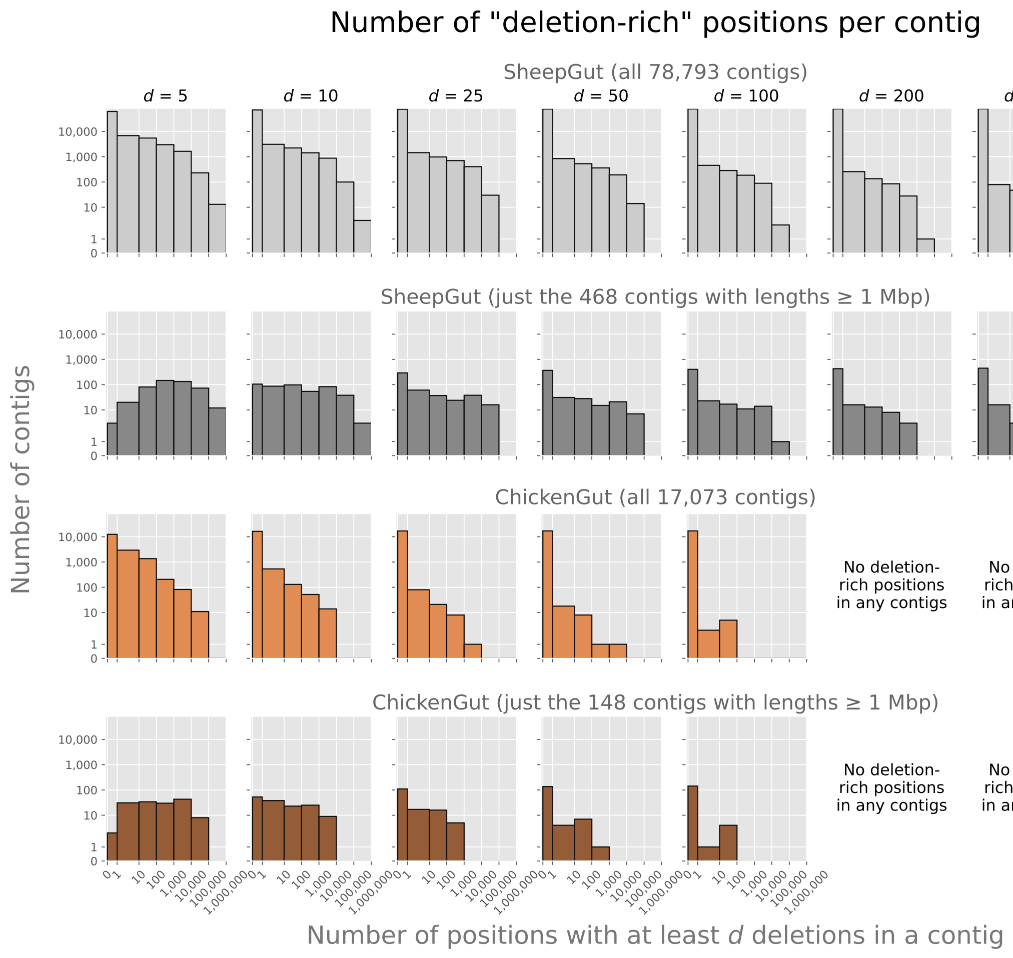


Figure S5: Histograms of the amounts of deletion-rich positions in each MAG. Each row corresponds to a different dataset, or subset of a dataset; each column describes a value of d we use to classify a position as deletion-rich. These plots demonstrate that, in general, many contigs in each dataset contain many deletion-rich positions, indicating that the high amount of "outlier" positions shown in BACT2 in Figs. S3 and S4 is not unique to this MAG. We omit histograms for $d \geq 200$ for the two ChickenGut datasets because no deletion-rich positions exist for these values of d in either dataset: we believe that this is primarily due to the coverages in the contigs in ChickenGut being, in general, smaller than those of SheepGut (Supplemental Material **"Demonstrating strainFlye on the ChickenGut dataset"**).

of HiFi data have shown gradual coverage drops and suggested that these may arise from problems in HiFi sequencing (Nurk *et al.*, 2020, 2022); these deletion-rich positions may have a similar origin.

Setting aside their origin, deletion-rich positions are outnumbered by other positions in the three selected MAGs. In BACT2, for example: for $d \in \{5, 10, 25, 50, 100, 200, 500, 1{,}000\}$, BACT2 (2,806,161 bp long) has $\{413{,}276, 222{,}869, 64{,}660, 17{,}458, 2{,}272, 652, 591, 541\}$ deletion-rich positions, respectively. The percentages of BACT2's length comprised of deletion-rich positions are thus $\{14.727\%, 7.942\%, 2.304\%, 0.622\%, 0.081\%, 0.023\%, 0.021\%, 0.019\%\}$, showing that (for large values of d) BACT2 has relatively few deletion-rich positions.

Implications of deletion-rich positions and other coverage drops. It is unclear what the "best practice" is for how to handle the presence of deletion-rich positions in these datasets—whether we should still include deletion-rich positions in our analyses, ignore these positions for the purposes of mutation calling, or even ignore entire contigs that contain many deletion-rich positions. For now, we do not explicitly filter these positions or the contigs that contain them, although doing so may be a promising avenue for future analyses of HiFi data.

Similarly, we do not take special measures to filter contigs with more gradual coverage drops like those in CAMP or BACT1. In general, mutations called in low-coverage regions in a contig should be subject to scrutiny; calling p-mutations while using a higher *minAltPos* value may improve the reliability of the called mutations (see also Supplemental Material **"Identifying mutations based solely on read counts"**). This is true even in contigs with coverage drops, since using higher values of *minAltPos* implictly increases the coverage needed to call a p-mutation; however, such a measure will necessarily result in identifying fewer mutations in low-coverage regions. Calling few mutations in low-coverage regions of a contig will have the unpleasant side effect of limiting our ability to phase this contig, and may result in the mis-identification of coldspots. To an extent, these problems are inherent to the dataset under consideration, although they may be possible to circumvent through more sophisticated mutation identification tools.

D Demonstrating strainFlye on the ChickenGut dataset

To show that strainFlye can be applied easily to other datasets besides SheepGut—and, in addition, to the output of other assemblers besides metaFlye (Kolmogorov *et al.*, 2020)—we applied it to a HiFi read-set (referred to as "ChickenGut") from a chicken gut metagenome that was assembled using hifiasm-meta (Feng *et al.*, 2022).

Input reads and hifiasm-meta assembly. We downloaded the reads in this dataset in SRA format (**"Methods"**), and used the fastq-dump tool to create a FASTQ file of these reads.

We began our analysis of this dataset using the hifiasm-meta assembly generated in (Feng *et al.*, 2022). Since hifiasm-meta's output includes multiple assembly graph files, we used the contigs in the assembly graph labelled `chicken.hifiasm-meta.p_ctg.gfa.gz`, based on the advice given in `https://github.com/xfengnefx/hifiasm-meta/issues/10`. This assembly graph includes 17,073 contigs.

Alignment and naïve p-mutation identification. Using strainFlye, we converted the assembly graph to a FASTA file of contigs (`strainFlye utils gfa-to-fasta`) and then aligned reads to these contigs (`strainFlye align`; see Supplemental Material **"Read alignment"**).

We then naïvely called p-mutations in these contigs, as described in section **"Computing mutation spectra"**, using `strainFlye call p-mutation` with $p = 1\%$. This value of 1% serves as the minimum value of p used in drawing the FDR curves below; as we will explain shortly, the coverage of MAGs in ChickenGut is generally lower than that in SheepGut, limiting our ability to call p-mutations with frequency rarer than 1%.

Decoy contig selection and FDR estimation. In order to visualize FDR curves of mutation identifications in this dataset's contigs, we performed FDR estimation using `strainFlye fdr estimate`. We provided a file of diversity indices produced by `strainFlye call p-mutation` as input to the FDR estimation step, in order to allow the step to automatically select a suitable decoy contig (**"Methods"**).

Since the average coverage of contigs in ChickenGut is generally lower than that in SheepGut, no contigs with lengths $\geq minLength = 1$ Mbp and average coverage $\geq minCov =$ 1,000x exist in this dataset (section **"Demonstrating strainFlye"**). In this situation, strainFlye raises an error explaining the situation to the user (**"Methods"**): here we discuss how we adjusted the parameters of decoy contig selection accordingly.

We can identify potential decoy contigs either by lowering our thresholds for minimum length, or for minimum average coverage. In total, 148 / 17,073 = 0.87% of contigs in ChickenGut have lengths $\geq minLength = 1$ Mbp; we thus focused on lowering our minimum average coverage threshold. These 148 contigs have average coverages ranging from 4x to 304x. Six of these contigs have average coverages of at least 200x; ten have average coverages of at least 150x; and 18 have average coverages of at least 100x. Although there is some ambiguity in what exactly to set the minimum average coverage as, we lowered $minCov$ to 100x. This resulted in the automatic selection of contig `s92.ctg000105c` (length 1.90 Mbp, average coverage 293x) as the decoy.

The selection of a decoy contig implies FDR estimates for the naïve p-mutation identifications of all other contigs in the dataset, for various values of p. Using the default $highFrequency = 5\%$ parameter to define a p-mutation as indisputable (**"Methods"**), we can draw FDR curves for the "rare" values of $p \in [1.00\%, 4.99\%]$.

Visualizing FDR curves. We present a visualization of eight high-diversity-index target contigs' FDR curves for p-mutation identification in Fig. S6. This is analogous to the FDR curve visualization for SheepGut shown in Fig. 2.

For Fig. 2, we selected target contigs for which to draw FDR curves by considering contigs with high diversity indices (given $p = 0.5\%$). Here, we took a different approach, and instead chose contigs with high numbers of p-mutations per megabase at $p = 1\%$. This is because the default $minReadNumber = 5$ parameter we used in determining "sufficient coverage" when computing the diversity indices (Supplemental Material **"Diversity index details"**) causes many of the diversity indices computed for small values of p to be undefined for the long contigs in ChickenGut, due to these contigs' generally lower coverages. This problem could be circumvented by re-computing diversity indices using a smaller $minReadNumber$ value (we note that this may also result in the selection of a different decoy contig), but for the sake of time we have taken this other approach.

Interestingly, in the SheepGut FDR curve (Fig. 2), the "Full" decoy context yielded relatively low FDR estimates for many target contigs (at many values of p), and the "CP2 & Tv & Nonsense" decoy context yielded relatively high FDR estimates. In Fig. S6 the situation is reversed; this may be a consequence of the lower coverage in ChickenGut resulting in fewer observed cases of these relatively rare types of mutations.

If we use the (relatively conservative) "CP2" decoy context, the best value of p for target contig `s7.ctg000008c` yielding an estimated FDR $\leq 1\%$ is $p = 1.52\%$. At this threshold, we identify 5,899 rare mutations per megabase in this target contig (19,022 rare mutations total), in addition to 503 "indisputable" mutations (p-mutations at $p = highFrequency = 5\%$). This demonstrates that a large amount of diversity lurks within this dataset.

This brief analysis illustrates the applicability of strainFlye to additional HiFi metagenomic datasets: we have shown that our approach for FDR estimation can be adjusted to work for datasets assembled by tools other than metaFlye, and with lower coverage than SheepGut.

Identifying deletion-rich positions in ChickenGut. In addition to drawing FDR curves for ChickenGut, we also analyze its alignment in Supplemental Material **"Coverages and deletion-rich positions"** in order to show that the pattern of "deletion-rich positions" occurs in this dataset as well as in SheepGut.

E Applying LoFreq to the SheepGut dataset

Running LoFreq. To compare LoFreq's calls with those of NaiveFreq, we ran LoFreq (version 2.1.3.1, installed using bioconda) on the three selected MAGs. Since LoFreq-NQ did not seem available in this version of LoFreq, we ran `lofreq call` with default parameters.

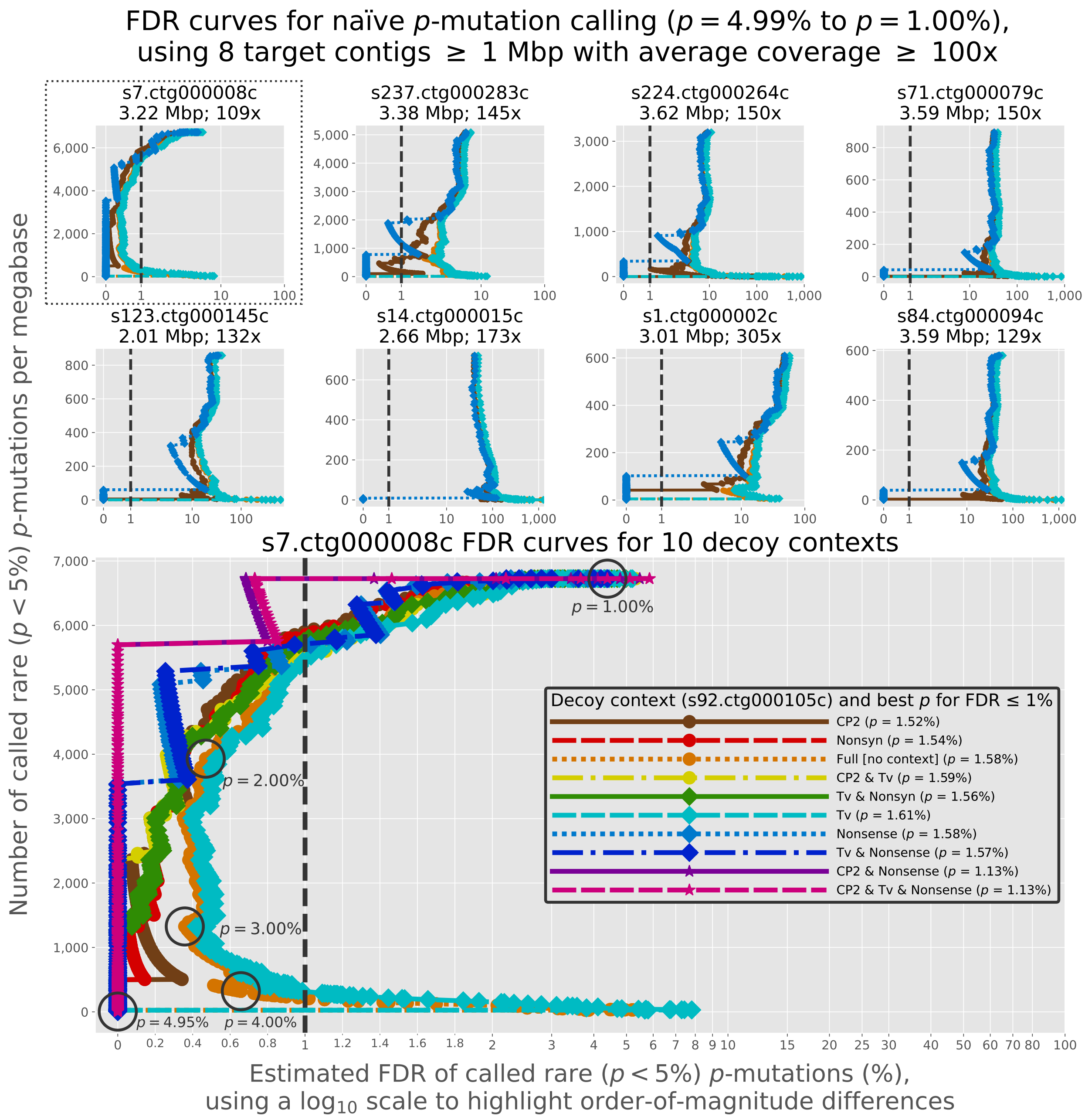


Figure S6: FDR curves for eight target contigs in ChickenGut. We use `s92.ctg000105c` as our decoy contig, and chose the target contigs for this figure as the eight contigs (meeting our length and coverage thresholds) with the largest number of p-mutations per megabase at $p = 1\%$. We draw a larger plot for the target contig `s7.ctg000008c`; we circle certain values of p on the "Full" decoy context curve in this plot for clarity. We note that the FDR curves drawn for `s7.ctg000008c` begin at $p = 4.95\%$ rather than $p = 4.99\%$, because the rare mutation rate of this contig is zero for larger values of p—this causes the estimated FDR for this target contig to be undefined. The colors and other visual representations used here for each decoy context (and combinations thereof) match those used in Fig. 2. In general, the coverages of the "long" contigs in ChickenGut are much smaller than those in SheepGut, resulting in the identification of fewer rare mutations in target and decoy contigs alike.

Running LoFreq on these three MAGs took 474,054.00 seconds (over 5 days and 11 hours); however, we note that there have been recent improvements to LoFreq which, using approximation methods, should decrease this runtime for high-coverage datasets like SheepGut (Kille *et al.*, 2021).

Throughout this Supplement (and in Figs. 3 and S9), we treat positions where LoFreq called multiple single-nucleotide variants (i.e. with distinct alternate nucleotides) as positions with a single $freq(pos)$ value, defining $freq(pos)$ based solely on the second-most-common nucleotide aligned to a position (section **"Computing mutation spectra"**). This simplifies the process of comparing LoFreq and NaiveFreq's results, since NaiveFreq only calls at most one mutation per position. For reference, there exist 0 / 66 / 1 positions at which LoFreq called multiple variants in CAMP / BACT1 / BACT2, respectively; since these account for a small fraction of the total variants LoFreq called in these MAGs (Fig. 3), ignoring them should not make a large difference in the results shown here.

Estimating the FDR of LoFreq's calls. The section **"Estimating the FDR of identified rare mutations using the TDA"** demonstrates how the target-decoy approach (TDA) can be applied to estimate the FDR of a set of mutations; here we illustrate this process using LoFreq outputs. LoFreq called 22, 9,641, and 79 rare mutations in CAMP, BACT1, and BACT2, respectively. At the frequency threshold $p = 2\%$, NaiveFreq called a similar number of rare p-mutations (17, 10,520, and 100 for CAMP, BACT1, and BACT2, respectively). It turns out that the sets of rare mutations identified by LoFreq and by NaiveFreq at $p = 2\%$ are somewhat similar: the numbers of overlapping rare mutations between these groups are 15, 8,033, and 43 for CAMP, BACT1, and BACT2. This suggests that, at least for this dataset, LoFreq primarily detected rare mutations with frequency of at least 2%. Here, we describe an analysis of FDRs which suggests that there exist many more lower-frequency rare mutations.

Using LoFreq's calls, the mutation rates for each MAG are 5.7×10^{-6}, 1.5×10^{-3}, and 9.4×10^{-6} for CAMP, BACT1, and BACT2, respectively. We can estimate the FDR of LoFreq's calls for BACT2 (using CAMP as a decoy) as $\frac{5.7\times10^{-6}}{9.4\times10^{-6}} \approx 60.6\%$, a very large FDR, indicating that either most identified mutations are false or that selection of CAMP as the decoy results in a highly inflated estimate of the FDR. Although NaiveFreq's calls at the frequency threshold $p = 2\%$ result in a lower estimated FDR of $\frac{4.4\times10^{-6}}{1.2\times10^{-5}} \approx 37.0\%$ for BACT2, this is still a high FDR that raises concerns about downstream analyses such as phasing.

On the other hand, the estimated FDR of LoFreq's calls for BACT1 (still using CAMP as a decoy) is only $\frac{5.7\times10^{-6}}{1.5\times10^{-3}} \approx 0.4\%$; NaiveFreq at the frequency threshold of $p = 2\%$ has a slightly lower estimated FDR of $\frac{4.4\times10^{-6}}{1.6\times10^{-3}} \approx 0.3\%$. Although both LoFreq and NaiveFreq at $p = 2\%$ result in the reliable identification of rare mutations with low FDR, we are still interested in extending the set of identified rare mutations while controlling the FDR. For example, lowering the frequency threshold of NaiveFreq to $p = 0.5\%$ results in the

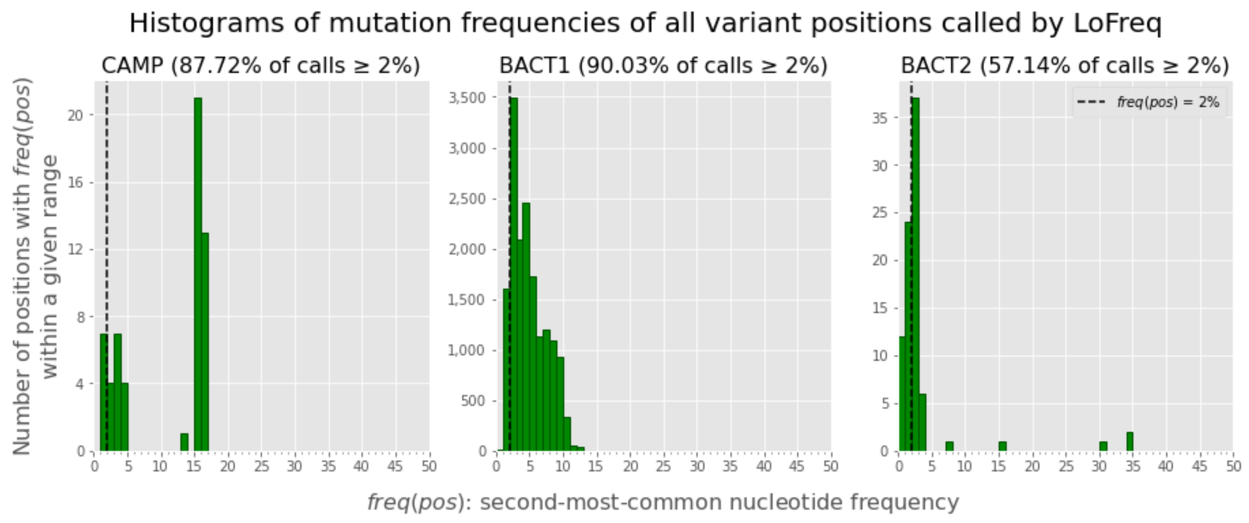


Figure S7: Histograms of $freq(pos)$ for LoFreq's variant calls across the three selected MAGs. These histograms demonstrate that, at least in the context of these MAGs in this dataset, LoFreq seems to mainly call variants with frequency of at least 2%. The titles of each histogram include the percentage of variants with $freq(pos) \geq 2\%$, and each histogram includes a dashed line showing 2%.

identification of 17,069 rare mutations in BACT1 (an additional 6,549 rare mutations as compared to $p = 2\%$) with a higher but still relatively low FDR estimate of 2.4%.

Visualizing mutation frequencies of LoFreq's calls. The above results, in addition to Fig. 3, demonstrate that the variants called by LoFreq are similar to those produced by NaiveFreq at $p = 2\%$. To provide another perspective on this similarity, Figure S7 shows histograms of $freq(pos)$ for all variants called by LoFreq across the three selected MAGs.

F Growth of the number of p-mutations per megabase as p decreases

Fig. 2 demonstrates how we can vary p to produce FDR curves for various contigs. To provide additional context to this analysis, Fig. S8 demonstrates how decreasing p monotonically increases the number of (rare) p-mutations identified in the three selected MAGs.

G Codon position analysis details

Normalized codon position mutation plots. The plots shown in Fig. 3 are useful when identifying relative patterns of mutation rates (for example, that the number of mutations in CP2 in a MAG is less than the number of mutations in CP3); however, since the total

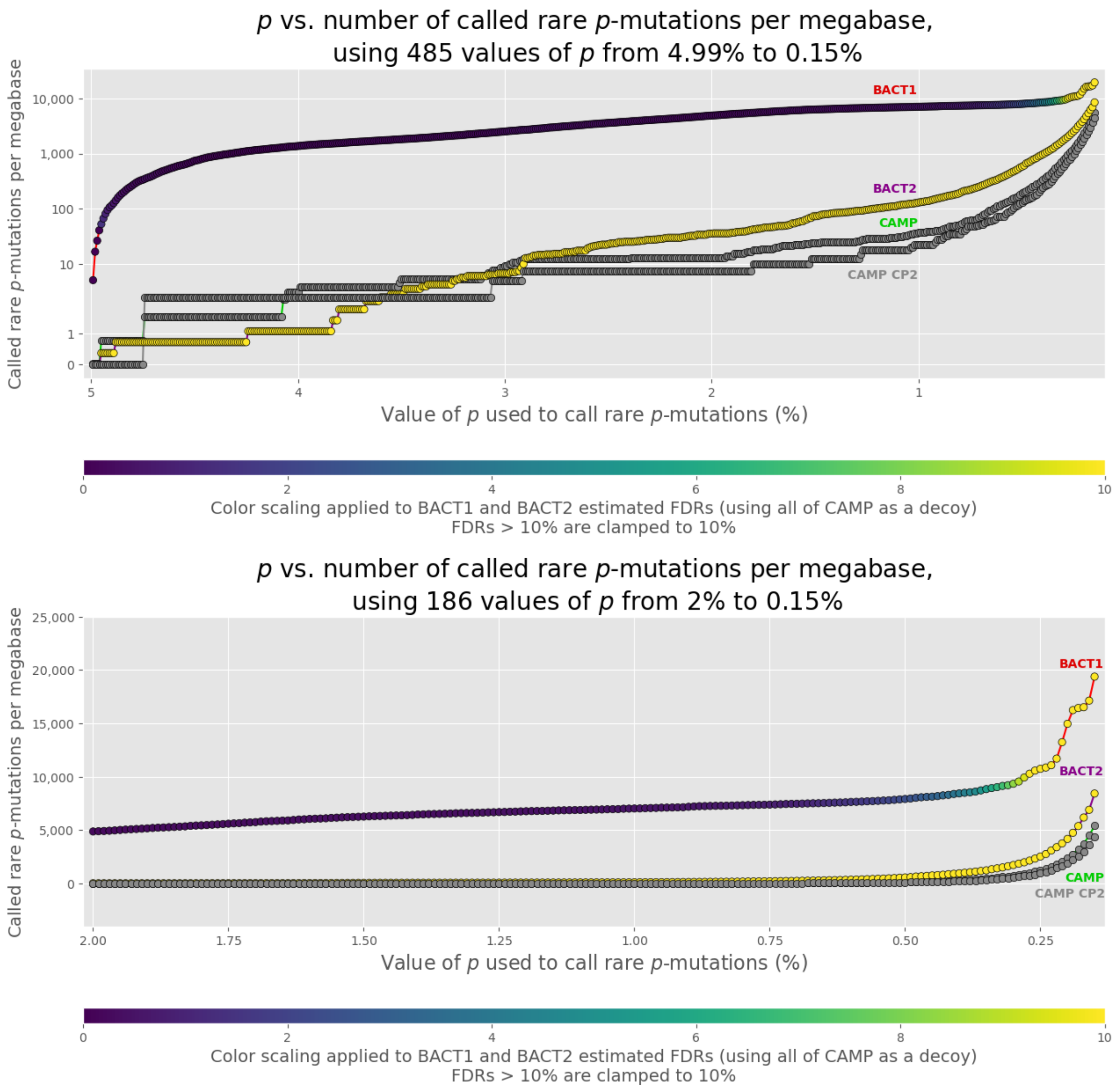


Figure S8: Decreasing p (x-axis) increases the number of identified rare p-mutations per megabase in the three selected MAGs (y-axis). (Top) Visualization of this relationship for 485 values of p, using a logarithmic scale on the y-axis. **(Bottom)** Visualization for a subset of these values of p, using a linear scale on the y-axis. Since BACT1 has the lowest average coverage of 1,415x among the three selected MAGs (Supplemental Material **"Coverages and deletion-rich positions"**), and since we mandate that NaiveFreq only calls mutations at positions with $alt(pos) > 1$, both plots use a minimum p of $\frac{2}{1{,}415} \approx 0.15\%$; this matches the minimum p used in Fig. 2. This plot also includes a curve representing a "decoy contig" composed of just the CP2 positions in CAMP located in a single predicted gene. BACT1 and BACT2's curves are colored based on the estimated FDR for each value of p, using all of CAMP as a decoy contig. We limit FDR color variation to within the range $[0\%, 10\%]$ in order to emphasize small differences in FDRs. The gray dots in BACT2's curve in the bottom figure indicate values of p for which no rare mutations were called, which prevented computation of the FDR due to division by zero.

amount of positions within protein-coding genes varies across MAGs, comparing the literal values between plots from different MAGs is challenging.

We can account for this by dividing each value (the number of single-gene mutated positions in CP1, CP2, and CP3, as well as the number of mutated positions in non-coding regions) by the total number of positions, mutated and non-mutated, fitting these criteria. This transforms the plots so that the y-axis values are now comparable: a y-axis value of 100% for a given CP indicates that all of the single-gene positions in this CP in this MAG were identified as mutations, while a value of 0% indicates that none of the positions in this CP in this MAG were identified as mutations. Fig. S9 is an analogue of Fig. 3 adjusted in this way.

Codon position mutation plot computation details. We also note that our plots only consider each position in a codon in isolation, so "mutations" where multiple positions in a codon change at once might inflate these numbers. For example, as shown in Fig. S12, BACT1 contains a few instances of AGA codons (coding for Arginine) being substituted for CGC (which also codes for Arginine). Although we would currently record this in the figures shown earlier as mutations at both CP1 and CP3, we may instead prefer to discard these mutations entirely. Limiting these analyses to consider codons where the consensus mutated codon differs by a single nucleotide, as in the Supplemental Material **"Constructing and visualizing mutation matrices"**, may be preferable (although these mutations should be relatively uncommon).

Although we have not explicitly accounted for the possibility that some of the predicted genes we consider are nonfunctional pseudogenes that do not currently experience selective pressure (Balakirev & Ayala, 2003), our clear recapitulation of these expected patterns demonstrates that these trends hold in practice.

H Nonsynonymous, nonsense, and transversion decoy contexts

From a decoy of positions to a decoy of mutations. We consider all positions occurring within a predicted protein-coding gene (in CP1/CP2/CP3), ignoring positions that are located within multiple predicted genes due to gene overlap. An arbitrary position i has three possible mutations into another nucleotide, based on this position's CP (three possibilities) and its parent codon (64 possibilities). We note this paper assumes use of the standard genetic code.

For an arbitrary position i, S_i of these three mutations are synonymous and N_i of these three mutations are nonsynonymous. For example, CP3 in the Lysine-coding codon AAA has $S_i = 1$ (AAG still codes for Lysine) and $N_i = 2$ (AAT and AAC both code for Asparagine).

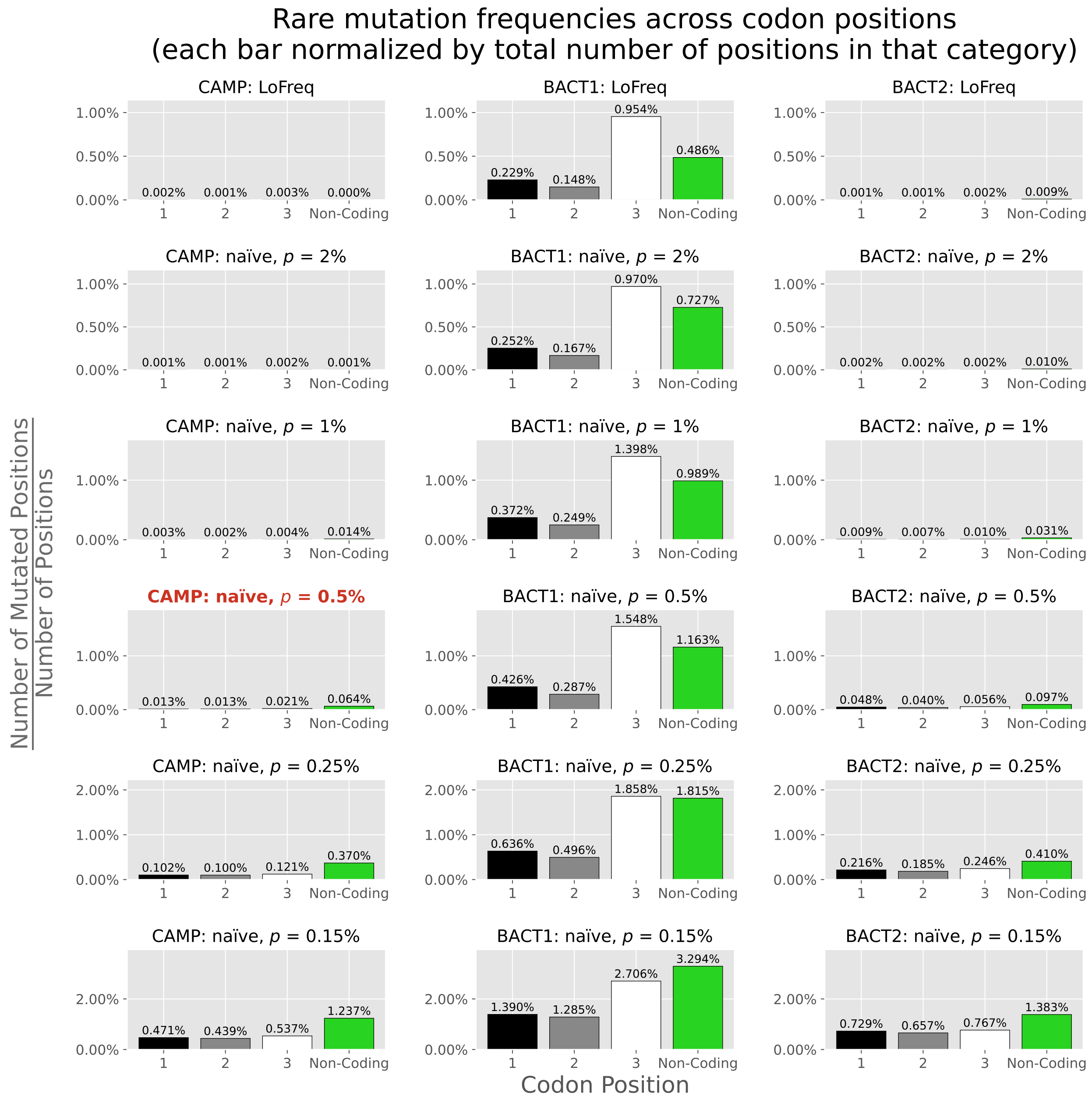


Figure S9: Rare mutation frequencies across all first, second, and third codon positions (as well as non-coding positions) for each of the three MAGs, using the same variant calling methods as in Fig. 3 and with y-axis values normalized by the total number of positions considered in each bar. Although the relative patterns are the same as in Fig. 3, y-axis values are now comparable across MAGs: this makes clear that BACT1 has relatively more mutations than either of the other two MAGs.

Similarly, if we limit our focus to positions i within the 61 sense codons (ignoring the three stop codons TAA, TAG, and TGA), we can define the values NNS_i and NS_i, corresponding respectively to non-nonsense and nonsense mutations. For example, CP1 in AAA has $NNS_i = 2$ (neither CAA nor GAA are stop codons) and $NS_i = 1$ (TAA is a stop codon).

Analogously to our computation of mutation rates for more traditional decoy contigs (of either an entire MAG, or just the CP2 positions within a MAG), we can compute a mutation rate for the decoy contig of possible nonsynonymous or nonsense mutations within a MAG.

Computing (non)synonymous mutation rates. We define the rate of nonsynonymous mutations as $R_N = \frac{M_N}{\sum_i N_i}$, where M_N is the total number of mutations observed in the decoy contig that are nonsynonymous with respect to the position's parent codon and $\sum_i N_i$ is the sum of N_i across all candidate mutation positions i located within the decoy contig. R_N can be used in place of $rate_{decoy}$ when estimating the FDR of identified mutations.

For comparison, we can compute a corresponding rate for synonymous mutations as $R_S = \frac{M_S}{\sum_i S_i}$. Like R_N, R_S is interpretable as a ratio of observed to possible mutations. Fig. S10 shows plots of R_S versus R_N, illustrating that generally $R_S > R_N$.

We note that the R_S and R_N values bear resemblance to the commonly used K_a and K_s values (Hurst, 2002); in computing R_S and R_N, we have attempted to adapt these ideas to the aggregate analysis of different strains of a MAG, rather than the analysis of distinct homologous genes.

Computing (non)sense mutation rates. We define R_{NNS} and R_{NS} as ratios of observed to possible non-nonsense and nonsense single-nucleotide mutations, computed analogously to R_S and R_N (albeit limited to positions located within the 61 sense codons). These values can be interpreted similarly to R_S and R_N, and are also visualized in Fig. S10. We expect to see more non-nonsense than nonsense codon mutations in real sequencing data, since nonsense mutations are usually harmful: and this is generally the case, although this pattern weakens as we reduce p.

Computing transversion mutation rates. As above, consider an arbitrary position i in a contig (not necessarily located within a predicted gene). The nucleotide located at i has three possible mutations into another nucleotide: of these three possible mutations, two are *transversions* (mutations from a purine into a pyrimidine, or vice versa) and one is a *transition* (a mutation from one purine into another, or from one pyrimidine into another) (Vogel, 1972). Although there are twice as many possible transversion mutations as transition mutations, transition mutations generally occur more often than transversions (Vogel, 1972; Kimura, 1980), to the point that variant callers' outputs are often evaluated using the ratio of transitions to transversions (Wang *et al.*, 2015; Wei *et al.*, 2011). Transversion mutations are thus an attractive option for constructing a decoy context, similar to nonsynonymous or nonsense mutations. We can compute the ratio of observed to possible transversion mutations analogously to how we compute R_N and R_{NS} above.

Combining decoy contexts. Fig. 2 demonstrates decoy contexts using CP2, nonsynonymous, nonsense, and transversion mutations, as well as various combinations of these contexts. In total, this figure shows ten distinct FDR curves: here we provide some details about the combinations of decoy contexts that these curves represent.

These combinations can be thought of as intersections. The decoy context of "CP2" and "Transversion," for example, will consider only possible transversion mutations located within CP2 positions. So, although the decoy context of just transversion mutations considers all positions in the decoy contig, the combination of "CP2" and "Transversion" implies focusing solely on positions located within CP2 of a single predicted gene.

If we consider a set containing the entries {"CP2", "Transversion", "Nonsynonymous", "Nonsense"}, the resulting power set has $2^4 = 16$ subsets, or combinations of these contexts. However, not all of these subsets are distinct from each other in practice: for example, all nonsense mutations are also nonsynonymous, so we can ignore combinations that include both "Nonsynonymous" and "Nonsense."

We can also ignore {"CP2", "Transversion", "Nonsynonymous"} because it is identical to {"CP2", "Transversion"}. In the standard genetic code, there is only one possible synonymous CP2 mutation (TAA $\longleftrightarrow$ TGA, both of which are stop codons): since A $\longleftrightarrow$ G is a transition mutation, the addition of the "Nonsynonymous" context to {"CP2", "Transversion"} does not change anything.

We are now left with eleven distinct decoy contexts and combinations thereof. The `strainFlye fdr estimate` command can produce up to eleven FDR curves, using any of these contexts. Although strainFlye supports computing it, we note that one of these eleven combinations ({"CP2", "Nonsynonymous"}) is omitted from Fig. 2. This is because—although this decoy context is technically different from the decoy context of just "CP2"—it is very difficult to visually distinguish these two contexts on a plot, since they only differ in whether or not they count the aforementioned TAA $\longleftrightarrow$ TGA mutation.

Details in the use of nonsynonymous, nonsense, and transversion decoy contexts. We ignore positions that are "unreasonable" (discussed in Supplemental Material **"Hotspot genes in the three selected MAGs"**), to simplify our interpretation of mutations. For example, when we identify nonsense mutations, we assume directionality in this mutation—i.e. that the mutation goes from a sense codon to a stop codon, and not the other way. The "Full" and CP2 decoy contigs do allow unreasonable positions; however, there are in total only 169 unreasonable positions across all three selected MAGs (for reference, 155 / 169 of these are in BACT1), so our exclusion or inclusion of them should not make much of a difference in our estimated FDRs. Anecdotally, these positions seem to be mostly artifacts of low-coverage regions.

When computing nonsynonymous and nonsense mutation rates, we note that our consideration of the positions within a codon in isolation ignores the possibility of multiple positions within a codon being mutated simultaneously—for example, if CP1 and CP2 of the codon AGT (coding for Serine) are mutated into the codon TCT, these mutations are ultimately synonymous since TCT still codes for Serine. However, viewed individually, neither of these two mutations would be treated as synonymous, since neither TGT (Cysteine) nor ACT (Threonine) codes for Serine. This is a limitation of our approach.

Relatedly, we do not explicitly account for alternative start codons. If a predicted gene begins with a codon like TTG or GTG, then we will treat this codon as if it coded for Leucine or Valine, respectively.

Multiallelic positions also pose a challenge, because NaiveFreq only performs "binary" mutation calling: it only classifies a position as mutated or not, based on a single alternate nucleotide. It thus cannot call multiple mutations (into multiple alternate nucleotides) at a single position, although more sophisticated tools like LoFreq (Wilm *et al.*, 2012) can do this. This means that these mutation rates may be slightly lower than they would be if we had accounted for multiple mutations at the same position. (However, as discussed in Supplemental Material **"Applying LoFreq to the SheepGut dataset"**, these sorts of mutations should be relatively rare.)

Lastly, across all $61 \cdot 3 = 183$ positions i in CP1, CP2, or CP3 of the 61 sense codons, there are $\sum_i NNS_i = 526$ non-nonsense mutations and only $\sum_i NS_i = 23$ nonsense mutations (Mort *et al.*, 2008). We emphasize that, since the number of nonsense mutations is much smaller than the number of non-nonsense mutations, the use of nonsense mutations for FDR estimation should be done with caution. For example, although a decoy contig that happens to have no nonsense mutations at all may imply a FDR estimate of 0%, this is not necessarily a useful estimate.

Barplots of (non)synonymous and (non)sense mutation rates. Fig. S10 shows plots of the rates of synonymous vs. nonsynonymous p-mutations and the rates of non-nonsense vs. nonsense p-mutations as p decreases. This figure is analogous to Fig. 3 in the section **"The target-decoy approach for estimating the FDR of identified mutations"**: both figures show how expected mutational patterns mostly remain constant across the three selected MAGs as we adjust p, with some exceptions.

I Identifying mutations based solely on read counts

As an alternative to p-mutation calling, given a *read count threshold* $r \geq 1$, NaiveFreq classifies *pos* as an *r-mutation* if $alt(pos) \geq r$. This is implemented in the `strainFlye call`

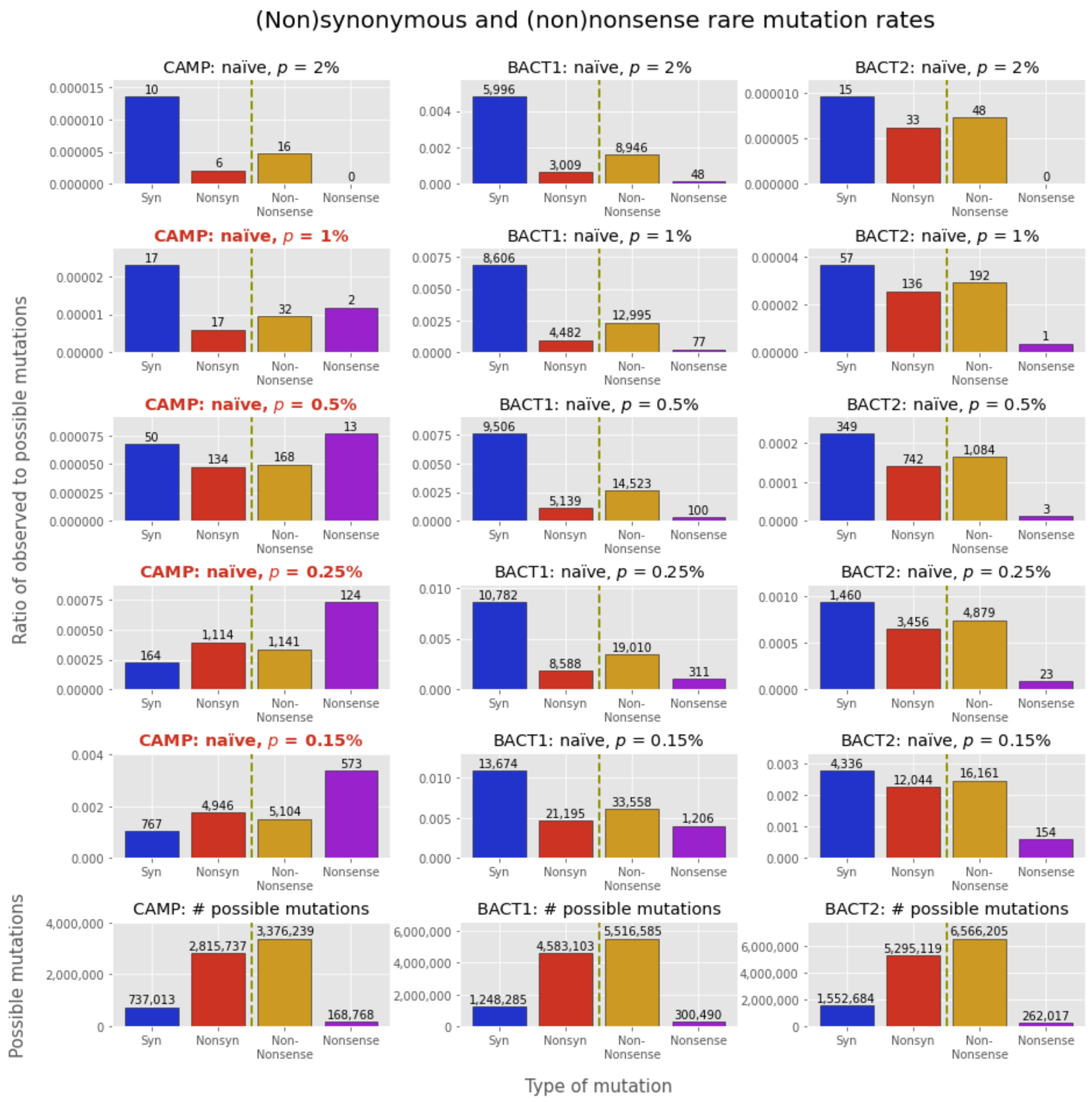


Figure S10: Barplots of R_S, R_N, R_{NNS}, and R_{NS} for various values of p. Since these values are all ratios of observed to possible mutations in a MAG, these ratios increase monotonically as p decreases (causing NaiveFreq to call more p-mutations). The text above the bars for all rows above the bottom row indicates the number of called mutations of each type for each MAG at each value of p; the bottom row shows the number of possible mutations of each type for each MAG. Notably, the relative distributions of possible synonymous vs. nonsynonymous and non-nonsense vs. nonsense mutations are very similar across the three selected MAGs. The ratio of R_S to R_N, and the ratio of R_{NNS} to R_{NS}, also vary with p. Due to selection, we would generally expect $R_S > R_N$ (i.e. relatively more synonymous than nonsynonymous mutations), and we would also expect $R_{NNS} > R_{NS}$ (more non-nonsense than nonsense mutations). The titles of plots where either of these expected patterns do not hold are highlighted in red, like in Fig. 3. This is the case for CAMP for $p \in \{1\%, 0.5\%, 0.25\%, 0.15\%\}$.

`r-mutation` command, which functions analogously to the `strainFlye call p-mutation` command discussed in the main text.

It is sometimes simpler to think in terms of counts than in frequencies, and in this sense r-mutations can have some use. The set of all r-mutations for a large value of r such as $r = 100$ provides, for example, an easy-to-interpret set of usually indisputable mutations (albeit one that will almost certainly miss many real rarer mutations).

Though r-mutation identification can be useful in certain circumstances, we generally caution against using r-mutations in the context of our FDR estimation methods. r-mutation identification does not explicitly account for coverage: contigs with higher coverages can accumulate many more r-mutations at a given r simply by chance. (For example, if a contig has coverage 1,000,000x, then its set of r-mutations at $r = 100$ becomes much less convincing.) p-mutations, while not perfect, adjust the "burden of proof" needed to call a p-mutation at a position based on this position's coverage—this mitigates the problem somewhat.

J Constructing and visualizing mutation matrices

Here we describe the construction and visualization of codon and amino acid mutation matrices for a given MAG, as discussed in the section **"Codon and amino acid mutation matrices"**. We provide the `strainFlye matrix` module for the construction of these matrices. Similarly to (Verbist *et al.*, 2015), we define a given codon as mutated or not based on considering the frequencies of 3-mers that span this codon in the alignment: this is analogous to the process of classifying individual positions in a MAG as p-mutated or not, as discussed in the section **"Computing mutation spectra"**.

Identifying codon p-mutations. Given a threshold $p \in (0\%, 50\%]$, we classify a codon as p-mutated analogously to how NaiveFreq classifies a position as p-mutated: in the case of a codon, we consider all three-nucleotide sequences aligned to this codon's three positions in a MAG.

We consider each codon within each predicted gene within a MAG, allowing the consideration of the same position(s) multiple times if they are present in overlapping genes. (Although we considered limiting these matrices to codons only present in a single gene, we elected not to do this because this might bias our analyses against the inclusion of codons located near the ends of genes—such as Stop codons.) We consider reads aligned to the MAG that cover all three positions in this codon within a single linear alignment, where every position in the read's alignment to this codon is a match or a mismatch operation. A read's alignment spells a *mutated 3-mer* in this codon if the read's sequence aligned to the codon differs from the reference codon sequence. For each codon, we compute all matched and mismatched (mutated) 3-mers aligned to the codon from covering read alignments.

After computing this information for every codon, we then ignore codons where the most common aligned 3-mer (or, in the case of a tie, any of the most common aligned 3-mers) at this codon is not exactly the same as the reference codon's 3-mer. This check allows us to ignore low-coverage or otherwise difficult-to-interpret codons where it is unclear how to define a "mutation" from this codon into another, and is analogous to how we also ignore "unreasonable" positions in other analyses in this paper. We implicitly make the assumption that these mutations occur *from* the reference codon.

For all codons that pass the above check, we then classify the codon as mutated or not similarly to how we classified positions as mutated in the section **"Computing mutation spectra"**: if the maximum-frequency mutated 3-mer aligned to the codon has a relative frequency (analogous to $freq(pos)$) greater than or equal to some threshold p and an absolute frequency (analogous to $alt(pos)$) greater than 1, we define the codon as p-mutated. Unlike our use of NaiveFreq in FDR estimation, we consider both rare and indisputable p-mutated codons.

Our exclusion of codons where the reference 3-mer is not also the most common aligned 3-mer means that, as with the definition of $freq(pos)$ in section **"Computing mutation spectra"**, the relative frequency of any p-mutated codon is constrained to the range of $(0\%, 50\%]$.

Once we identify codon mutations, we can then compute codon mutation frequencies and construct a mutation matrix.

Computing codon mutation frequencies. Using these identified codon mutations, we compute the total number of (X, Y)-mutations (the number of times codon X mutates into a different codon Y) and the total number of X-mutations (the number of times X mutates into any of the other 63 codons, including Y). We define the *codon mutation frequency* of X into Y as the number of (X, Y)-mutations divided by the number of X mutations.

Visualizing codon mutation frequencies. Fig. S11 is a 64x64 *codon mutation matrix* representing all single-nucleotide codon mutation frequencies throughout BACT1, using the threshold $p = 0.5\%$ to naïvely call codons as p-mutated. Fig. S11 reveals large variations in the mutation rates of various codons encoding the same amino acid, indicating that codon-based matrices may be more sensitive for strain comparison. For example, the Glycine-encoding codons GGA and GGG have mutation rates 3% and 11%, respectively. Interestingly, our analysis reveals that rare codons typically have much higher mutation rates than frequent codons for the same amino acid.

Fig. S12 shows "full" codon mutation matrices. The main differences between these matrices and the matrix shown in Fig. S11 are the inclusion of all codon mutations in each row (not just the 9 single-nucleotide mutations from a given codon); the representation of mutations in the matrix and *Syn* column as raw numbers, rather than percentages; the

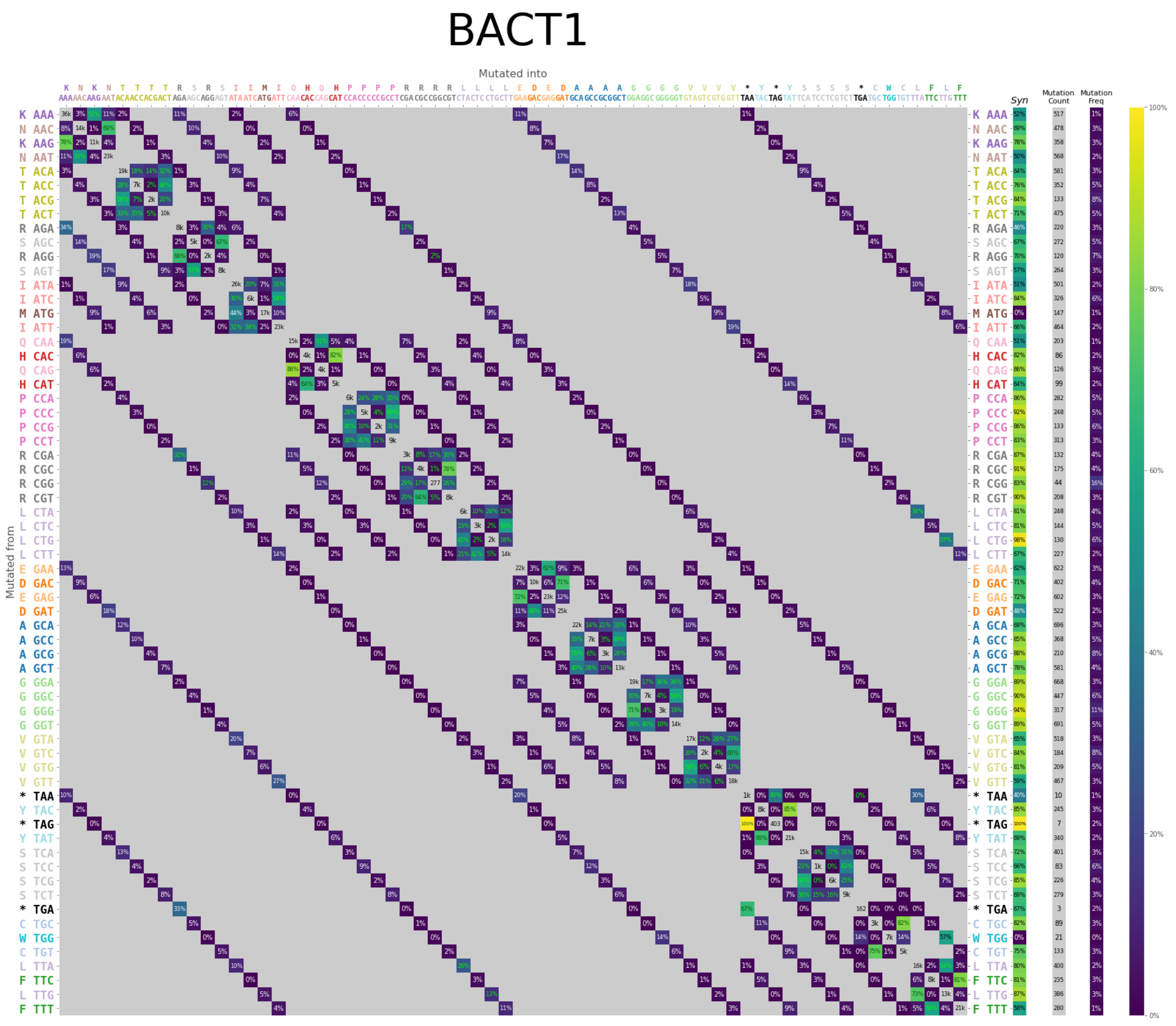


Figure S11: The 64x64 single-nucleotide codon mutation matrix for BACT1 (using the frequency threshold $p = 0.5\%$). The numbers on the main diagonal show the number of times a codon occurs in all genes in the MAG, rounded to the nearest thousand (for codons that occur less than 1,000 times, the raw number is shown). Percentages off of the main diagonal show the fraction of times a given mutation from codon X into codon Y was observed, relative to the total number of single-nucleotide codon mutations from codon X in this MAG. The sum of the non-diagonal entries in each row of the matrix, then, should be approximately 100%. The text fonts representing numbers in the non-diagonal cells of the matrix are colored white or black if the mutation represented in this cell is nonsynonymous, and light green or dark green if this mutation represented in this cell is synonymous. The labels of the matrix are annotated with both the codon sequence and the corresponding amino acid / stop codon translation, abbreviated as a single letter (Cornish-Bowden, 1985). The column labelled *Syn* shows the sums of synonymous single-nucleotide mutation percentages across each row in the matrix (these sums are also included as text within these columns). The column labelled "Mutation Count" shows the total number of mutations of this row's codon in BACT1's genes (*including both single-nucleotide and non-single-nucleotide codon mutations*). The column labelled "Mutation Freq" shows "Mutation Count" for each codon's row divided by the total number of occurrences of this codon in BACT1's genes. The gradient on the right is a legend mapping percentages to colors in the viridis colormap (Smith & van der Walt, 2015). The shown matrix only reflects single-nucleotide codon mutations; however, a "full" version of this matrix is shown in Fig. S12.

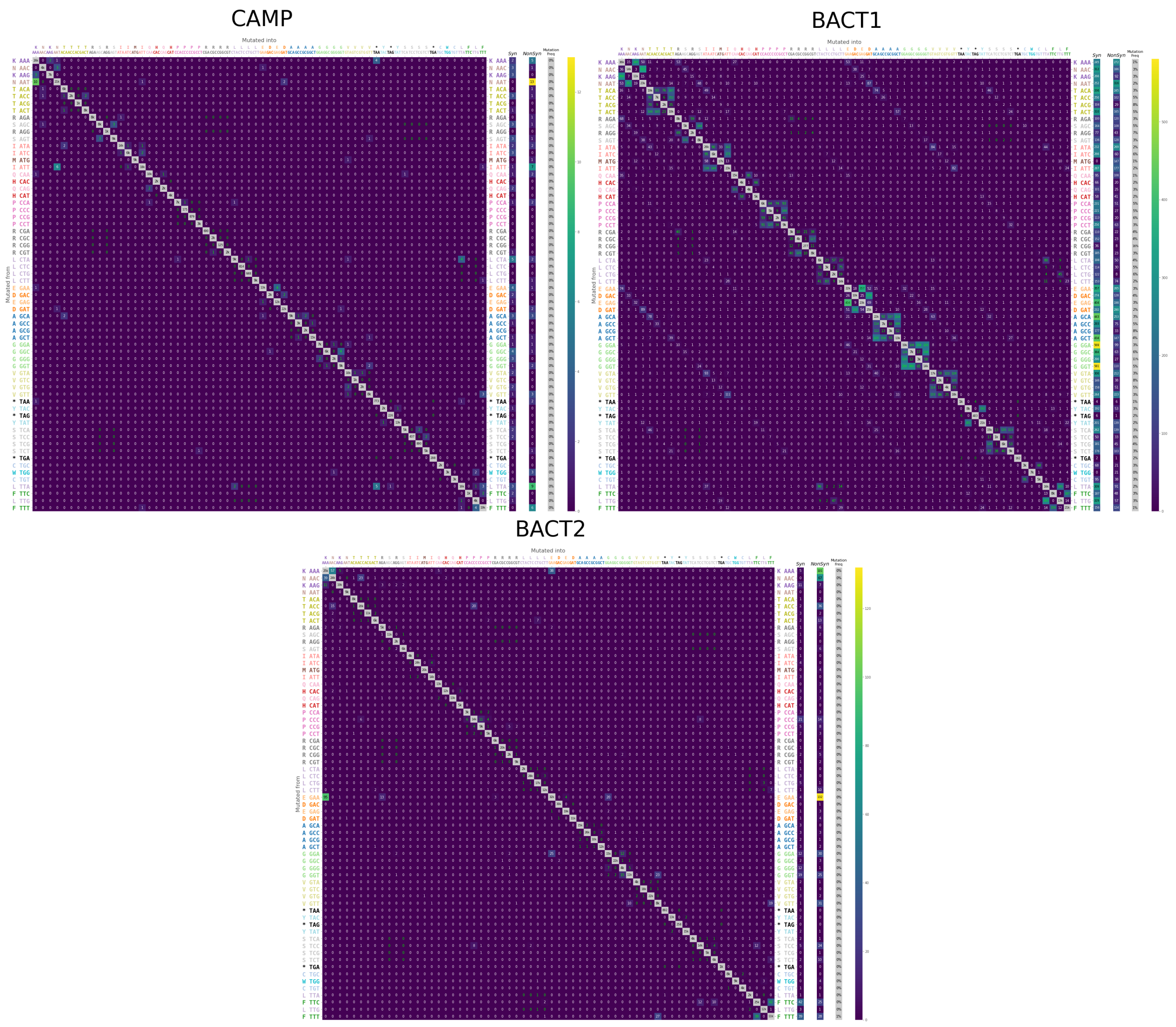
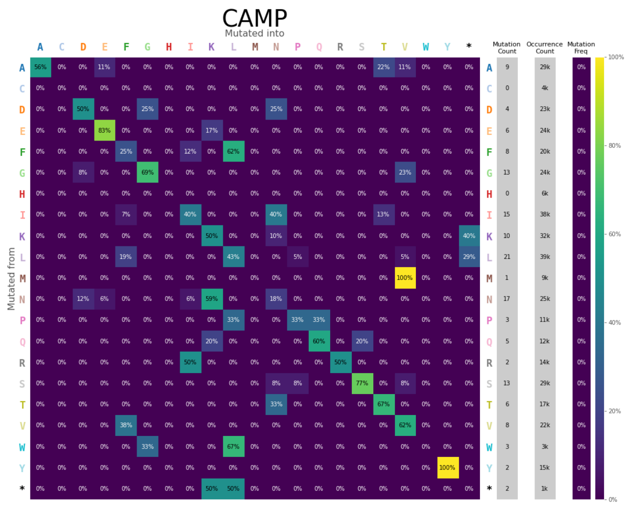
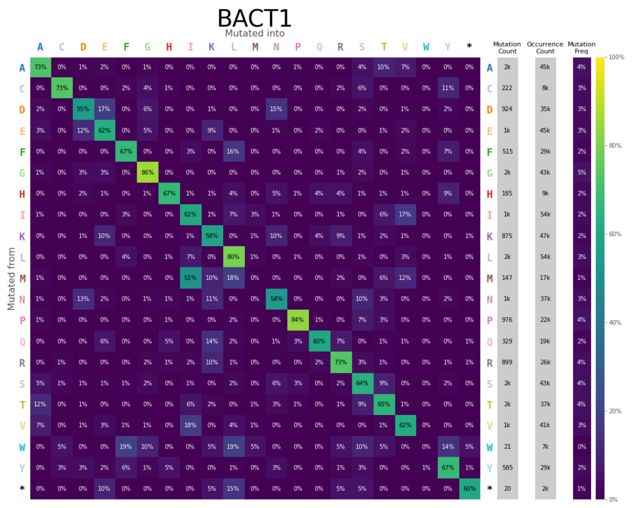


Figure S12: "Full" codon mutation matrices for the three selected MAGs (using the frequency threshold $p = 0.5\%$).

presence of a *NonSyn* column instead of the "Mutation Count" column; and the lack of color in the codon mutation rate column (since coloring percentages using the same scale by which the raw numbers are colored is an "apples-to-oranges" comparison).

Visualizing amino acid mutation counts. Fig. S13 shows the 21x21 amino acid / stop codon mutation matrices for each of the three selected MAGs, derived from their corresponding codon mutation matrices shown in Fig. S12. As with Supplemental Material **"Nonsynonymous, nonsense, and transversion decoy contexts"**, we do not account for alternative start codons.





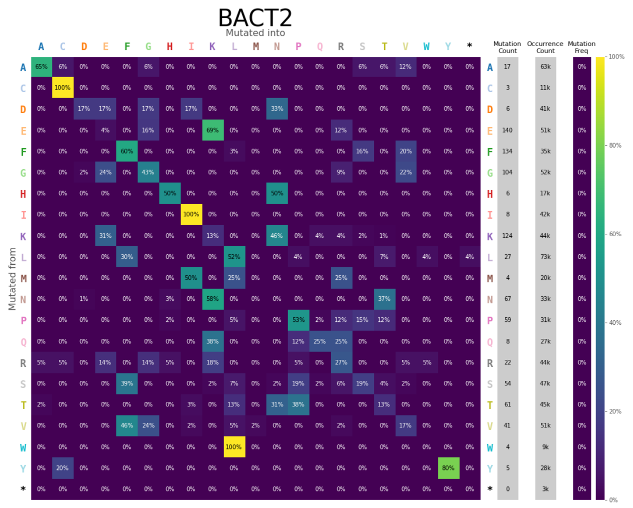


Figure S13: The 21x21 amino acid / stop codon mutation matrices derived from codon mutation data for the three selected MAGs (using the frequency threshold $p = 0.5\%$). Percentages in each matrix show the number of times a given mutation from one amino acid / stop codon into another was observed based on the codon mutation data (including both single-nucleotide and non-single-nucleotide codon mutations), divided by the total number of mutations (both synonymous and nonsynonymous) seen from this row's amino acid / stop codon. The denominators of these percentages for each row are represented in the "Mutation Count" column. The numbers in the "Occurrence Count" columns show the number of times an amino acid / stop codon occurs in all genes in a MAG. The numbers in the "Mutation Freq" columns show "Mutation Count" divided by "Occurrence Count", analogously to the "Mutation Freq" column in Fig. S12. Numbers are colored white or black arbitrarily, in order to improve legibility on dark or light cells. Synonymous codon mutations (e.g. GCA into GCC, both of which code for Alanine) are represented as values on the main diagonals. Amino acids are referred to by their single-letter abbreviations (Cornish-Bowden, 1985), with the $*$ character (located in the final row and column of each matrix) representing stop codons. The gradients on the right are legends mapping percentages to colors in the viridis colormap (Smith & van der Walt, 2015).

K Diversity index details

Here we provide further information on diversity indices introduced in the section **"Diversity indices"**.

Defining "sufficient coverage" for p-mutation calling. Here we assume that strains of a given organism are sequenced exhaustively, and that the alignments of all reads are correct. Although these assumptions may be broken in practice (McLaren *et al.*, 2019), they should be acceptable in the context of high-coverage HiFi datasets. Given these assumptions we would expect to see, for each strain with population frequency p, $p \cdot C$ copies of this strain's sequence at a given position in the alignment (where C is the coverage at the position). For example, a strain with population frequency $p = 25\%$ would on average be represented at 25x coverage at a given position in the alignment with coverage $C = 100$x; similarly, a strain with population frequency $p = 0.8\%$ would on average be represented at 8x coverage at a position with coverage $C = 1{,}000$x.

We thus define the minimum sufficient coverage for some p as $minSuffCov = \frac{minReadNumber}{p}$, where $minReadNumber$ (default value 5) is a small positive parameter and p is represented as a number in the range $(0, 0.5]$ rather than as a percentage. That is, if we want to find mutations with a given population frequency using solely the sequencing data at hand, without taking into account further biological insights about error rates, we need to have sufficient sequencing coverage to where we would expect to observe $minReadNumber$ copies of these mutations (Töpfer, 2017). This relation between $minSuffCov$ and p makes clear that the burden of proof for calling a relatively "common" mutation (e.g. $p = 50\%$) is much lower than the burden of proof for calling a relatively "rare" mutation (e.g. $p = 0.5\%$).

Defining "sufficient coverage" for r-mutation calling. Supplemental Material **"Identifying mutations based solely on read counts"** details the identification of r-mutations, which are called using read counts rather than frequencies. There is less need to impose an additional requirement of sufficient coverage for these particular mutations.

However, knowledge of which positions are or are not sufficiently-covered has other uses—for example, as shown below, when defining entire contigs as sufficiently-covered or not. We thus define a position as sufficiently-covered to call a given r-mutation using a quantity named $minCovFactor$ (default value 2): the minimum sufficient coverage for some r is defined as $minSuffCov = minCovFactor \cdot r$. Since r corresponds to the count of the second-most-common nucleotide at a position, the default value of $minCovFactor = 2$ means that this condition will always hold if we have called an r-mutation in the first place (although it will also hold for many positions at which there exist no r-mutations).

Computing the diversity index for sufficiently-covered contigs. We have shown above how to compute the minimum sufficient coverage for calling p- or r-mutations for arbitrary positions in a contig, when computing diversity indices. We (somewhat crudely) define a contig G itself as sufficiently-covered if at least 50% of its positions are sufficiently-covered.

We define the diversity index for a sufficiently-covered contig G as the number of called p- or r- mutations in the sufficiently-covered positions in G, divided by the total number of sufficiently-covered positions in G. If a contig is not sufficiently-covered, we do not define a diversity index for it.

Information about high-diversity-index edges. Fig. 4 reveals a number of edges in SheepGut with high diversity indices. Here we provide further information about these edges in order to help in the interpretation of this figure.

For reference, the most diverse edge at $p = 50\%$ is edge 3091 in the graph, which has a length of 1 Mbp and a coverage (reported by metaFlye) of 40x. Kaiju (Menzel *et al.*, 2016) classified this edge as *Dorea formicigenerans*.

The most diverse edge at $p = 25\%$ is edge 15931, which has a length of 1.1 Mbp and a coverage of 33x. Kaiju classified this edge as an uncultured *Flavonifractor* sp.

The most diverse edge for $p \in \{10\%, 5\%, 2\%, 1\%, 0.5\%\}$ is edge 3030, which has a length of 1.9 Mbp and a coverage of 1,011x. Kaiju classified this edge in the phylum *Firmicutes*.

The most diverse edge at $p = 0.25\%$ is edge 23917, which has a length of 1 Mbp and a coverage of 1,973x. Kaiju classified this edge as *Phocaeicola vulgatus*.

Although edges 3030 and 23917 both have very high coverage, neither were selected for the analyses in the rest of this paper because these edges were located in the largest "hairball" component of the assembly graph. (Edges 3091 and 15931 are also located in the "hairball," for reference.)

L Hotspot genes in the three selected MAGs

For each gene in each MAG, we can compute values for each gene describing the mutations observed for the positions in this gene. These values include the *mutation rate* as well as more specific values such as the *nonsynonymous mutation rate* (defined as the fraction of mutated positions in the gene where the highest-frequency alternate nucleotide causes a nonsynonymous mutation in the position's codon within the gene; more details are described later in this Supplement).

For convenience's sake, here we ignore positions where the reference nucleotide is not also the consensus (i.e. most frequently seen in the alignment at this position) nucleotide. Positions that do not meet this criteria, termed "unreasonable" positions, are not included as mutations when computing the mutation rates shown in Table S2. This simplifies the computation of the nonsynonymous mutation rate, since it becomes easier to say that a given nucleotide represents a mutation “from the reference”; this is analogous to how certain codons are ignored in the Supplemental Material **“Constructing and visualizing mutation matrices”**.

Out of a total of 1,297 / 1,761 / 2,567 genes in CAMP / BACT1 / BACT2, respectively, 110 / 1,511 / 677 genes have nonzero p-mutation rates (using the threshold $p = 0.5\%$). Of these genes, 98 / 1,171 / 566 genes have nonzero nonsynonymous mutation rates. (The rounded average gene lengths for each MAG are 919 / 1,106 / 897 bp, respectively.) To assist in identifying specific genes of interest, Table S2 presents information about the ten genes with the highest mutation rates for each of the selected MAGs. Given this information, we can now investigate the mutation spectra of these genes in an attempt to learn more about the strain(s) of these MAGs present in the dataset, for example as shown in Fig. 5.

Computing the nonsynonymous mutation rate of a gene. Table S2 makes use of the *nonsynonymous mutation rate*, a value we defined for each gene. Here we describe some minor details of the computation of this value.

When computing this metric we consider all mutated positions from all codons in a gene. An example gene containing 99 positions—with two mutated positions, both in the same codon within the gene, and each causing a nonsynonymous mutation by themselves—would have a nonsynonymous mutation rate of $2/99 \approx 2.02\%$.

We note that our definition of a mutation as “nonsynonymous” assumes that the contig in which this mutation occurs follows the standard genetic code, and that we do not explicitly account for alternative start codons. These assumptions match those used in the Supplemental Material sections **“Nonsynonymous, nonsense, and transversion decoy contexts”** and **“Constructing and visualizing mutation matrices”**.

A rare corner-case in this definition occurs when a given mutated position has more than one highest-frequency alternate nucleotide. For example, we could imagine CP3 of the codon CAA. This codon codes for the amino acid Glutamine. If, in CP3 of this codon, 100 aligned reads have an A, 10 aligned reads have a C, 10 aligned reads have a G, and 1 aligned read has a T, then this position will be labelled as a p-mutation for $p = 0.5\%$, since $10/121 \approx 8.26\%$, which is greater than 0.5%. However, it is ambiguous whether or not this mutated position causes a nonsynonymous mutation: C and G are tied as the highest-frequency alternate nucleotide, and CAC causes a nonsynonymous mutation (coding for Histidine) but CAG does not cause a nonsynonymous mutation (coding for Glutamine).

In the case of this sort of tie, the alternate nucleotide selected for determining whether or not this mutation is nonsynonymous or not is arbitrary. However, we expect that this case should generally be rare.

CAMP

Left	Right	Length (bp)	#	Nonsyn. Mutation Rate	Mutation Rate
1,208,927	1,210,075	1,149	1217	0.26%	2.96%
1,053,365	1,055,938	2,574	1056	0.31%	0.97%
1,197,031	1,197,267	237	1205	0.84%	0.84%
1,287,424	1,287,777	354	1295	0.56%	0.56%
890,033	890,569	537	886	0.37%	0.56%
223,065	223,976	912	229	0.55%	0.55%
1,056,052	1,059,237	3,186	1057	0.16%	0.50%
493,154	493,558	405	477	0.49%	0.49%
1,231,190	1,231,402	213	1243	0.00%	0.47%
684,152	684,607	456	674	0.44%	0.44%

BACT1

Left	Right	Length (bp)	#	Nonsyn. Mutation Rate	Mutation Rate
1,041,656	1,042,084	429	868	16.32%	20.05%
206,606	207,304	699	184	12.73%	18.74%
1,402,353	1,402,718	366	1185	9.29%	16.94%
266,551	267,051	501	241	7.78%	15.77%
1,326,288	1,327,073	786	1118	7.00%	15.27%
724,871	725,443	573	586	10.65%	15.18%
1,458,950	1,460,104	1,155	1233	9.09%	15.06%
1,041,043	1,041,654	612	867	9.64%	14.38%
1,156,180	1,157,316	1,137	966	9.15%	14.16%
601,203	602,750	1,548	486	8.07%	13.95%

BACT2

Left	Right	Length (bp)	#	Nonsyn. Mutation Rate	Mutation Rate
2,797,869	2,798,756	888	2561	0.90%	4.84%
2,512,322	2,513,143	822	2288	0.49%	3.77%
1,696,990	1,697,364	375	1575	1.33%	2.13%
2,216,402	2,216,542	141	2030	1.42%	2.13%
2,626,301	2,626,774	474	2420	1.48%	1.90%
177,371	177,700	330	155	1.21%	1.82%
660,412	660,522	111	646	1.80%	1.80%
2,510,205	2,512,007	1,803	2286	0.50%	1.72%
2,101,831	2,102,010	180	1926	1.11%	1.67%
269,131	269,250	120	249	1.67%	1.67%

Table S2: The ten most mutated genes in the three selected MAGs (using the threshold $p = 0.5\%$ to classify a position as mutated or not). The "#" column shows the number assigned to each gene in the SCO files output by Prodigal for each MAG. We sort genes by their total mutation rate (the rightmost column); the mutation rates for the top ten genes in BACT1 are much higher than those from the other two MAGs.

Coverages of highly-mutated genes. Fig. S14 shows the coverages of the highest-mutation-rate genes for each of the three selected MAGs. This figure demonstrates that the coverages these genes are uniformly high, and that the mutation spectra of these genes (shown in Fig. 5 in the main text) are thus reliable.

M Identifying strains in the most mutated gene of BACT1

The most mutated gene in BACT1 (gene 868 of length 429 bp) exhibits a complex pattern of mutations illustrated in Fig. 5 (middle subfigure) with three main levels of mutated positions, at approximately 1.5%, 4.5%, and 6.5%. Below we attempted to group all 1,273 reads spanning this gene into clusters representing putative strains.

We transformed each read into a 429-dimensional binary vector, where the i-th value of a read is set to 1 if this read differs from the BACT1 reference MAG (with a mismatch or deletion/skip) at the i-th position of gene 868, and is set to 0 otherwise. We then performed k-means clustering on these 429-dimensional binary vectors for $k \in [1, 20]$ using scikit-learn (Pedregosa *et al.*, 2011). A plot of the performance of each of these runs of k-means clustering, assessed using scikit-learn's "intertia" measurement, is shown in Fig. S15 (left). This plot contains a clear "elbow" at $k = 4$ after which improvements in classification begin to plateau, indicating that 4 is likely an acceptable number of clusters (Hardy, 1994). This corroborates the mutation spectrum of this gene in Fig. 5 (middle), which contains the aforementioned three rough levels of mutated positions' frequencies (in addition to a fourth level of mostly un-mutated positions). In order to provide an additional view of how these reads' vectors vary, Fig. S15 (right) shows a principal component analysis plot of a matrix of the binary vectors, with points colored by their assigned cluster from the k-means clustering algorithm at $k = 4$. The majority of reads belong to the first cluster, which contains relatively "un-mutated" reads.

We note that the first draft of this analysis did not consider deletions/skips in a read at a given position in the gene to result in a 1 in that read's binary vector at this position (in these cases, the vector would contain a 0 at this position). This version of the analysis resulted in the selection of an "elbow" at $k = 3$ clusters. From manual inspection in IGV (Thorvaldsdóttir *et al.*, 2013), the reads aligned to this gene contain many deletions; this is why we have presented this particular analysis in a way that considers deletions as mutations.

N Plots of mutation locations

Fig. S16 visualizes the locations of mutations in CAMP, BACT1, and BACT2. Although mutations are spread throughout these MAGs, especially for small values of p, there are ob-

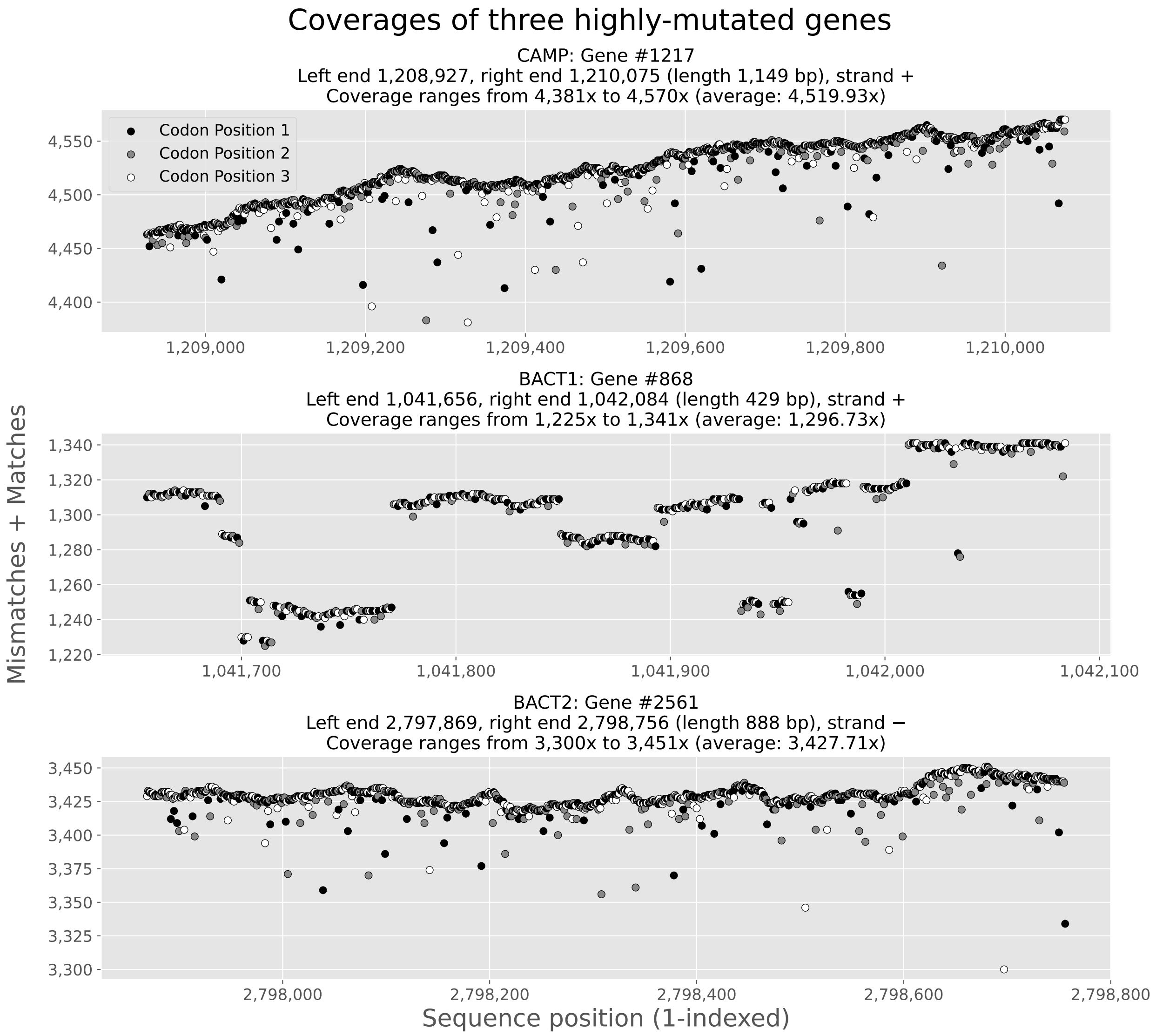


Figure S14: Scatterplot showing $mismatches(pos)+matches(pos)$, also known as coverage, for each position pos within the highest-mutation-rate genes in CAMP, BACT1, and BACT2. As in Fig. 5, positions are colored by their codon position within the parent genes shown here. This plot demonstrates that the coverages of these genes are all high ($\geq$ 1,000x) and somewhat uniform. Strangely, coverage increases slightly over the length of the highest-mutation-rate gene in CAMP; although this may just be a sequencing artifact, it may indicate that this genome is being actively replicated, since such genomes have higher coverage near the origin of replication (Korem *et al.*, 2015). (We thus highlight the location of this gene in Figure S17 in the Supplemental Material **"Growth dynamics"**.) The coverage of the highest-mutation-rate gene in BACT1 contains more dramatic "jumps," although these are primarily due to this plot not counting deletions towards coverage.

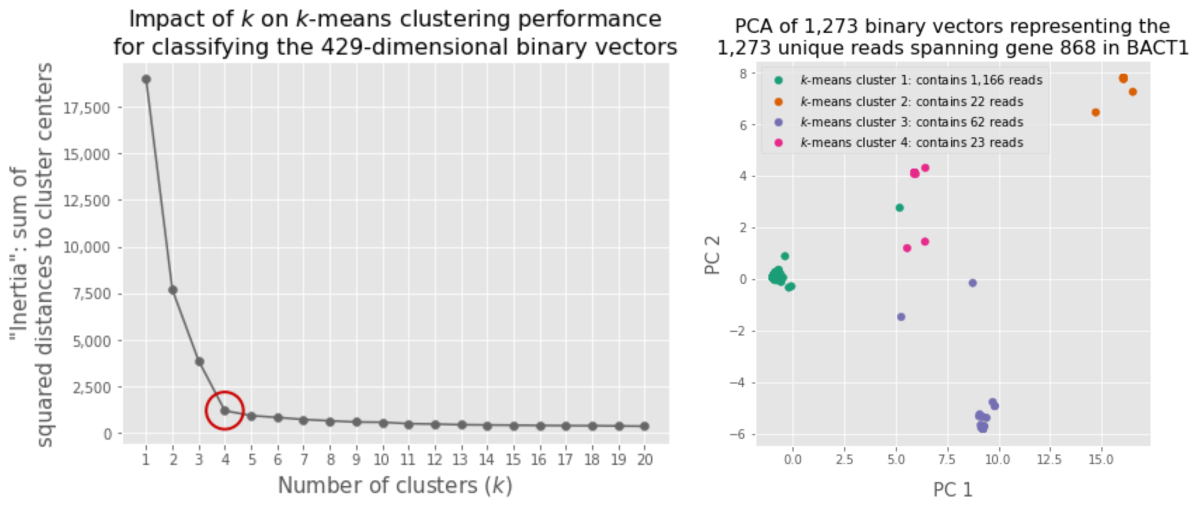


Figure S15: (Left) Plot of k versus k-means clustering performance for the 1,273 429-dimensional binary vectors representing reads spanning gene 868 in BACT1. Per the scikit-learn documentation at `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`, the "inertia" measurement describes the "sum of squared distances of samples to their closest cluster center" (Pedregosa *et al.*, 2011); the use of inertia as the clustering evaluation metric is based on the tutorial given at `https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/`. We highlight $k = 4$ here, which represents a clear "elbow" in this plot. **(Right) Principal component analysis (PCA) plot of the 1,273 429-dimensional binary vectors.** Each vector's point is colored by its assigned cluster from k-means clustering with $k = 4$, using the "Dark2" color map from ColorBrewer (Brewer *et al.*, 2021). The two-dimensional visualization provided by PCA mostly separates the four clusters. We performed PCA using scikit-learn (Pedregosa *et al.*, 2011).

vious coldspots. For example, there are two particularly large regions without any identified p-mutations located near 1.6–1.8 Mbp in BACT1: these regions have no p-mutations even when $p = 0.5\%$, a surprising result. Supplemental Material **"Investigating coldspots"** further investigates the coldspots in BACT1.

Hotspots can also be identified from Fig. S16, although they are less easy than coldspots to distinguish in this visualization. For example, when $p = 10\%$ and when $p = 5\%$, 34 of the 35 p-mutations in CAMP are represented by the single black bar located near the 1.2 Mbp point in the MAG: these 34 mutations, it turns out, are all contained within a single highly-mutated gene in CAMP (shown in Fig. 5).

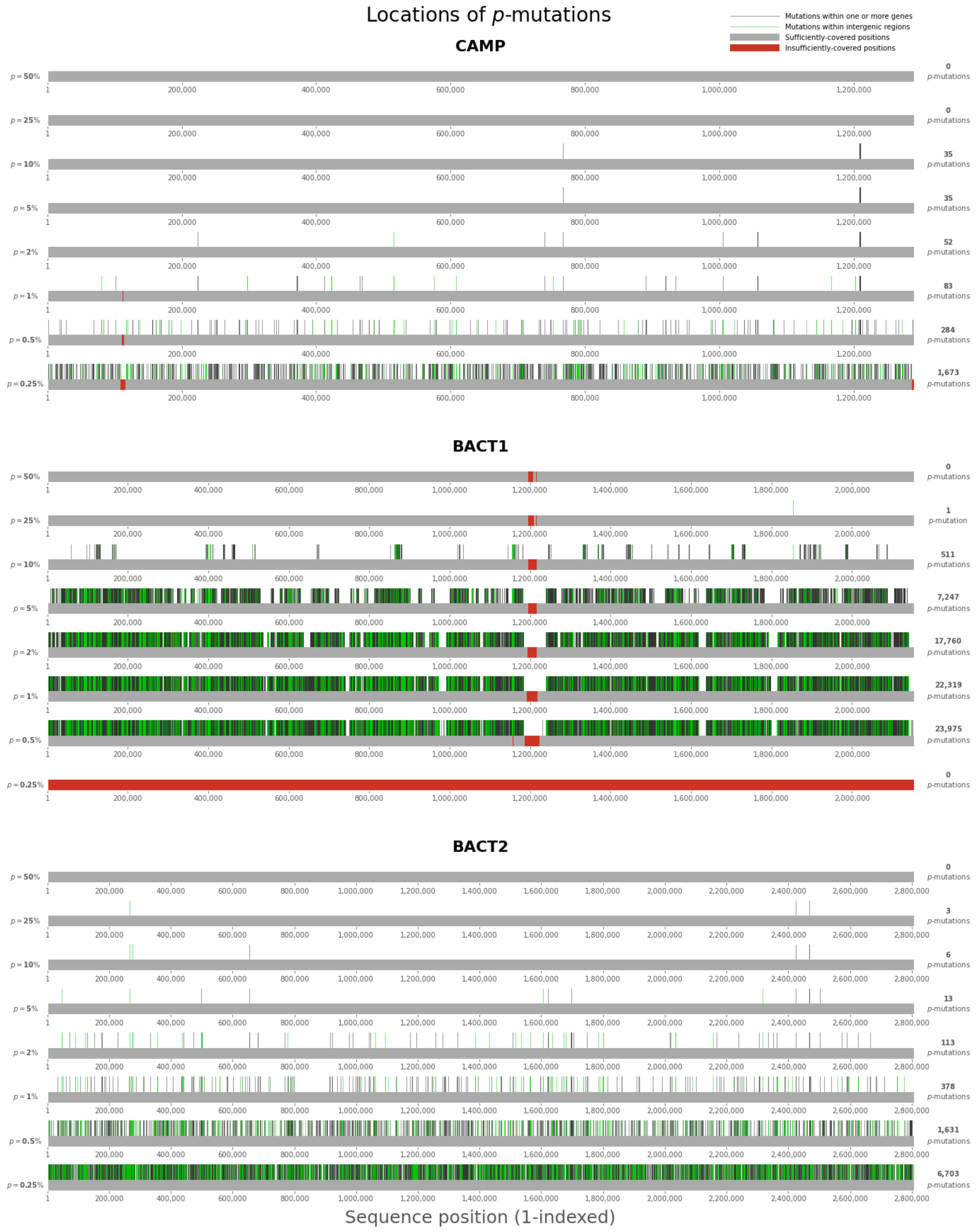
O Investigating coldspots

Taxonomic classifications of long coldspots in BACT1. At $p = 0.5\%$, and without attempting to call p-mutations at positions with coverage less than 1,000x (Fig. S16), BACT1 contains 7 "coldspots" corresponding to regions of length $\geq$ 5,000 bp without any p-mutations. Table S3 describes the location, length, average coverage, and taxonomic classifications of each of these coldspots.

We suspected that some of these coldspots might correspond to conserved sequences (e.g. parts of 16S rRNA genes) that would not vary much across genomes. To check this, we ran barrnap (Seemann, 2018) on BACT1; although barrnap predicted multiple rRNA genes, none of these predicted genes' coordinates intersected with any of the seven coldspots investigated here. The cause of these coldspots is thus still unclear. The two coldspots for which BLASTN identified matches to viral genomes may indicate recent prophage insertions that could justify the lack of mutations in these regions (Manna *et al.*, 2004), although additional work would be needed to prove this.

Evaluating statistical significance of coldspots. Although it is tempting to assume that long "gaps" between mutations in a contig have some biological meaning, gaps between mutations could arise by chance. To provide additional context for the gaps identified in a dataset, the `strainFlye spot cold-gaps` command can compute a simple *p-value* describing the probability of the longest gap in a contig being at least a certain length. The computation of this p-value relies on the null hypothesis that each position in the contig is defined as mutated or not independently, using a fixed contig-specific mutation rate. This assumption reduces the problem of computing this p-value to the classical problem of analyzing the distribution of the length of the longest run of heads in a sequence of coin tosses (Bateman, 1948).

For each contig with at least one mutation and at least one gap meeting a user-set minimum length threshold, we define the *mutation rate* of this contig, q, as the ratio of mutated positions in this contig (m) to the total number of positions in this contig (n)



Locations of p-mutations
Mutations within one or more genes
Mutations within intergenic regions
Sufficiently-covered positions
Insufficiently-covered positions
CAMP
p = 50% 0 p-mutations
p = 25% 0 p-mutations
p = 10% 35 p-mutations
p = 5% 35 p-mutations
p = 2% 52 p-mutations
p = 1% 83 p-mutations
p = 0.5% 284 p-mutations
p = 0.25% 1,673 p-mutations
1 200,000 400,000 600,000 800,000 1,000,000 1,200,000
BACT1
p = 50% 0 p-mutations
p = 25% 1 p-mutation
p = 10% 511 p-mutations
p = 5% 7,247 p-mutations
p = 2% 17,760 p-mutations
p = 1% 22,319 p-mutations
p = 0.5% 23,975 p-mutations
p = 0.25% 0 p-mutations
1 200,000 400,000 600,000 800,000 1,000,000 1,200,000 1,400,000 1,600,000 1,800,000 2,000,000
BACT2
p = 50% 0 p-mutations
p = 25% 3 p-mutations
p = 10% 6 p-mutations
p = 5% 13 p-mutations
p = 2% 113 p-mutations
p = 1% 378 p-mutations
p = 0.5% 1,631 p-mutations
p = 0.25% 6,703 p-mutations
1 200,000 400,000 600,000 800,000 1,000,000 1,200,000 1,400,000 1,600,000 1,800,000 2,000,000 2,200,000 2,400,000 2,600,000 2,800,000
Sequence position (1-indexed)

Figure S16: Locations of p-mutations throughout the three selected MAGs. Black or green vertical lines indicate p-mutations; black lines correspond to mutations located within at least one gene, while green lines correspond to mutations located within intergenic regions. The values of p shown here are the same as those shown in the context of the diversity index plots in Fig. 4. p-mutations are only identified for sufficiently-covered positions (represented by gray horizontal bars); insufficiently-covered positions are marked with red horizontal bars instead. We determine "sufficient coverage" as described in Supplemental Material **"Diversity index details"** for p-mutations, using $minReadNumber = 5$. As was also the case in Fig. 4, BACT1 is not sufficiently covered for $p = 0.25\%$ (the minimum sufficient coverage for this value of p is $\frac{5}{0.0025} =$ 2,000x). These plots are a very coarse, high-level way of visualizing mutation locations across these MAGs, and identifying small single-nucleotide-level details is challenging due to the limited resolution available. However, these plots are nonetheless useful for making high-level comparisons between the distributions of p-mutations across these MAGs. We note that MAGs are represented here linearly rather than circularly, a distinction from the figures in (Sekowska *et al.*, 2016; Wilm *et al.*, 2012). This representation is used both to simplify the comparison of many different sequences and values of p, and because the sequence corresponding to CAMP is not completely assembled into a single circular sequence (as discussed in Supplemental Material **"Assembly graph"**).

(Geller *et al.*, 2015). Under our null hypothesis, the mutation status of each position in a contig can be thought of as the result of a Bernoulli trial with probability of "failure" (the position is mutated) set to $q = \frac{m}{n}$, and probability of "success" (the position is not mutated) set to $p = 1 - q$. If the null hypothesis were true, then the number of non-mutated positions in a contig should thus follow the Binomial distribution with expected value $np = n\big(1 - \frac{m}{n}\big) = n - m$.

This null hypothesis is an obvious oversimplification that will be broken in practice: for example, Fig. 3 makes clear that the probability of a position being mutated is not the same for all positions in a contig. However, we submit that it is an acceptable approximation, especially since these p-values are intended only as a starting point for further analysis of these gaps.

For each contig, we can compute the probability of the longest gap in the contig (equivalently, the longest run of "successes" in a sequence of Bernoulli trials) using the final equation shown in (Bateman, 1948). We allow the user to choose between attempting to compute exact p-values using the equation from (Bateman, 1948), or using the approximation method shown in (Naus, 1982). For example: using NaiveFreq p-mutation calling at $p = 0.5\%$ (and ignoring whether or not sufficient coverage exists, unlike in Table S3 and Fig. S16), the longest coldspot in BACT1 has length 19,167 bp (corresponding to the fifth coldspot listed in Table S3). The probability of seeing a gap at least this long by chance, considering BACT1's length and the amount of $p = 0.5\%$ mutations contained therein, is 6.0498×10^{-91} (computed using the exact method from (Bateman, 1948)).

We note that these p-values come with some caveats in addition to the obvious simplicity of the null hypothesis used. For one, we do not correct for multiple testing (so blindly classi-

Left	Right	Length (bp)	Coverage	Kaiju	BLASTN
740,701	750,399	9,699	1,261x	*Clostridium* sp. CAG:1013	“No significant similarity found”
974,177	979,570	5,394	1,226x	*Bacteroides fragilis*	*Emticicia oligotrophica* DSM 17448
1,183,798	1,229,941	46,144	432x	*Tannerella* sp. CAG:118	*Podoviridae* sp.
1,229,943	1,239,536	9,594	1,384x	*Prevotella disiens*	*Podoviridae* sp.
1,618,448	1,637,614	19,167	1,454x	*Bacteroides uniformis*	*Phocaeicola salanitronis* DSM 18170
1,798,600	1,813,374	14,775	1,374x	*Alistipes* sp. CHKCI003	Uncultured organism clone VC1D523TF
2,140,896	2,147,770	6,875	1,345x	*Bacteroidales*	*Ruminococcus* sp. JE7A12

Table S3: Information about all coldspots of length $\geq$ 5,000 bp in BACT1. Coldspots are listed from left to right in the MAG. For the purposes of this table we define coldspots as regions of BACT1 containing no p-mutations at $p = 0.5\%$, and we only attempt to call p-mutations at sufficiently-covered positions (where the coverage is at least $\frac{5}{0.005}$ = 1,000x, as discussed in Supplemental Material **“Diversity index details”**). As in Table S1, the “Coverage” column describes the rounded average coverage of each region: most coldspots have coverages consistent with the BACT1 average of 1,415x, although the third coldspot (containing the large region of insufficiently-covered positions in the middle of BACT1 shown in Fig. S16) has a much lower coverage. We provide taxonomic classifications to provide a basic analysis of why these parts of BACT1 are “cold”—due, for example, to having evolutionarily conserved functions or arising from prophage insertions. The “Kaiju” column describes the taxonomic assignments from Kaiju (Menzel *et al.*, 2016) of each region, computed as described in Table S1. Since Kaiju was only able to classify the entirety of BACT1 as *Bacteria*, the diversity of taxonomic classifications shown for different regions of this MAG is not necessarily surprising. This diversity could be due to a variety of factors: for example, this MAG could correspond to a novel species, or it could contain multiple strain-level genomes collapsed into a single sequence. This latter situation is common for many MAGs, for many metagenomic assemblers; this is in part a consequence of us running metaFlye without the `--keep-haplotypes` flag, as well (Supplemental Material **“Assembly graph”**). Since some of these coldspots are relatively short (Kaiju relies on protein-level matches), we also used BLASTN 2.12.0+ (Zhang *et al.*, 2000) on the NCBI nucleotide collection database (Morgulis *et al.*, 2008) with default parameters to attempt to assign matches to these regions. The “BLASTN” column describes the scientific name of the sequence listed first in the BLASTN results for each coldspot region, sorting sequence matches by ascending E-values; BLASTN did not identify any matches for the leftmost coldspot.

fying all longest-gap p-values less than 0.05, for example, as "significant" may be problematic when considering datasets with many contigs). Additionally, the equation for computing the p-value of a gap is given only for a sequence of Bernoulli trials; in the rare case where the longest coldspot gap in a contig is the "loop-around" gap from the rightmost mutation to the leftmost mutation, we implicitly ignore that this break in the contig exists, and act as if this gap were instead in the middle of the contig. This may technically violate the assumptions made by the equation, but we do not imagine it should complicate the interpretation of these p-values.

P Growth dynamics

Here we demonstrate simple analyses of the three selected MAGs' growth dynamics by comparing their coverages (Supplemental Material **"Coverages and deletion-rich positions"**) and GC skews. We emphasize that strainFlye performs these analyses without reliance on reference databases.

A global maximum in a coverage plot usually indicates the origin of replication for a bacterial genome that is undergoing replication, and the ratio of this global maximum's coverage to a global minimum's coverage (the *peak-to-trough ratio*, or *PTR*) can measure *in vitro* bacterial growth rates (Korem *et al.*, 2015). Similarly, the global minimum of *GC skew* of a bacterial genome can also be indicative of its origin of replication (Lobry, 1996; Grigoriev, 1998). Fig. S17 shows plots of coverage and GC skew for the three selected MAGs, demonstrating a clear anticorrelation between coverage and skew. To simplify these plots, we bin each MAG's coverage and skew; the `strainFlye dynam covskew` command can compute these binned quantities.

We can compute the PTR (Korem *et al.*, 2015) for each MAG by computing the ratio of normalized coverage at the bin with the global minimum of the skew to the ratio of normalized coverage at the bin with the global maximum of the skew; both extrema are highlighted in Fig. S17. CAMP's PTR is $\frac{1.18}{0.93} \approx 1.27$; BACT1's PTR is $\frac{1.01}{0.93} \approx 1.08$; and BACT2's PTR is $\frac{1.28}{0.91} \approx 1.40$. All of these values are consistent with the distributions of PTRs shown in Fig. 1(B) of (Korem *et al.*, 2015), and they indicate that—if the organisms represented by these MAGs are indeed undergoing replication—the replication rate is higher for BACT2 and CAMP than for BACT1. Although further work would be needed to validate these claims, our ability to make these observations shows that HiFi-based metagenomics enables the evaluation of the growth dynamics of a metagenome.

We close by noting that, beyond serving as an example of the utility of complete metagenomics, information about PTRs can be useful in a variety of contexts when studying human-associated microbiota (Joseph *et al.*, 2022). Prior work has also linked mutation rates and replication timing within a single yeast chromosome (Lang & Murray, 2011); so, information about putative replication origins has potential to inform a "prior" about how likely mutations in certain regions of a MAG are, extending on the approaches for identifying rare mutations described in this paper.

Q The link graph structure for haplotype visualization

Here we define *link graphs* for representing mutations within a MAG that likely co-occur on the same strain. Nodes in the link graph represent *alleles* of mutated positions, and edges connect alleles that are observed together in the reads. The link graph resembles the two graph structures (the "simple graph" and "probabilistically weighted graph") for metagenomic haplotyping proposed in (Nicholls *et al.*, 2021). The link graph differs from the graphs proposed in that paper in that non-adjacent pairs of mutations can be connected; the edge weighting methods are also somewhat simpler than those defined for the probabilistically weighted graph (Nicholls *et al.*, 2021). That said, we expect that for many cases these graphs should be fairly similar. We provide the `strainFlye link` module for the construction of these graphs.

Defining a link graph. We define one link graph per MAG. For each position i at which a mutation was called, we define $\text{reads}(i, N)$ as the number of reads with nucleotide $N \in \{A, C, G, T\}$ at position i. We add up to four nodes for each mutated position: each node (i, N) represents the occurrence of a specific nucleotide at position i, and is only added if $\text{reads}(i, N) > 1$.

We then consider all pairs of nodes (i, N_i) and (j, N_j), where $i \neq j$. We define a read as an (i, j, N_i, N_j)-*read* if its aligned nucleotide at position i is N_i and its aligned nucleotide at position j is N_j. We define $\text{reads}(i, j, N_i, N_j)$ as the number of (i, j, N_i, N_j)-reads. We can now define the *link weight* between two alleles:

$$\text{link}(i, j, N_i, N_j) = \frac{\text{reads}(i, j, N_i, N_j)}{\max(\text{reads}(i, N_i), \text{reads}(j, N_j))}$$

We also define $\text{spanCount}(i, j) = \sum_{N_i \in \{A,C,G,T\}} \left(\sum_{N_j \in \{A,C,G,T\}} \left(\text{reads}(i, j, N_i, N_j) \right) \right)$: this is the total number of reads spanning two positions i and j (summing across all combinations of alleles).

Finally, we connect two nodes (i, N_i) and (j, N_j) by an undirected edge of weight $\text{link}(i, j, N_i, N_j)$ if (i) $\text{spanCount}(i, j) \geq \textit{minSpan}$, and (ii) $\text{link}(i, j, N_i, N_j) > 0$. In the context of the high-coverage MAGs in SheepGut, we set the *minSpan* parameter to 501; lower values of this parameter will be required for lower-coverage MAGs, or for lower-coverage sequencing projects. Visualizations of the distributions of $\text{reads}(i, j, N_i, N_j)$ and $\text{link}(i, j, N_i, N_j)$ across the link graphs of the three selected MAGs in SheepGut are shown in Fig. S18.

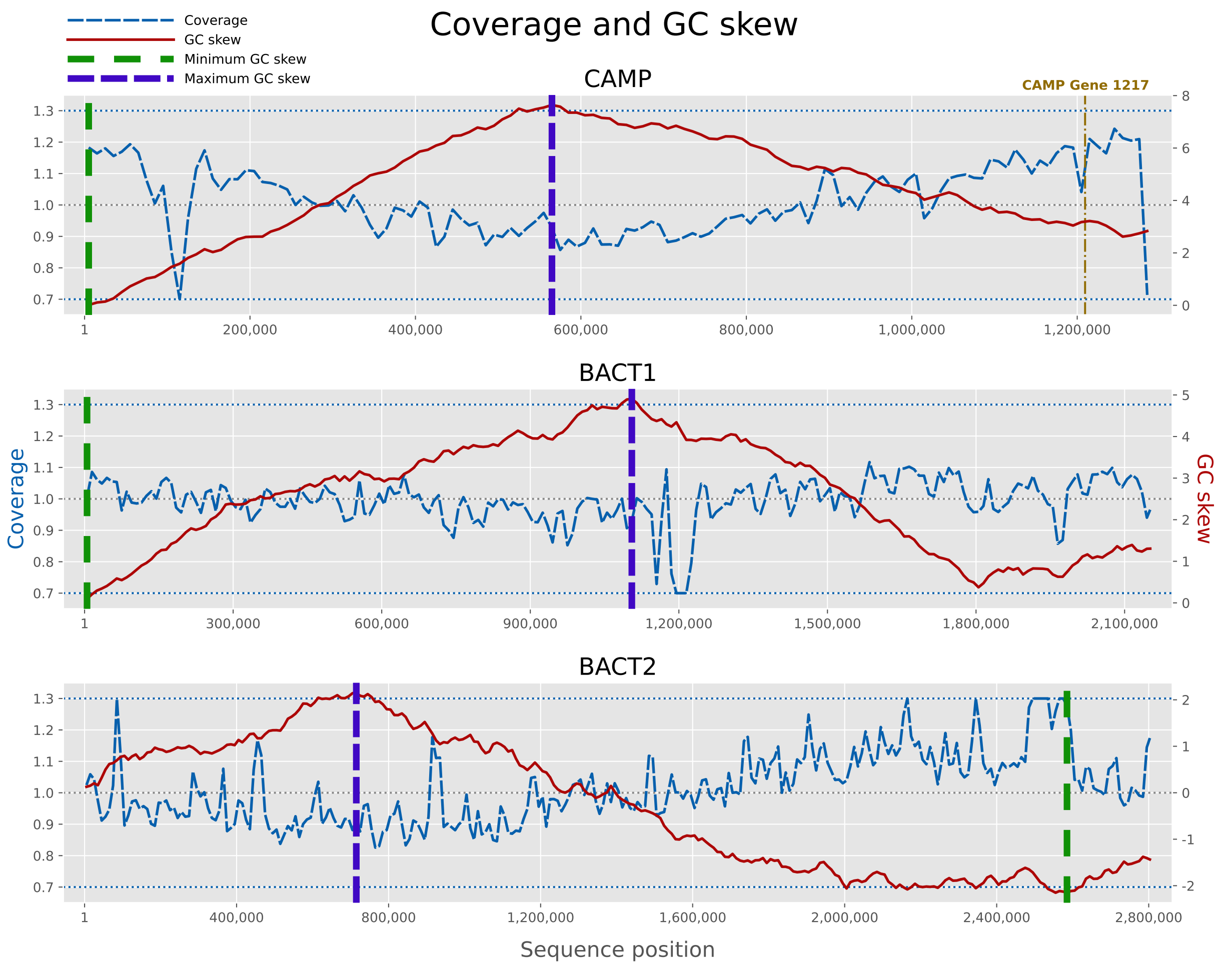


Figure S17: Coverage and GC skew throughout the three selected MAGs. We bin and normalize coverages in order to simplify comparisons (both between coverage and skew for the same MAG, and between coverages of different MAGs). Starting at the left end of each MAG, each 10 kbp of positions are combined into a single bin. (The rightmost bin for each MAG can contain less than 10 kbp positions.) We compute the median coverage of each bin and then compute the entire MAG's median coverage (M) by taking the median of all bins' median coverages. We then divide each bin's median coverage by M in order to normalize each bin's coverage. These normalized coverages are clamped to the range $[0.7, 1.3]$ to limit the visual impact of outliers that likely represent sequencing artifacts: these boundaries, in addition to $y = 1$, are represented as horizontal dotted lines on each plot. We compute GC skew for the same bins of 10 kbp used for coverage. The GC skew of a bin containing G instances of G and C instances of C is defined as $\frac{G-C}{G+C}$, plus—for all bins after the leftmost—the GC skew of the bin immediately to the left of the current bin (Grigoriev, 1998). The midpoints of the bins with the global minimum and maximum GC skew in each MAG are highlighted with vertical lines, a choice inspired by Fig. 1 of (Korem *et al.*, 2015). We also highlight the midpoint of gene 1217, the highest-mutation-rate gene in CAMP, as another vertical line in the CAMP plot. This enables us to follow up on our observation in Fig. S14 in Supplemental Material **"Hotspot genes in the three selected MAGs"** that this gene's coverage is increasing. Here we can confirm that this gene is located in a region of the MAG where, in general, coverage is increasing and GC skew is decreasing; this may be indicative that this region of the CAMP is being actively replicated. Although all three MAGs' skew plots have clear global maxima, only BACT2's skew plot has an "obvious" global minimum. The fact that CAMP is not completely assembled, as discussed in Supplemental Material **"Assembly graph"**, may be a causal factor for its lack of a clear global minimum skew.

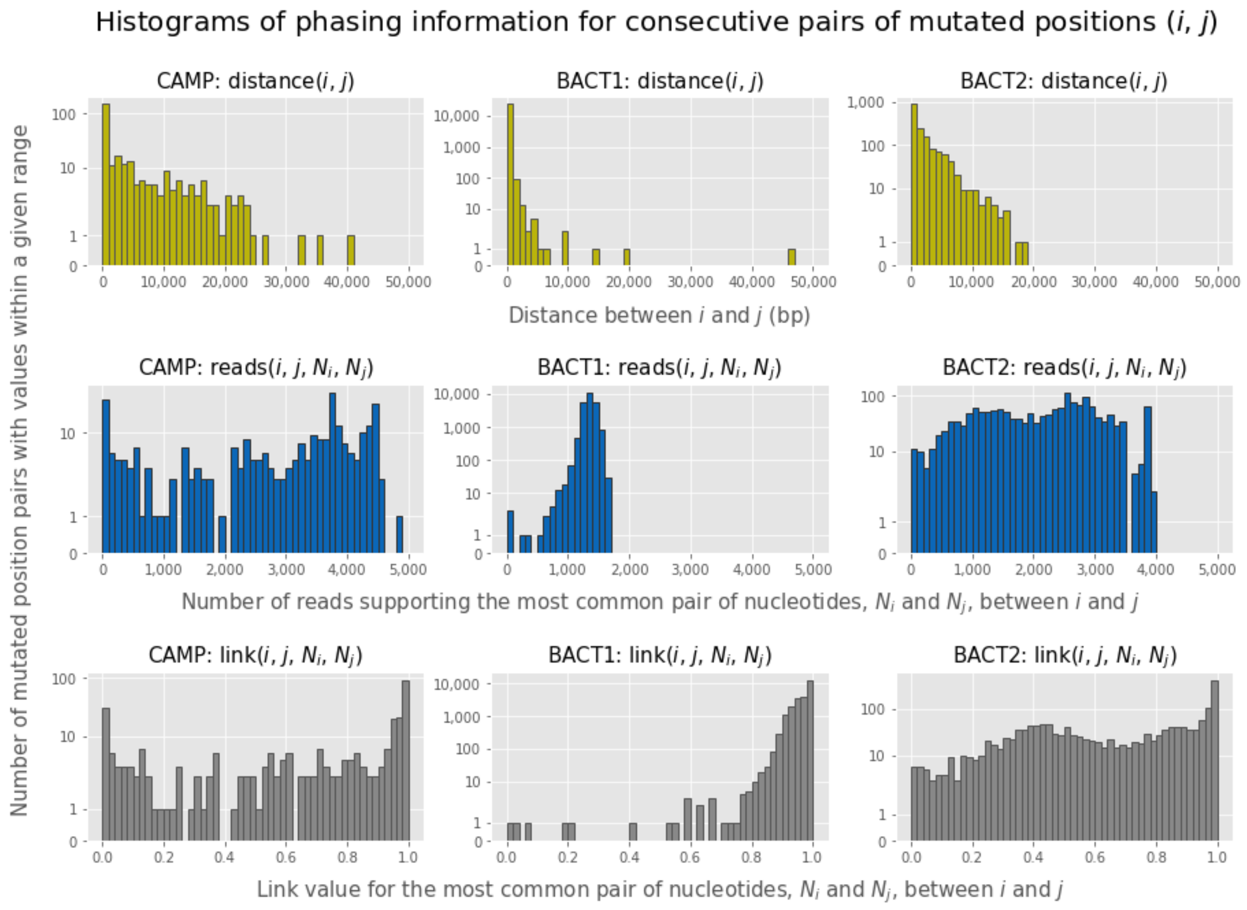


Figure S18: Histograms showing various phasing statistics for each pair of consecutive mutated positions (i, j) in the three selected MAGs. Mutated positions were identified by NaiveFreq at $p = 0.5\%$, and filtered to just positions with a coverage of at least 1,000x. Two mutated positions i and j are considered as a pair of *consecutive* mutated positions if no other mutated positions (using the same criteria of $p = 0.5\%$ and minimum coverage $\geq$ 1,000x) occur in between i and j. (For BACT1 and BACT2, which are represented as connected components of a single circular sequence in the assembly graph, we consider the rightmost mutation r and the leftmost mutation ℓ as a pair of consecutive mutated positions, since reads can "loop around" these two MAGs' sequences in the alignment.) The top row shows a histogram of the distance between i and j. The middle row shows reads(i, j, N_i, N_j): for each pair (i, j), only the most frequent pair of nucleotide positions (N_i, N_j) is included here. The bottom row shows link(i, j, N_i, N_j), selecting only the most frequent shared haplotype in the same way as the second row. We note that, if a pair of consecutive mutated positions (i, j) are not spanned by at least one read, then reads(i, j, N_i, N_j) = link$(i, j, N_i, N_j) = 0$ for this pair. (This is the case for all pairs with distances above 25,000 bp shown in the first row of plots.)

Visualizing link graphs. Connected components of the link graph represent groups of mutations that tend to co-occur. Here we visualize components of two link graphs from the SheepGut dataset, produced using the set of mutated positions identified by NaiveFreq at $p = 0.5\%$ with coverage of at least $minCov = 1{,}000$x.

As expected, the largest few connected components (sorted by number of nodes per component) in the link graph of BACT1, which contains many rare mutations, are much larger than the largest components of CAMP and BACT2. Fig. S19 shows the largest connected component of the link graphs for BACT1 and CAMP. Nodes are colored according to the gene(s) in which the node's position is contained. This indicates that the BACT1 component spans a large region of BACT1 and that the CAMP component spans a comparatively small region of CAMP. This is a consequence of highly-mutated MAGs being easier to phase than sparsely mutated ones (Nicholls *et al.*, 2021).

Link graphs can be helpful for visualizing what portions of a MAG are possible to phase, but their general usefulness is limited. strainFlye thus also supports the ability to generate smoothed haplotypes directly, as discussed in the section **"Phasing identified mutations"**. These analyses and visualizations demonstrate the ability of HiFi reads to span long regions of highly-mutated MAGs, and thus indicate putative haplotypes.

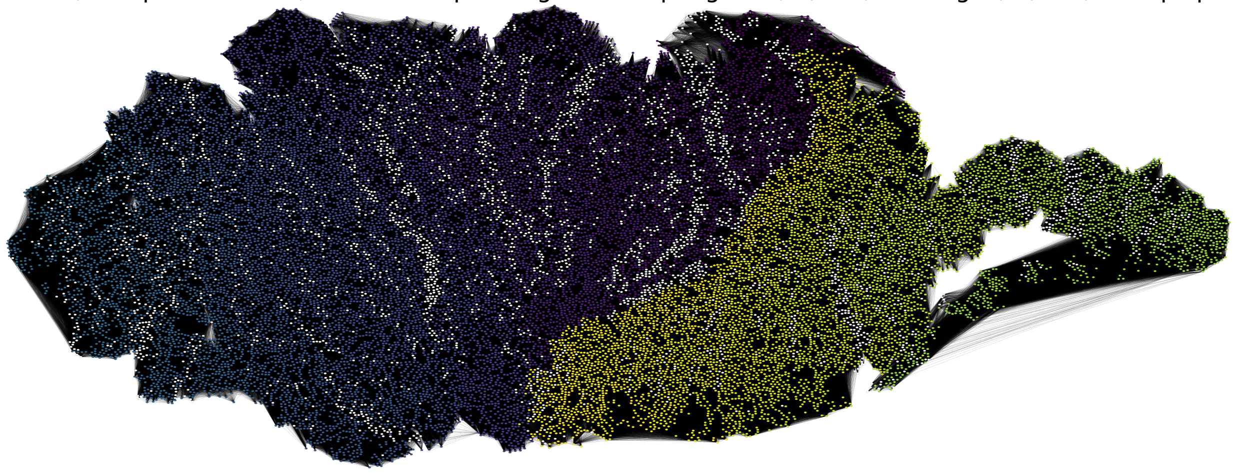
R Haplotypes of the most mutated gene in CAMP

During the development of strainFlye's `smooth` module, we tested our pipeline against a simple example—the region surrounding the highest-mutation-rate gene in CAMP (Fig. 5 (top))—in order to provide a simple benchmark for our ability to recover strain-level haplotypes from a MAG. We found that using the jumboDBG module of LJA (Bankevich *et al.*, 2022) by itself, or using the jumboDBG and multiplexDBG modules of LJA without any error correction step, resulted in assembly graphs more complex than we expected. We wanted to investigate methods for simplifying this graph while limiting the removal of real strain-level variation; here we document these tests.

Analysis of haplotypes of this gene. We limit our focus only to reads that surround the highest-mutation-rate gene (gene 1217) in CAMP. To do this, we define a region of positions in CAMP that includes the entirety of gene 1217, as well as 25 kbp before and after this gene. We will only consider smoothed reads whose corresponding linear alignments to CAMP overlap with this region (we do not generate any virtual reads for this particular analysis).

We define M as the set of all 34 mutated positions (given $p = 10\%$, based on Fig. 5 (top)) located in gene 1217 in CAMP. There are 4,054 smoothed reads originating from linear alignments that span all positions in M without deletions or skips. We extracted the haplotypes from these smoothed reads at all positions in M. For the purposes of this Supplement, we now define a *haplotype* of a smoothed read as a string of 34 characters, such

BACT1, component 1: 24,780 nodes spanning 784 unique genes, 4,225,301 edges, 2,150,722 bp span



CAMP, component 1: 110 nodes spanning 9 unique genes, 3,473 edges, 35,757 bp span

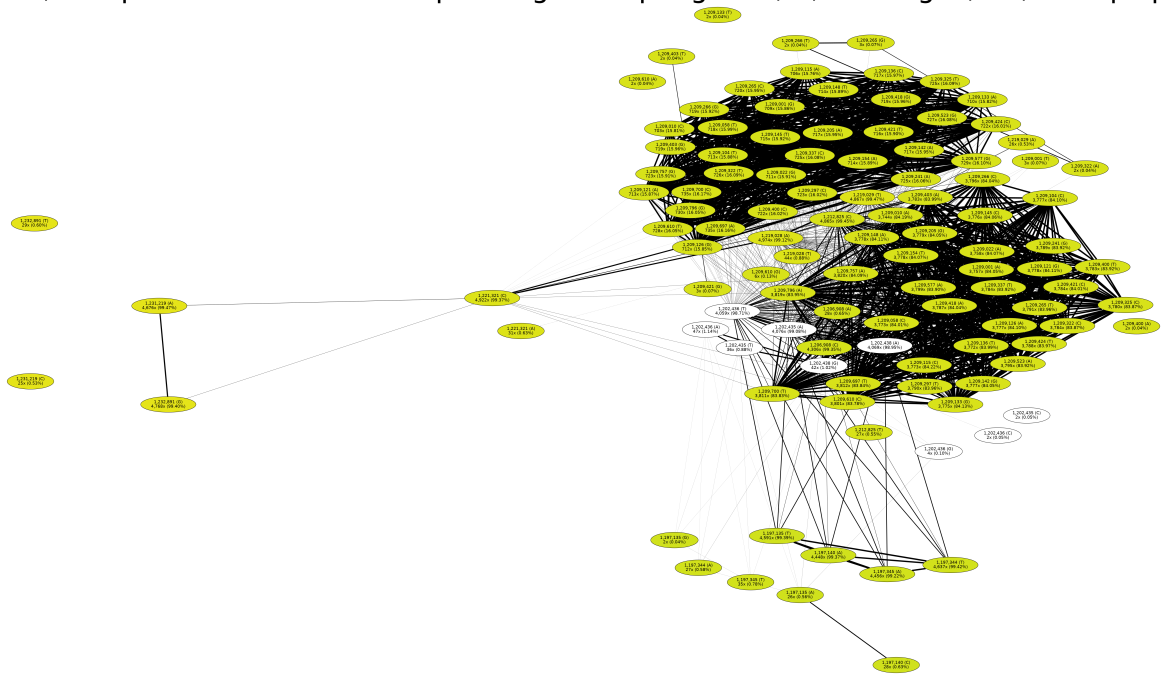


Figure S19: Largest components in the link graphs of BACT1 (top) and CAMP (bottom). Nodes (alleles) are colored by the gene(s) in which their corresponding position is located. Nodes present in a single gene are colored using the viridis gradient (Smith & van der Walt, 2015): darker (lighter) colors correspond to genes closer to the start (end) of the MAG. Positions present in multiple genes are colored gray, and positions located in intergenic regions are colored white. Nodes in the BACT1 (CAMP) component span a large (small) region of BACT1 (CAMP). The smaller CAMP component includes additional labels for each node: (i) position in the MAG, (ii) nucleotide at this position, (iii) the frequency with which this nucleotide was observed at this position (reads(i, N)), and (iv) the relative frequency of this nucleotide at this position. Many of the mutated positions in this component are located within the gene shown in Fig. 5 (top). Edges' thicknesses in the CAMP component's visualization are scaled by their link values, so that thicker edges roughly correspond to higher link values. Interestingly, this component contains two large "clique"-like structures including many nodes that have high-link edges between each other. These two structures correspond to the two different haplotypes of CAMP within this gene shown in Fig. 5 (top), with one haplotype at roughly 16% relative frequency and another haplotype at roughly 84% relative frequency. We visualized these graphs using Graphviz (Gansner & North, 2000), using the sfdp (Hu, 2005) layout method.

that the n-th character of this haplotype corresponds to this smoothed read's nucleotide at the n-th mutated position in gene 1217.

Fig. S20(a) shows a sequence logo that confirms that these mutated positions are consistently dominated by a "reference" (∼84% frequency, at 3,430x coverage) and "alternate" (∼16% frequency, at 600x coverage) haplotype. These two haplotypes are herein referred to as the "primary" haplotypes of this gene.

There are, however, 14 unique haplotypes present in the smoothed reads which differ from the primary haplotypes. The first question we had about these "non-primary" haplotypes was how similar they were to the two primary haplotypes. We computed the Hamming distance (Hamming, 1950) between all unique haplotypes and the "reference" haplotype; a histogram of these Hamming distances is shown in Fig. S20(b). This histogram clarifies that all non-primary haplotypes closely resemble the reference or alternate haplotypes.

To get a sense of how common the non-primary haplotypes are compared to each other (e.g. "does one of these haplotypes occur frequently?"), we plot a histogram of these haplotypes' coverages in Fig. S20(c). This histogram reassures us that all non-primary haplotypes are extremely uncommon, considering that the total coverage of smoothed reads spanning all mutated positions in gene 1217 in CAMP is 4,054x.

As we consider whether these non-primary haplotypes represent real or technical variation, we next investigate whether these non-primary haplotypes consistently vary at certain mutated positions. The 34 mutated positions from which we have constructed these haplotypes are all already "hotspots," but it may turn out that some of these positions may vary in the same way for many unique haplotypes—this could indicate that these haplotypes are evolutionarily related to each other. (A more formal way of conducting this analysis might

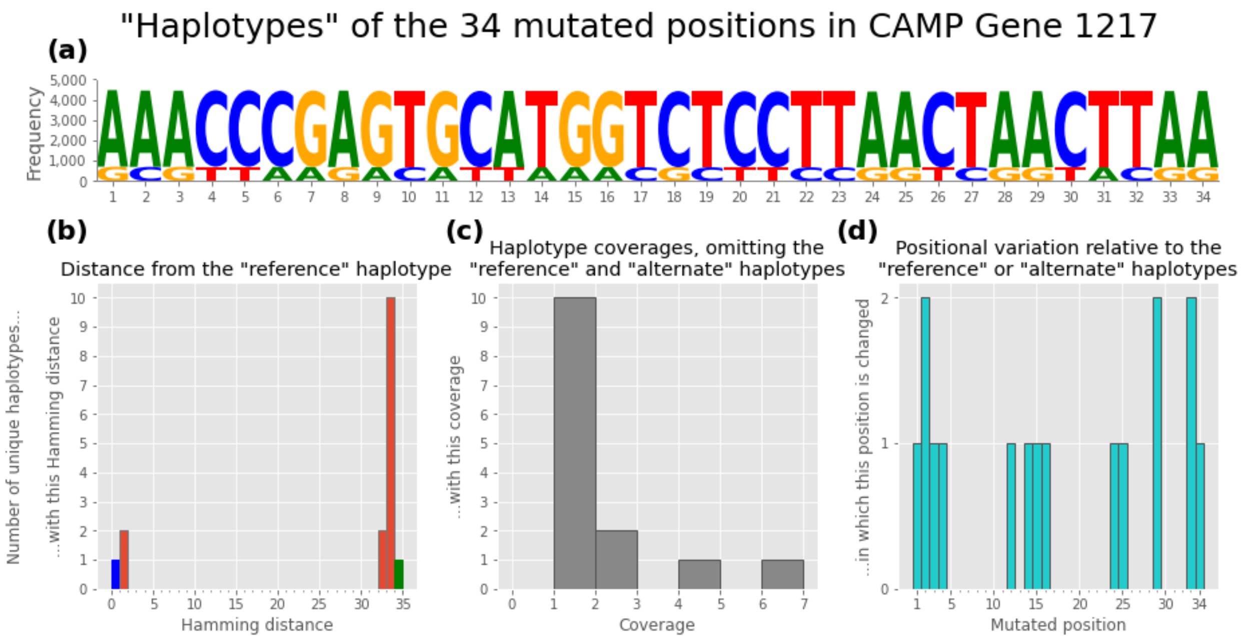


Figure S20: Haplotypes of the 34 mutated positions ($p = 10\%$) located within gene 1217 of CAMP. (a) Sequence logo of these mutated positions' aligned nucleotides' frequencies. Since we created this plot based on the haplotypes represented in the smoothed reads, it thus only includes the two most common aligned nucleotides at each of these mutations (as discussed in **"Methods"**). However, this caveat should not make a large difference. This logo was visualized using Logomaker (Tareen & Kinney, 2020). **(b)** Histogram of Hamming distances for all of the 16 unique haplotypes spanning gene 1217 in CAMP, relative to the reference haplotype. Since these haplotypes are 34-character strings, the Hamming distance of a given haplotype is constrained to within the range $[0, 34]$. This histogram includes the reference and alternate haplotypes for illustrative purposes: the reference haplotype (which has a Hamming distance of 0 to itself) is located in the leftmost bin and colored blue, while the alternate haplotype (which has the maximum possible Hamming distance of 34, since it disagrees with the reference haplotype at every mutated position) is located in the rightmost bin and colored green. This plot shows that most haplotypes' Hamming distances are similar to that of either the reference or alternate haplotype, implying that most of these haplotypes represent small deviations from these two "primary" haplotypes. **(c)** Histogram of haplotype coverages for all unique non-primary haplotypes. The two primary haplotypes are omitted from this histogram due to their relatively large coverages (3,430x and 600x); the non-primary haplotypes all have coverages of less than 10x, indicating that these haplotypes may correspond to sequencing errors or extremely rare mutations. For reference, the highest-coverage (6x) non-primary haplotype, `GAGTTAAGACATTAAACGCTTCCGGTCGGTACGG`, has a Hamming distance to the reference haplotype of 33, meaning that this haplotype disagrees with the alternate haplotype at only one mutated position (the second, at which it has an A instead of a C). **(d)** Chart of positional variation for the non-primary haplotypes. The x-axis corresponds to the 34 mutated positions in gene 1217. For each of the non-primary haplotypes, we compute two Hamming distances: D_R is the Hamming distance to the reference haplotype, and D_A is the Hamming distance to the alternate haplotype. If $D_R > D_A$, we add "+1" for all differing positions relative to the alternate haplotype; if $D_R < D_A$, we add "+1" for all differing positions relative to the reference haplotype. ($D_R \neq D_A$ for all haplotypes, and—as indicated in panel (b)—all but two of the non-primary haplotypes have $D_R > D_A$.) This plot indicates that the distribution of differing positions is roughly uniform across the non-primary haplotypes.

involve constructing a phylogenetic tree.) As a simple way to check this, we plot the number of times each of the 34 mutated positions varies from either the reference or alternate haplotypes in Fig. S20(d).

Fig. S20(d) provides evidence against the idea that these non-primary haplotypes consistently vary from the reference or alternate haplotypes at the same mutated positions, since—rather than being biased toward changes at a certain mutated position—the distribution of differences across each mutated position is roughly uniform (albeit with many zeroes, since there are only 14 unique non-primary haplotypes and 34 possible mutated positions).

Taken together, these results thus serve as (imperfect) justification for ignoring these non-primary haplotypes of gene 1217: in general these haplotypes are low-coverage (Fig. S20(c)), only differ subtly from the primary haplotypes (Fig. S20(b)), and do not seem to differ from these primary haplotypes in consistent ways (Fig. S20(d)). For a first-pass analysis, we thus propose removing low-coverage edges in the graph that likely arise due to these haplotypes.

Assembly of haplotypes of this gene. Based on our analysis of the haplotypes of gene 1217 in CAMP (Fig. S20), we restructured our use of LJA to include a simple coverage filter applied after running jumboDBG but before running multiplexDBG. This coverage filter removes edges from the graph with k-mer coverages below a user-configurable threshold $minKmerCov$ (default value 10; we note that "k-mer coverage" is not exactly comparable to the conventional definition of coverage).

Running LJA in this way on the smoothed reads intersecting gene 1217 in CAMP (using $minKmerCov = 10$) results in a simple assembly graph containing two isolated edges. We can determine the haplotype of gene 1217 represented by an edge by aligning this edge's sequence to CAMP using minimap2 (Li, 2018) and then extracting the nucleotides of this edge that have been matched to all mutated positions. For the sake of brevity, in the remainder of this Supplement we refer to this process as *haplotype extraction*.

Using haplotype extraction, we confirmed that one of the edges produced by LJA with $minKmerCov = 10$ represents the reference haplotype, while the other edge represents the alternate haplotype. From our perspective, this is the "ideal" result for this simple example, and this bodes well for the application of this approach to other, more complex read-sets (although more thorough testing on different types of communities would be ideal).

Benchmarking assembly of haplotypes of this gene. For comparison's sake, we also ran six other assembly methods in order to determine which haplotypes of gene 1217 in CAMP were recovered by these methods. We ran metaFlye (with default parameters), metaFlye (with the `--keep-haplotypes` flag), jumboDBG (with $k = 5{,}001$), LJA (with no error correction), LJA (with its default "mowerDBG" error correction), and LJA (with the simple k-mer coverage filter described above, but using $minKmerCov = 1$ instead of 10).

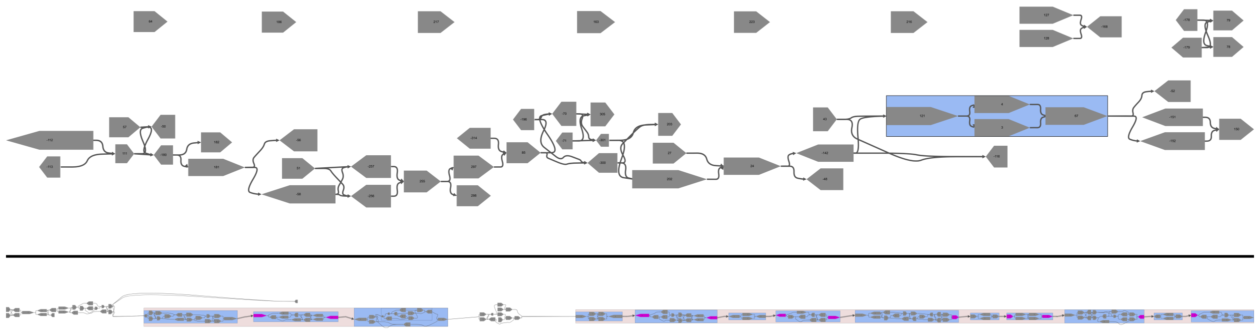


Figure S21: Multiplex de Bruijn graphs produced by LJA for the BACT1 (top) and BACT2 (bottom) MAGs' smoothed and virtual reads. The BACT1 graph is surprisingly simple; this is likely a consequence of many potential smoothed reads being discarded, as discussed in **"Methods"**. The BACT2 graph, however, resembles CAMP's (Fig. 6) in that it primarily consists of a series of consecutive "bubble" patterns. We produced these visualizations using MetagenomeScope (Fedarko *et al.*, 2017).

Three of these methods (both metaFlye runs, and LJA with mowerDBG error correction) produced assemblies containing only one edge. Using haplotype extraction, we confirmed that each of these lone edges corresponded to the reference haplotype. These assemblies are thus "over-corrected," since they do not include the alternate haplotype (which clearly corresponds to a real alternate strain).

The other three of these methods (jumboDBG, LJA with no error correction, and LJA with $minKmerCov = 1$) produced assemblies containing more than two edges. jumboDBG's assembly contained 32 edges, while the two LJA runs' assemblies both contained five edges. Although all three assemblers recovered the reference and alternate haplotypes (as determined by haplotype extraction of the assembled edges), the fact that they recovered other haplotypes is undesirable—since, as discussed earlier, these other haplotypes likely represent noise. These assemblies are thus "under-corrected" for this particular example, relative to the run of LJA with $minKmerCov = 10$.

S Smoothed haplotype assembly graphs

Here we present additional visualizations of the assembly graphs produced by LJA for MAGs' smoothed and virtual reads, as described in the section **"Phasing identified mutations"**. Fig. S21 shows MetagenomeScope (Fedarko *et al.*, 2017) visualizations of the multiplex de Bruijn graphs for BACT1 and BACT2, analogous to the visualization for CAMP shown in Fig. 6 in the section **"Phasing identified mutations"**.

Supplemental References

Antipov, Dmitry, Rayko, Mikhail, Kolmogorov, Mikhail, & Pevzner, Pavel A. 2022. viralFlye: assembling viruses and identifying their hosts from long-read metagenomics data. *Genome Biology*, **23**(1), 1–21.

Balakirev, Evgeniy S, & Ayala, Francisco J. 2003. Pseudogenes: are they "junk" or functional DNA? *Annual review of genetics*, **37**(1), 123–151.

Bankevich, Anton, Bzikadze, Andrey V, Kolmogorov, Mikhail, Antipov, Dmitry, & Pevzner, Pavel A. 2022. Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads. *Nature Biotechnology*, 1–7.

Bateman, G. 1948. On the power function of the longest run as a test for randomness in a sequence of alternatives. *Biometrika*, **35**(1/2), 97–112.

Brewer, Cynthia, Harrower, Mark, Sheesley, Ben, Woodruff, Andy, & Heyman, David. 2021. *ColorBrewer: Color Advice for Maps*. `https://colorbrewer2.org`.

Casjens, Sherwood. 1998. The diverse and dynamic structure of bacterial genomes. *Annual review of genetics*, **32**(1), 339–377.

Cornish-Bowden, Athel. 1985. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Research*, **13**(9), 3021–3030.

Dicenzo, George C, & Finan, Turlough M. 2017. The divided bacterial genome: structure, function, and evolution. *Microbiology and Molecular Biology Reviews*, **81**(3), e00019–17.

Fedarko, Marcus, Ghurye, Jay, Treangen, Todd, & Pop, Mihai. 2017. MetagenomeScope: web-based hierarchical visualization of metagenome assembly graphs. *Pages 630–632 of: Proceedings of the 25th International Symposium on Graph Drawing and Network Visualization*, vol. 10692. Springer.

Feng, Xiaowen, Cheng, Haoyu, Portik, Daniel, & Li, Heng. 2022. Metagenome assembly of high-fidelity long reads with hifiasm-meta. *Nature Methods*, 1–4.

Gansner, Emden R, & North, Stephen C. 2000. An open graph visualization system and its applications to software engineering. *Software: practice and experience*, **30**(11), 1203–1233.

Geller, Ron, Domingo-Calap, Pilar, Cuevas, José M, Rossolillo, Paola, Negroni, Matteo, & Sanjuán, Rafael. 2015. The external domains of the HIV-1 envelope are a mutational cold spot. *Nature Communications*, **6**(1), 1–9.

Grigoriev, Andrei. 1998. Analyzing genomes with cumulative skew diagrams. *Nucleic Acids Research*, **26**(10), 2286–2290.

Hamming, Richard W. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, **29**(2), 147–160.

Hardy, André. 1994. An examination of procedures for determining the number of clusters in a data set. *Pages 178–185 of: New approaches in classification and data analysis*. Springer.

Hu, Yifan. 2005. Efficient, high-quality force-directed graph drawing. *Mathematica journal*, **10**(1), 37–71.

Hurst, Laurence D. 2002. The Ka/Ks ratio: diagnosing the form of sequence evolution. *Trends in genetics: TIG*, **18**(9), 486–486.

Joseph, Tyler A, Chlenski, Philippe, Litman, Aviya, Korem, Tal, & Pe'er, Itsik. 2022. Accurate and robust inference of microbial growth dynamics from metagenomic sequencing reveals personalized growth rates. *Genome Research*, **32**(3), 558–568.

Kille, Bryce, Liu, Yunxi, Sapoval, Nicolae, Nute, Michael, Rauchwerger, Lawrence, Amato, Nancy, & Treangen, Todd J. 2021. Accelerating SARS-CoV-2 low frequency variant calling on ultra deep sequencing datasets. *Pages 204–208 of: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE.

Kimura, Motoo. 1980. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, **16**(2), 111–120.

Kolmogorov, Mikhail, Bickhart, Derek M, Behsaz, Bahar, Gurevich, Alexey, Rayko, Mikhail, Shin, Sung Bong, Kuhn, Kristen, Yuan, Jeffrey, Polevikov, Evgeny, Smith, Timothy PL, *et al.* 2020. metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, **17**(11), 1103–1110.

Korem, Tal, Zeevi, David, Suez, Jotham, Weinberger, Adina, Avnit-Sagi, Tali, Pompan-Lotan, Maya, Matot, Elad, Jona, Ghil, Harmelin, Alon, Cohen, Nadav, *et al.* 2015. Growth dynamics of gut microbiota in health and disease inferred from single metagenomic samples. *Science*, **349**(6252), 1101–1106.

Lang, Gregory I, & Murray, Andrew W. 2011. Mutation rates across budding yeast chromosome VI are correlated with replication timing. *Genome Biology and Evolution*, **3**, 799–811.

Li, Heng. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**(18), 3094–3100.

Li, Heng, Handsaker, Bob, Wysoker, Alec, Fennell, Tim, Ruan, Jue, Homer, Nils, Marth, Gabor, Abecasis, Goncalo, & Durbin, Richard. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics*, **25**(16), 2078–2079.

Lobry, Jean R. 1996. Asymmetric substitution patterns in the two DNA strands of bacteria. *Molecular biology and evolution*, **13**(5), 660–665.

Manna, Dipankar, Breier, Adam M, & Higgins, N Patrick. 2004. Microarray analysis of transposition targets in Escherichia coli: the impact of transcription. *Proceedings of the National Academy of Sciences*, **101**(26), 9780–9785.

McLaren, Michael R, Willis, Amy D, & Callahan, Benjamin J. 2019. Consistent and correctable bias in metagenomic sequencing experiments. *Elife*, **8**, e46923.

Menzel, Peter, Ng, Kim Lee, & Krogh, Anders. 2016. Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nature Communications*, **7**(1), 1–9.

Morgulis, Aleksandr, Coulouris, George, Raytselis, Yan, Madden, Thomas L, Agarwala, Richa, & Schäffer, Alejandro A. 2008. Database indexing for production MegaBLAST searches. *Bioinformatics*, **24**(16), 1757–1764.

Mort, Matthew, Ivanov, Dobril, Cooper, David N, & Chuzhanova, Nadia A. 2008. A meta-analysis of nonsense mutations causing human genetic disease. *Human mutation*, **29**(8), 1037–1047.

Naus, Joseph I. 1982. Approximations for Distributions of Scan Statistics. *Journal of the American Statistical Association*, **77**(377), 177–183.

Nicholls, Samuel M, Aubrey, Wayne, De Grave, Kurt, Schietgat, Leander, Creevey, Christopher J, & Clare, Amanda. 2021. On the complexity of haplotyping a microbial community. *Bioinformatics*, **37**(10), 1360–1366.

Nurk, Sergey, Walenz, Brian P, Rhie, Arang, Vollger, Mitchell R, Logsdon, Glennis A, Grothe, Robert, Miga, Karen H, Eichler, Evan E, Phillippy, Adam M, & Koren, Sergey. 2020. HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *Genome Research*, **30**(9), 1291–1305.

Nurk, Sergey, Koren, Sergey, Rhie, Arang, Rautiainen, Mikko, Bzikadze, Andrey V, Mikheenko, Alla, Vollger, Mitchell R, Altemose, Nicolas, Uralsky, Lev, Gershman, Ariel, *et al.* 2022. The complete sequence of a human genome. *Science*, **376**(6588), 44–53.

Parks, Donovan H, Imelfort, Michael, Skennerton, Connor T, Hugenholtz, Philip, & Tyson, Gene W. 2015. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Research*, **25**(7), 1043–1055.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, *et al.* 2011. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, **12**, 2825–2830.

Seemann, Torsten. 2018. *barrnap 0.9: rapid ribosomal RNA prediction*. `https://github.com/tseemann/barrnap`.

Sekowska, Agnieszka, Wendel, Sofie, Fischer, Emil C, Nørholm, Morten HH, & Danchin, Antoine. 2016. Generation of mutation hotspots in ageing bacterial colonies. *Scientific Reports*, **6**(1), 1–7.

Smith, Nathaniel, & van der Walt, Stéfan. 2015. *A Better Default Colormap for Matplotlib*. SciPy 2015. `https://www.youtube.com/watch?v=xAoljeRJ3lU`.

Tareen, Ammar, & Kinney, Justin B. 2020. Logomaker: beautiful sequence logos in Python. *Bioinformatics*, **36**(7), 2272–2274.

The SAM/BAM Format Specification Working Group. 2022. *Sequence Alignment/Map Format Specification*. `https://samtools.github.io/hts-specs/SAMv1.pdf`.

Thorvaldsdóttir, Helga, Robinson, James T, & Mesirov, Jill P. 2013. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in bioinformatics*, **14**(2), 178–192.

Töpfer, Armin. 2017. *Juliet - One Click Minor Variant Calling*. `https://www.pacb.com/wp-content/uploads/4May2017_ArminToepfer_JulietMinorVariantCalling.pdf`.

Verbist, Bie MP, Thys, Kim, Reumers, Joke, Wetzels, Yves, Van der Borght, Koen, Talloen, Willem, Aerssens, Jeroen, Clement, Lieven, & Thas, Olivier. 2015. VirVarSeq: a low-frequency virus variant detection pipeline for Illumina sequencing using adaptive base-calling accuracy filtering. *Bioinformatics*, **31**(1), 94–101.

Vogel, F. 1972. Non-randomness of base replacement in point mutation. *Journal of Molecular Evolution*, **1**(4), 334–367.

Wang, Jing, Raskin, Leon, Samuels, David C, Shyr, Yu, & Guo, Yan. 2015. Genome measures used for quality control are dependent on gene function and ancestry. *Bioinformatics*, **31**(3), 318–323.

Wei, Zhi, Wang, Wei, Hu, Pingzhao, Lyon, Gholson J, & Hakonarson, Hakon. 2011. SNVer: a statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data. *Nucleic Acids Research*, **39**(19), e132–e132.

Wenger, Aaron M, Peluso, Paul, Rowell, William J, Chang, Pi-Chuan, Hall, Richard J, Concepcion, Gregory T, Ebler, Jana, Fungtammasan, Arkarachai, Kolesnikov, Alexey, Olson, Nathan D, *et al.* 2019. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature Biotechnology*, **37**(10), 1155–1162.

Wick, Ryan R, Schultz, Mark B, Zobel, Justin, & Holt, Kathryn E. 2015. Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*, **31**(20), 3350–3352.

Wilm, Andreas, Aw, Pauline Poh Kim, Bertrand, Denis, Yeo, Grace Hui Ting, Ong, Swee Hoe, Wong, Chang Hua, Khor, Chiea Chuen, Petric, Rosemary, Hibberd, Martin Lloyd, & Nagarajan, Niranjan. 2012. LoFreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-population heterogeneity from high-throughput sequencing datasets. *Nucleic Acids Research*, **40**(22), 11189–11201.

Zhang, Zheng, Schwartz, Scott, Wagner, Lukas, & Miller, Webb. 2000. A greedy algorithm for aligning DNA sequences. *Journal of Computational biology*, **7**(1-2), 203–214.