Xander: Gene Targeted Metagenomics

Jordan A Fish*[1,2], Yanni Sun[2] , James M Tiedje[1], James R Cole[1]

[1]Center for Microbial Ecology, Michigan State University [2]Department of Computer Science and Engineer, Michigan State University

Email: Jordan Fish*- fishjord@msu.edu; Yanni Sun - yannisun@msu.edu; James M Tiedje - tiedjej@msu.edu; James R Cole - colej@msu.edu;

*Corresponding author

Abstract

Background: Metagenomics can provide important insight in to microbial communities. It can be used to analyze entire genomes and takes full advantage of increasing sequencing capacity. However analyzing large metagenomic datasets has proven to be very computationally challenging with even modest metagenomic datasets requiring 256 gigabytes of memory or more (cite hmp assembly, titus). As dataset size progresses from 50 gigabases to 500 gigabases and beyond new methods are needed for deriving understanding from metagenomic datasets. We present a method for assembling protein coding sequences for gene(s) of interest from a metagenomic dataset which uses a compressible graph format and only assembles targeted data to drastically reduce the amount of memory and processing time required.

Results: . . .

Conclusions: . . .

Background

Metagenomics has the potential to help answer many questions but has faced scalability challenges stemming from the amount of raw sequencing data generated by metagenomic experiments. Metagenomic assembly has been an area of interest in recent years with the early datasets assembled using single genome assembly approaches. The tendency for single genome assemblers to only assemble a few dominant organisms has been an impetus to develop metagnomic specific assembly methods. Currently metagenomic assembly approaches focus on separating individual organism genomes out of the metagenomic data to be worked on individual (Metavelvet, Partitioning).

We propose a gene targeted approach for assembling metagenomic datasets called Xander. Xander is a De Bruijn Graph (cite) assembler that uses external information to perform an informed traversal of the assembly graph instead of an exauhstive traversal. Xander uses profile Hidden Markov Models (HMM) (cite) to target specific genes for assembly by guiding the assembly graph traversal. By using a HMM the most probable paths can be explored first to limit the area of the assembly graph that must be explored. In addition to limiting the graph traversal the HMMs provide a measure of how well the resulting assembled sequence matches the supplied model. Using this gene targeted approach allows for a functional based analysis of the data, by allowing users to directly examine genes involved in biologically interesting pathways.

Gene targeted assembly is less resource intensive and faster than whole genome metagenomic assembly. In addition to the De Bruijn Graph, only small paths relative to the graph's size must be kept in memory. Further reduction in the memory usage are achieved by using a probabilistic data structure for holding the De Bruijn Graph in memory, a Bloom filter [1] (cite kmer percolation). By targeting relatively small segments of the assembly graph by using an HMM to guide assembly, we limit how much of the graph must be explored during assembly, providing a speed up over whole genome approaches.

This Method is more sensitive than whole genome assembly and more specific than individual read-based approaches for functional analysis. Using an assembly based approach provides more context from which to make a classification decision as to whether a stretch of sequence belongs to the target gene family or not. Using HMM probabilities to guide local assembly helps to ensure we explore the most relevant paths to assemble sequences most likely to represent the gene of interest.

Other work in the field of targeted assembly are EMERIGE, Mira, and Nucleating Assembly. EMERIGE is an expectation maximization algorithm for assembling target sequences by iterative read mapping. The Mira assembler can work in a targeted mode by using mirabait to fish out all reads that overlap with a reference set and assembling on those reads. Nucleating assembly is another iterative algorithm that extends matches from a nucleating site using an iterative blast based approach. Xander is unique from these approaches since it does not use the read set after the De Bruijn Graph is built and can accomedate hyper-variable regions.

Results and Discussion

Xander's performance was evaluated using the Human Metabiome Project's (HMP) whole genome shotgun (WGS) mock community datasets with Ribosomal Protein L2 (rplB) selected as the target gene. *rplB* was selected because it is a well conserved single copy gene. Each organism has a single copy of the *rplb* and several copies in the HMP mock community overlap by one or more 21-mer. The HMP mock community consists of 22 human gut associated microorganisms with sequenced genomes listed in Table . The HMP mock community WGS datasets consisted of a total of one gigabase of 75 basepair Illumina sequences available from NCBI's Short Read Archive under accession numbers SRR172902, SRR172903 which were combined for all analyses. The annotated whole genome records for each organism were downloaded from GenBank and the Coding Sequence (CDS) annotation was extracted.

The combined dataset was quality filtered by trimming reads at quality score 2 as recommended by Illumina [2], a summary of the trimming results is contained in Table **??**. A Bloom filter was built from the combined dataset in 8 minutes 41 seconds with an estimated 0.001% false positive rate. The Bloom filter was built with a k-mer size of 21 using four hash functions and was one gigabyte in size.

To evaluate the coverage of the HMP mock community dataset the combined read set was mapped to the whole genome sequences for all the organisms using Bowtie2 (cite) summarized in Fig . Two organisms were removed from further analysis: Candida albicans was removed because it did not have a *rplB* annotation and Listeria monocytogenes was removed because only a hand full of reads mapped to it.

The starting vertices for Xander's search were selected by first aligning the reference genome's *rplB* sequences to the *rplB* model used for searching. The protein alignment was then used to align the nucleotide sequences for each reference genome's *rplB* sequence. The first 21 consecutive nucleotide residues aligned to the rplB model from each reference genome's *rplB* sequence along with the model position the first resiude occupied were used as the search start points. A total of 20 starting vertices were selected, one from each of the 20 organisms selected.

Xander was then run on the selected starts and a summary of the results are show in Table . Using the HMP mock community dataset with the start points selected Xander was able to assemble 20 *rplB* gene sequences with an average protein-protein identity to the reference gene of 90%. About half of the resulting sequences were partial assemblies which was due to cuts in the assembly graph caused by zero coverage for parts of the reference genomes which can be seen in the average coverage data in Table and in detail in

Supplemental 1. Since several of the organisms overlapped by at least one 21-mer the paths in the assembly graph crossed which combined with sequencing errors lead to the differences seen between the expected protein sequence and assembled contig.

Cuts in the assembly graph caused more problems than partial assemblies since if there is no path to the end of the model the A* search devolves in to an exhaustive traversal of the graph, something we specifically wanted to avoid. To deal with this two heuristic pruning methods were developed based on the log odds ratio comparing the probability the current path was generated by the HMM or a null model described in the Methods below. The effects of the different pruning methods can be seen in Fig **??**. By making the pruning heuristic more strict Xander considers far fewer nodes enabling faster searching and discarding non-target path segments.

The HMM for *rplB* was built using the seed sequences from the Functional Gene Repository (cite frontiers article). These seed sequences were used to build an HMM for each gene using a modified version of HMMER3 using the

```
--enone
```

option to disable sequence weighting (cite). HMMER3's default settings were tuned for detecting remote paralogs (cite person's phd thesis) where Xander is targeting close homologs. The default priors sometimes caused extensive searching of nonproductive insert and delete paths. HMMER3's source code was modified to change the prior probabilities for the $delete \rightarrow match$ and $insert \rightarrow match$ transitions to 95% probability, $delete \rightarrow delete$ and $insert \rightarrow insert$ transitions to 5% probability. The modifications to HMMER3 are available as a patch file against version 3.0.

Conclusions

Methods

Graph Structure

A novel graph structure was created that combined a De Bruijn Graph (DG) and HMM together in a single combined assembly graph (CG) for assembling genes of interest. A vertex in CG is created for every pair of vertices u, v in DG and HMM:

$$\forall (u,v)\, u \in \text{DG},\, v \in \text{HMM}$$

each vertex in CG combines the information in u and v. The total number of vertices in CG will be

$$|V(DG)| * |V(HMM)|$$

where V(G) is the vertex set of the graph G. Vertices in CG are generated as needed to reduce the memory requirements.

The edge set $E(CG)$ was defined as follows: suppose w_i, and $w_j \in V(CG)$ and were made by combining vertices v_i with u_i and v_j with u_j respectively with v vertices from the De Bruijn Graph and u vertices from the HMM.

$$\overrightarrow{w_i w_j} \in E(\text{CG}) \leftrightarrow \overrightarrow{v_i v_j} \in E(\text{DG}) \text{ and } \overrightarrow{u_i u_j} \in E(\text{HMM})$$

. That is, an edge exists in CG if and only if an edge connects the vertices they were created by combining. The weight of an edge $\overrightarrow{uv}$ in CG are the defined as sum of the transition and emission probabilities taken from the HMM.

$$w(\overrightarrow{uv}) = P_{transition}(u \rightarrow v) + P_{emission}(v)$$

The emission symbol is the unique character in the K-mer contained in v.

The De Bruijn Graph is constructed in nucleotide space regardless of whether the HMM is modeling protein or nucleotide sequences. When searching with a protein HMM the De Bruijn Graph is traversed in protein space by walking three nodes in any one direction at a time. The emission symbol then becomes the three unique characters at the end of the K-mer translated to protein. The codon reading frame is fixed based on the vertex chosen to begin graph traversal.

A K-mer matching heuristic was used to identify K-mers in the experimental reads that were present in a set of reference sequences for the gene of interest. The reference sequences were aligned to the HMM to annotate the K-mer matches with position in the HMM. The K-mer and the model position from the aligned reference, and implicit match HMM state were combined to form a search starting vertex in CG.

To find overlapping K-mers an exact seed matching approach was used. The reference sequences were broken up in to K-mers and stored in a hash table. Each read was then decomposed into K-mers that were then looked up in the hash of the references K-mers. For use with a protein HMM a seed length of $\lfloor K/3 \rfloor$ was used and input reads were translated in to all six reading frames. When assembling multiple genes of interest the reference sets can be combined together into a single hash so potential search starts can be identified in a single pass over the reads.

Assemblies in Xander are done using the A* search algorithm [3] for finding paths through the CG. The A* implementation in Xander was modified to find the highest scoring path instead of the lowest cost path. The set of goal vertices is defined as any vertex in the last model position that is in the match or delete state. The scoring function for a path P is defined as:

$$S(P) = \sum_{i=0}^{|P|} w(P_i P_{i+1})$$

where w(...) is the weight of the edge between two vertices in P.

The heuristic cost function for a vertex v is defined as:

$$h(v) = P_{v_{state} \to match} + \sum_{i=v_{modelposition}+1}^{M} P_{match \to match}(i, i+1)$$

the sum of the most likely state transitions from a v's state to the end of the model. Where P is the probability of the given transition and M is the length of the HMM.

To ensure The log-odds edge weights used by the heuristic score and scoring function were monotonic the following transformation is applied to every edge in CG:

$$w(\overrightarrow{uv}) = w(\overrightarrow{uv}) - max(P_{emission}(v_{HMMstate})) \quad u_{HMMstate} \neq i$$

Since this heuristic score will never overestimate the actual score it meets the admissibility criteria for A*, and additionally since the scoring function is monotonic a closed set is not required.

Since search starting vertices can be in any model position, not just the beginning of the model, an additional HMM is built from the reverse of the seed alignment used to build the forward HMM. Using this reverse model Xander can traverse paths in both directions from a starting vertex. The contigs generated by each search direction are reported seperately; a tool is included with Xander to combine the two contigs fragments in to a single contig.

A Kth Shortest Path algorithm [4] [5] to find multiple high scoring paths from a single starting vertex. Yen's algorithm was modified so that if an edge had been seen in any of the K-1 shortest paths it was not considered in subsequent candidate generation iterations. This ensured that each next shortest path generated contained at least one new vertex.

Xander implements a path pruning heuristics to remove paths that are unlikely to yield contigs that match the model well. When a node is opened the probability of the path to that point is calculated and compared to the probability of the path being generated from a null model (cite hmmer2 null model) and the node is discarded if the log odds ratio is below a threshold value θ. This heuristic pruning is done in addition to the A* search. The log-odds-ratio threshold can be tuned using a command line switch to balance the trade-off between sensitivity and running time.

Xander was implemented in the Java programming language and is distributed under the terms of the GPLv3 License available from https://github.com/rdpstaff/Xander-HMMgs. Xander uses a Bloom filter to store a compressed representation of the De Bruijn Graph. Xander consists of three primary tools; one for building a Bloom filter De Bruijn Graph, a tool for identifying starting positions, the core search tool. Support programs and scripts are also included with Xander for manipulating file formats, combining contig fragments and filtering Xander results.

Any of the tools can be replaced with a 3^{rd} party tool using difference heuristics as long as the resulting file matches the expected format. For example the starting vertex identification can be replaced with a 3^{rd} party tool so long as the resulting file contains the starting kmer and starting model position.

Authors contributions

Text for this section ...

Acknowledgements

Text for this section ...

References

1. Bloom BH: **Space/time trade-offs in hash coding with allowable errors**. *CACM* 1970, **13**(7):422–426.
2. Mann T: **Illumina Quality Scores** 2009, [https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0B-lLYVUOliJFYjlkNjAwZjgtNDg4ZC00MTIyLTljNjgtMmUzN2M0NTUyNDE3&hl=en_US].
3. Hart PE, Nilsson NJ, Raphael B: **A Formal Basis for the Heuristic Determination of Minimum Cost Paths**. *IEEE Transactions of Systems Science and Cybernetics* 1968, **4**(2):100–107.
4. Yen JY: **Finding the K Shortest Loopless Paths in a Network**. *Management Science* 1971, **17**(11):712–716.
5. Lawler EL: **A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem**. *Management Science* 1972, **18**:401–405.

Figures

Figure 1 - HMP Mock Community Read Mapping

Percentage of reads mapped (unnormalized) to the reference organism. For organisms with more than one chromosome only the reads mapping to the chromosome containing the rplB gene were counted.

Percentage of Reads Mapped Per Reference

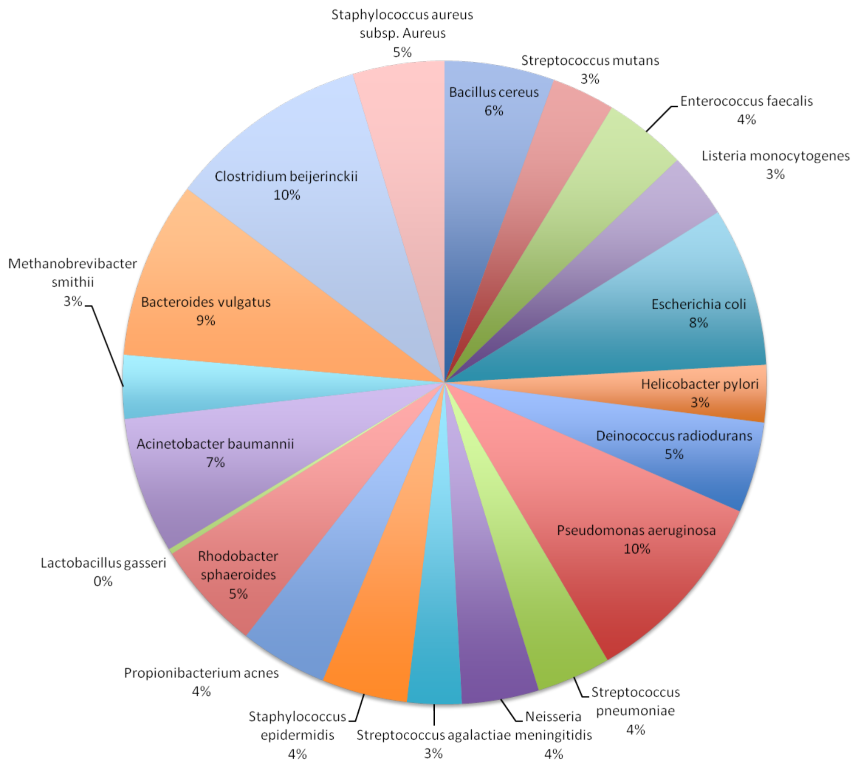
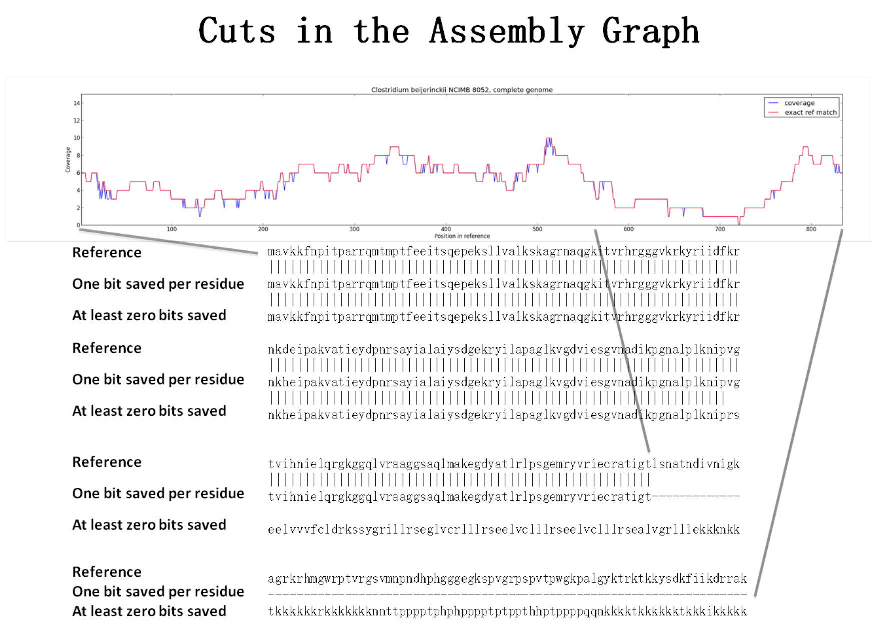


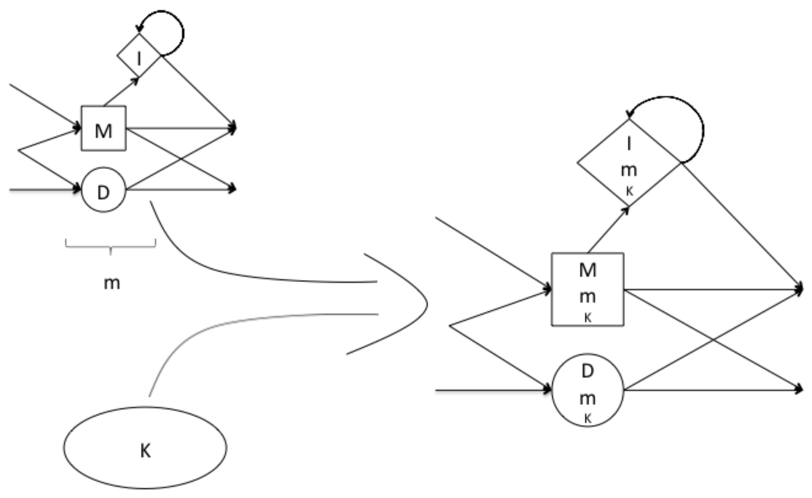
Figure 2 - Comparision of pruning heursitics

Effect of cuts in the assembly graph on resulting contig plus effectiveness of different heuristic path pruning techniques.



```
Reference                   mavkkfnpitparrqmtmptfeeitsqepeksllvalkskagrnaqgkitvrhrgggvkrkyriidfkr
                            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
One bit saved per residue   mavkkfnpitparrqmtmptfeeitsqepeksllvalkskagrnaqgkitvrhrgggvkrkyriidfkr
                            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
At least zero bits saved    mavkkfnpitparrqmtmptfeeitsqepeksllvalkskagrnaqgkitvrhrgggvkrkyriidfkr

Reference                   nkdeipakvatieydpnrsayialaiysdgekryilapaglkvgdviesgvnadikpgnalplknipvg
                            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
One bit saved per residue   nkheipakvatieydpnrsayialaiysdgekryilapaglkvgdviesgvnadikpgnalplknipvg
                            |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
At least zero bits saved    nkheipakvatieydpnrsayialaiysdgekryilapaglkvgdviesgvnadikpgnalplknipaprs

Reference                   tvihnielqrgkggqlvraaggsaqlmakegdyatlrlpsgemryvriecratigtlsnatndivnigk
                            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||
One bit saved per residue   tvihnielqrgkggqlvraaggsaqlmakegdyatlrlpsgemryvriecratigt-------------

At least zero bits saved    eelvvvfcldrkssygrillrseglvcrlllrseelvclllrseelvclllrsealvgrlllekkknkk

Reference                   agrkrhmgwrptvrgsvmnpndhphgggegkspvgrpspvtpwgkpalgyktrktkkysdkfiikdrrak
One bit saved per residue   ----------------------------------------------------------------------
At least zero bits saved    tkkkkkrkkkkkknnttpppptphphppppttptppthhptppppqqnkkkktkkkkkktkkkikkkkk
```

Figure 3 - Combined Graph Structure



Tables

Table 1 - HMP Mock Community Composition

Organisms in the HMP mock community and accession number of the GenBank record from which annotations were harvested.

† indicates genome records with incomplete annotations, annotations from another assembly of a synonmous strain (the accession number in parathesis) were used instead. * indicates genomes that were removed from the final analysis.

Organism Name	Strain
Streptococcus mutans	NN2025 DNA, complete genome
Listeria monocytogenes*	L99 serovar 4a, complete genome
Acinetobacter baumannii	ATCC 17978, complete genome
Acinetobacter baumannii	ATCC 17978 plasmid pAB1, complete sequence
Acinetobacter baumannii	ATCC 17978 plasmid pAB2, complete sequence
Actinomyces odontolyticus	ATCC 17982 Scfld020 & Scfld021 genomic scaffold, whole genome shotgun sequen
Actinomyces odontolyticus	ATCC 17982 Scfld020 genomic scaffold, whole genome shotgun sequence
Bacillus cereus	ATCC 10987, complete genome
Bacillus cereus	ATCC 10987 plasmid pBc10987, complete sequence
Bacteroides vulgatus	ATCC 8482, complete genome
Candida albicans*	SC5314 Assembly 21
Clostridium beijerinckii	NCIMB 8052, complete genome
Deinococcus radiodurans	R1 chromosome 1, complete sequence
Enterococcus faecalis	OG1RF chromosome, whole genome shotgun sequence
Escherichia coli	K12, complete genome
Helicobacter pylori	26695, complete genome
Lactobacillus gasseri	ATCC 33323, complete genome
Methanobrevibacter smithii	ATCC 35061, complete genome
Neisseria meningitidis	MC58, complete genome
Propionibacterium acnes	KPA171202, complete genome
Pseudomonas aeruginosa	PAO1, complete genome
Rhodobacter sphaeroides	2.4.1 chromosome 1, complete sequence
Rhodobacter sphaeroides	2.4.1 chromosome 2, complete sequence
Rhodobacter sphaeroides	2.4.1 plasmid A, partial sequence
Rhodobacter sphaeroides	2.4.1 plasmid B, complete sequence
Rhodobacter sphaeroides	2.4.1 plasmid C, complete sequence
Rhodobacter sphaeroides	2.4.1 plasmid D, complete sequence
Rhodobacter sphaeroides	2.4.1 plasmid E, complete sequence
Staphylococcus aureus subsp. aureus	USA300_TCH1516, complete genome
Staphylococcus aureus subsp. aureus	USA300_TCH1516 plasmid pUSA300HOUMR, complete sequence
Staphylococcus aureus subsp. aureus	USA300_TCH1516 plasmid pUSA01-HOU, complete sequence
Staphylococcus epidermidis	ATCC 12228, complete genome
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-06, complete sequence
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-05, complete sequence
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-04, complete sequence
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-03, complete sequence
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-02, complete sequence
Staphylococcus epidermidis	ATCC 12228 plasmid pSE-12228-01, complete sequence
Streptococcus agalactiae	2603V/R, complete genome
Streptococcus pneumoniae	TIGR4, complete genome

Table 2 - Summary of Xander results for the HMP Mock community

Output of running Xander on the HMP mock community WGS dataset. The forward and reverse searches for each reference are combined in a single row in the table with the total time for each direction's search reported. The fragment nats (probability log base e) represent the score A* was optimizing on while the bits saved reflects the probability the sequence comes from the HMM. The average coverage of the *rplB* region computed by bowtie mapping for each reference is also reported.

Reference Organism	Reference Length	Starting State	Protein Length	left fragment nats	left fra
Bacillus cereus	274	250	31	-14.744	
Streptococcus mutans	263	256	283	-14.55	
Enterococcus faecalis	277	270	284	-11.49	
Escherichia coli	274	271	61	-3.263	
Helicobacter pylori	277	271	66	-1.742	
Deinococcus radiodurans	276	271	282	-3.17	
Pseudomonas aeruginosa	274	271	41	-1.983	
Streptococcus pneumoniae	278	270	284	-15.275	
Neisseria meningitidis	278	271	23	-4.065	
Streptococcus agalactiae	263	270	284	-14.878	
Staphylococcus epidermidis	278	271	284	-1.652	
Propionibacterium acnes	279	271	282	-1.108	
Rhodobacter sphaeroides	280	271	281	-4.922	
Lactobacillus gasseri	277	193	2	-21.946	
Acinetobacter baumannii	243	238	284	-32.284	
Methanobrevibacter smithii	242	177	17	-22.914	
Bacteroides vulgatus	274	271	281	-2.035	
Clostridium beijerinckii	278	271	201	-2.702	
Staphylococcus aureus subsp. aureus	278	269	284	-5.063	

Additional Files

Additional file 1 — Sample additional file title

Additional file descriptions text (including details of how to view the file, if it is in a non-standard format or the file extension). This might refer to a multi-page table or a figure.