High-performance tools for precise microbiome characterization

A DISSERTATION SUBMITTED TO THE FACULTY OF THE UNIVERSITY OF MINNESOTA BY

Gabriel A. Al-Ghalith

IN PARTIAL FULFILLMENT OF THE REQUIERMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Advisor: Dan Knights

August 2018

Acknowledgements

I would like to extend special thanks to Dan Knights, Chad Myers, Ran Blekhman, and Cheryl Gale for serving on my graduation committee.

I would like to acknowledge my adviser, Dr. Dan Knights, for his mentorship and tireless support. I would also like to acknowledge the members and volunteers of the Knights Lab for their countless contributions, discussions, and use of my tools. I would like to extend special thanks to Lia Harrington for extensive code comments and guidance.

A special dedication to my loving father, Asad Al-Ghalith, and my mother, the late Linda Knight, may she rest in peace.

ABSTRACT

The microbiome, defined as the vast number of microorganisms inhabiting both human and non-human environments, has been associated with human disease as well as other important ecological phenomena. However, its quantitative study is complicated in part by measurement error and computational limitations, pointing to a need for more sensitive and reproducible DNA sequence analysis techniques. To this end, I have developed a variety of improved methods including a flexible short-read quality control pipeline, curated databases of marker genes and whole genomes, streamlined OTU picking software, and a high-throughput optimal aligner with taxonomy interpolation. Together, these methods represent advancements over traditional sequence analysis pipelines and may improve the quality of downstream statistical analyses.

Table of Contents

List of Tables iv
List of Figures v
INTRODUCTION 1
I. SHI7: A self-learning pipeline for multi-purpose short-read DNA quality control 6
II. NINJA-OPS: fast, accurate marker gene alignment using concatenated ribosomes 24
III. BURST enables mathematically optimal short-read alignment for big data 49
IV. Other contributions to comparative genomics and database generation/search. 68
Bibliography 88

List of Tables

Table 1. Results of SHI7 learning module. 15
Table 2. Overview of algorithmic optimizations in BURST. 57
Table 3. Alignment through the ages. 59
Table 4. BURST LCA with UNITE vs LTP. 71
Table 5. Comparison of kafan and gzip (GZ) modes with no preprocessing................................ 75
Table 6. Overview of algorithmic optimizations in aKronyMer.. 78

List of Figures

Figure 1. Microbiome workflow and selected contributions. 4
Figure 2. Basic quality control workflow. 10
Figure 3. Histogram of stitched reads. 13
Figure 4. Example of SHI7 QC effects on BLAST alignments. 14
Figure 5. Schematic of the NINJA pipeline. 29
Figure 6. Effects of singleton filtering on taxonomy. 34
Figure 7. NINJA-OPS vs USearch 8: speed. 38
Figure 8. NINJA-OPS speedup without denoising. 39
Figure 9. NINJA-OPS vs USearch 8: aggregate accuracy. 40
Figure 10. NINJA-OPS vs USearch 8: per-read accuracy. 41
Figure 11. Significantly expressed taxa in an IBD dataset. 42
Figure 12. Proportion of successful database matching. 43
Figure 13. NINJA-OPS vs USearch 8: genus-level concordance. 43
Figure 14. Burrows-Wheeler transform on concatenated sequences. 44
Figure 15. NINJA-OPS speedup per dataset with filtering. 46
Figure 16. Schematic of BURST. 56
Figure 17. Simplified example DNA alignment accelerated by BURST. 61
Figure 18. BURST vs Bowtie2: best hit and LCA taxonomy 63
Figure 19. BURST reveals species relatedness through shared genes. 64
Figure 20. Illustration of BURST "Capitalist" disambiguation vs LCA. 65
Figure 21. Practical example of CAPITALIST. 66
Figure 22. Phylogenetic ITS marker tree. 72
Figure 23. ITS tree distinguishes communities by body site. 72
Figure 24. Kafan speed vs compression. 75
Figure 25. Illustration of UTree's operation. 77
Figure 26. Rapid de novo amplicon phylogeny. 80
Figure 27. Snapshot of aKronyMer phylogeny from ~100,000 full genomes. 81
Figure 28. aKronyMer analysis of gut timeseries. 82
Figure 29. aKronyMer groups controls distinct from antibiotic users. 83
Figure 30. aKronyMer distinguishes between body sites. 84
Figure 31. Performance of bcov for strain detection in an example dataset. 86

INTRODUCTION

The microbiome is defined by the myriad microbes living in, on, and around us. Due to its intimate relationship with its environment, including its human hosts, it holds tremendous promise for unlocking secrets of human disease and wellbeing, as well as unraveling mysteries spanning from global warming to the evolution of virulence. However, as of the time of writing, the field of microbiome analysis remains in its infancy, enabled only recently by the emergence of ubiquitous high-throughput DNA sequencing technology. This technology has empowered us to probe the hitherto unfathomable diversity of microbes, the extent of which eluded us until a few decades ago despite centuries of earlier attempts to culture them in the laboratory. However, this new technology is not without problems of its own.

The abundance and diversity of the microbes comprising these microbiomes entails the collection of massive amounts of measurement data in the form of short DNA fragments, which must be cast into meaningful biological features prior to use in solving problems such as predicting clinical outcomes or ecological modeling. This arguably places the field within the burgeoning realm of “big data,” inheriting as such both the latter’s power and limitations. Adding to this complexity is the fact that these pieces of DNA may be ambiguous; they could be shared by multiple microbes, or exist in multiple places within the genome of a single microbe, or a combination of these. Further, our databases remain incomplete; there are many microbial species and genes we have yet to characterize, further complicating the use of microbes as meaningful features. These DNA readouts are also error-prone, introducing fuzziness into the database lookup problem in addition to meriting multiple steps of quality control to maintain analytical rigor.

Tools currently exist in the microbiome sciences to address these problems to some extent, but they suffer from numerous shortcomings. One issue is that there are many potential tools for each step, the order, use, and configuration of which requires expert knowledge to operate for best results. This increases the barrier to entry for performing even basic analyses, and also increases time investment in learning an operating multiple toolsets along with their interactions. Another distinct issue is in the quality of existing tools themselves, which may be cumbersome to install, slow to operate, or inaccurate in the results produced.

As highlighted at the top of Figure 1, there are 3 distinct stages to processing metagenomic communities for analysis. The first stage, quality control, aims to manicure raw DNA reads produced by sequencing instruments and get them into a usable, high-quality format for downstream processing. This task is currently accomplished with disparate tools such as Cutadapt [1] to remove sequencing adaptors, FLASh [2] to join overlapping paired end reads together, QIIME [3] or Trimmomatic [4] to filter reads containing low-quality bases, and potentially other programs or pipeline stages to convert the reads into formats usable by downstream tools (i.e. conversion to multiplexed FASTA format [3]).

The second stage in microbiome data processing involves microbial community profiling. Existing tools for this task exist but are proprietary ("black box" and costly), inaccurate, slow to run and install, or a combination of these [5]. The standard tool in QIIME [3] until very recently was the proprietary USearch [6] tool, and more recently changed to the slower but open-source sortMeRNA tool [7]. Other tools exist along this spectrum, trading off quality for performance, or both for open-source status. Examples include popular general purpose read mappers such as bowtie2 [8] as well as aligners such as BLAST [9].

The third stage in the pipeline comprises the umbrella of statistical analyses that are performed on the microbial census data. Common to many workflows is the investigation of microbial diversity metrics within and between samples (alpha and beta diversity, respectively). Phylogenetic analyses are also common for use in ecological distance metrics such as UniFrac [10]. Different statistical frameworks are appropriate for each type of analysis, such as longitudinal analysis or compositional data analysis.

It has been my goal in this PhD work to create and provide accurate, high-performance tools to tackle these issues – from initial quality control to microbiome characterization and analysis. Starting from the beginning of a typical microbiome analysis pipeline (Figure 1), I have created a pipeline called "SHI7" (pronounced "Shizen"), which is a quality control tool that automatically learns the characteristics of the raw data on which it is applied, enabling it to tune its own parameters and produce manicured, high-quality sequences as its output. I have also developed a pipeline for rough characterization of the microbiome based on heuristic sequence alignment called "NINJA-OPS," and due to what I believed to be lingering limitations of this method, I went on to create "BURST," a metagenome-scale exhaustive optimal (non-heuristic) short read aligner intended for multi-genome high-throughput alignment with robust error tolerance. Given the reliance of both of the latter tools on a reference database of known microbes, I have also produced software called "aKronyMer" that rapidly performs database-free microbiome characterizations, including alpha and beta diversity analysis as well as phylogeny generation, with only the raw short-read data as its input.

I have also produced new databases for use with my database-dependent tools, including a human- and murine-associated microbial genome database, data handling pipelines for database and phylogeny generation, and a novel fungal ITS-based fungal phylogeny for marker gene "mycobiome" studies. I pioneered a lossless data compression codec called "kafan." Using some of my other tools (SHI7, NINJA-OPS, aKronyMer, and BURST), I also created an ultra-fast, scalable *de novo* clustering/sequence variant isolation pipeline with the intention of dramatically speeding up the database-free identification of microbes from amplicon sequences. In addition, I have produced a tool that dramatically speeds up heuristic database-dependent taxonomic profiling called "UTree," which samples tiny "fingerprints" from short reads to identify candidate matches it then aggregates using a voting or lowest-common-ancestor scheme to infer the taxonomic composition of a community. Finally, I have made contributions to various downstream microbiome analysis projects that make use of these and other techniques, including statistics and machine learning.

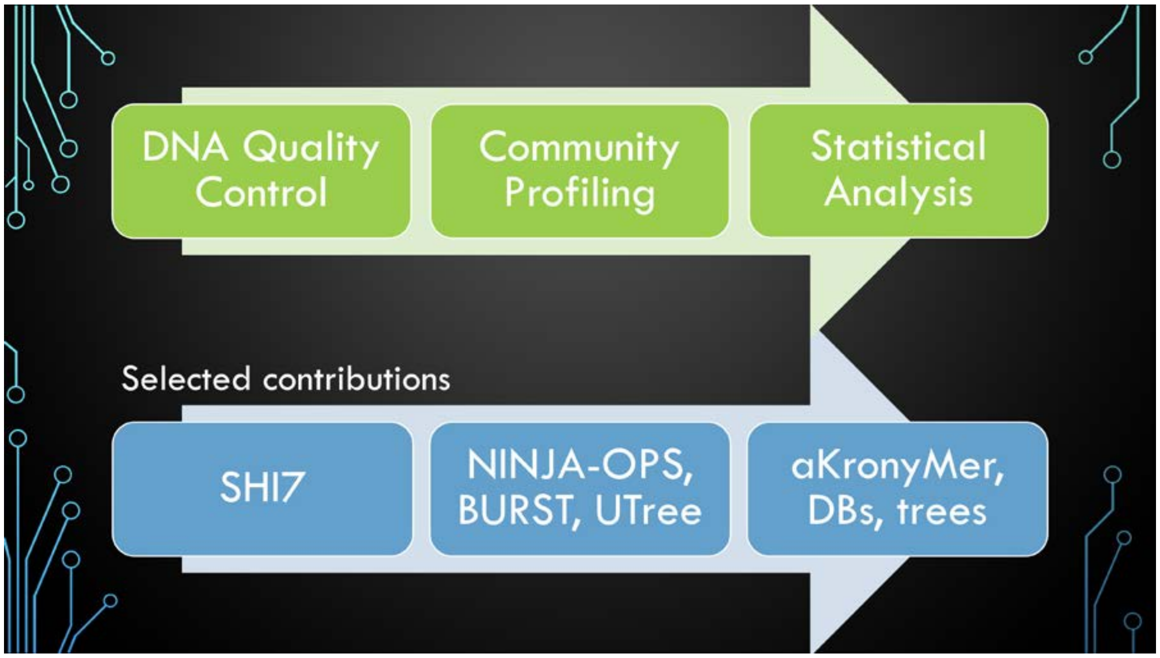


Figure 1. Microbiome workflow and selected contributions. A typical microbiome analysis workflow is shown in the top arrow, with each of my selected contributions positioned in the corresponding location in the lower arrow.

This thesis will focus on an important subset of my various tools, namely the quality control pipeline (SHI7), the amplicon characterization pipeline (NINJA-OPS), and the optimal metagenomic aligner (BURST). These were selected for inclusion based on relevance, publication status (all are published or in submission), and personal interest. It is my hope that you find this presentation of my work as beneficial and enriching as I did producing it. Onward!

Gabe

(References are provided at the end of this thesis.)

I. SHI7: A self-learning pipeline for multi-purpose short-read DNA quality control

(First published: Al-Ghalith, Gabriel A., et al. "SHI7 Is a Self-Learning Pipeline for Multipurpose Short-Read DNA Quality Control." *MSystems* 3.3 (2018): e00202-17.)

Summary.

Next-generation sequencing technology is of great importance for many biological disciplines. However, due to technical and biological limitations, the short DNA sequences produced by modern sequencers require numerous quality control (QC) measures to reduce errors, remove technical contaminants, or merge paired end reads together into longer or higher quality fragments. Many tools for each step exist, but choosing the appropriate methods and usage parameters can be challenging because the parameterization of each step depends on the particularities of the sequencing technology used, the type of samples being analyzed, and the stochasticity of the instrumentation and sample preparation. Further, end users may not know all of the relevant information about how their data were generated, such as the expected overlap for paired-end sequences or type of adaptors used, to make informed choices. This increasing complexity and nuance demands a pipeline that combines existing steps together in a user-friendly way, and when possible learns reasonable quality parameters from the data automatically. We propose a user-friendly quality-control pipeline called SHI7 (canonically pronounced "shizen"), which aims to simplify quality control of short read data for the end user by predicting presence and/or type of common sequencing adaptors, what quality scores to trim, whether the dataset is shotgun or amplicon sequencing, whether reads are paired end or single end, and whether pairs are stitchable, including the expected amount of pair overlap. We hope that SHI7 will make it easier for all researchers, expert and novice alike, to follow reasonable practices for short-read data quality control.

Importance. Quality control of high-throughput DNA sequencing data is an important but sometimes laborious task requiring background knowledge of the sequencing protocol used (such as adaptor type, sequencing technology, insert size/stitchability, paired-endedness, etc). Quality control protocols typically require applying this background knowledge to selecting and executing numerous quality control steps with the appropriate parameters, which is especially difficult when working with public data or data from collaborators who use different protocols. We have created a streamlined quality control pipeline intended to substantially simplify the process of DNA quality control from raw machine output files to actionable sequence data. In contrast to other methods, our proposed pipeline is easy to install and use, and attempts to learn the necessary parameters from the data automatically with a single command.

Introduction.

Next-generation sequencing (NGS) technology has become increasingly common across the biological sciences (1). The emergence of quality control (QC) software in tandem with the influx of NGS data highlights a need for measures to reduce noise, improve base-call quality, increase read length, filter out spurious sequences, split sequencing lanes by barcode for pooled sequencing runs, and otherwise improve the signal-to-noise ratio present within the large volume of data used to drive downstream analyses and make important decisions.

However, the increasing number of sequencing protocols available can make it difficult for a non-technical user to understand how to tune subtly different QC parameters when processing raw data. Different sequencing facilities use different techniques to shear longer DNA molecules into sufficiently short fragments for the sequencing instrument to process. Further, different DNA preparation kits may be used, and with different sequencing platform adaptors. There may be further points of difference as well, depending on the type of study performed. For instance, whereas shotgun sequencing methods attempt uniform coverage over all input DNA molecules, amplicon sequencing methods seek to minimize sequencing cost by targeted amplification of specific (marker) genes (2).

Common workflows

Despite numerous differences, the basic QC workflow for short-read sequencing has some common ground following sample preparation (Figure 2 shows a simplified schematic in context of microbiome sequencing). Essentially, the result of a typical paired-end sequencing run results in one or more pairs of FASTQ files containing raw sequence information for 100-300bp sequences along with quality scores representing the sequencing instrument's measure of confidence in the accuracy of each base call. This is useful because a rudimentary quality control procedure may read these scores and determine, through a set of logical parameters, how much of each individual short read to retain. These scores may also be used directly by downstream applications to weight the influence of particular bases in alignments, such as in recent versions of the popular bowtie2 aligner (3). In some cases, there are multiple samples contained within a single sequencing lane, each with its unique sample barcode, as multiplex marker gene sequencing (4), which subsequently must be demultiplexed in order for downstream analyses to differentiate among the various samples.

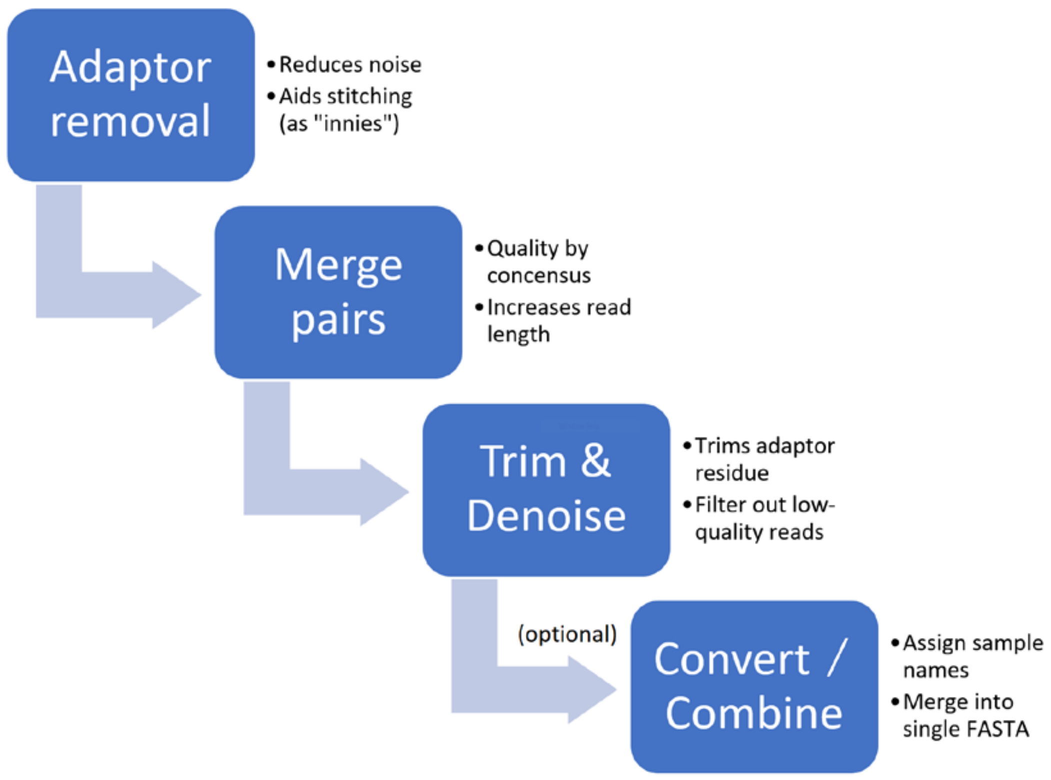


Figure 2. Basic quality control workflow. A linear schematic shows steps in a typical quality control procedure for marker gene (microbiome) data. The process flows from removing known technical artifacts to assembling short contiguous regions to trimming remaining contamination post-stitching and creating a final set (or optionally, single pooled file) of sequences in the desired format (FASTA or FASTQ). Notable exceptions to this procedure exist; for instance, pairs may not be stitchable depending on insert size for shotgun sequencing.

At an early stage in the QC process, it is essential to remove sequencing instrument adaptors introduced by the sequencing platform chemistry. Depending on the protocol used, some data may be more heavily contaminated with adaptors than others (5). The presence of adaptors can influence downstream analyses, including other QC steps, particularly if downstream global alignment (e.g. clustering) or end-to-end alignment (e.g. most short read mapping) will be performed. Because these adaptor nucleotides will not be present in reference sequences or databases, their presence in the reads will decrease alignment scores. In some cases, paired-end sequencing protocols allow for the paired ends to overlap one another, allowing the two reads in a pair to be merged or "stitched" together to form a single, often longer or higher quality, contig.

Throughout the region where both reads in a pair overlap, consensus quality determination is possible in cases of disagreement between pairs by retaining the higher quality of any two discordant bases. If the region of overlap is shorter than each individual read, this stitching also allows for the assembly of the two reads into a single longer contig. Merging pairs hence both improves quality in the region of overlap and extends the read, both of which improve the accuracy of downstream analysis (6). After stitching, any remaining poor quality regions, often located near the ends of the reads where the average base quality is lowest in general (6), can be trimmed until an acceptable quality is achieved throughout the read. Finally, these quality-controlled reads may be converted into the simpler FASTA format devoid of quality information, and in some domains such as microbiome analysis, samples may be pooled into a single file with sequence headers indicating which biological sample a read came from, using standards-compliant formatting (7).

Each of these typical steps in QC has received extensive study, and there exist a variety of tools for performing these steps. Under the reasonable baseline assumption that any such tool has a profile of strengths and weaknesses, it is not our goal here to perform extensive meta-analyses thereof, but instead to provide a user-friendly pipeline integrating a small number of well-known tools under a highly simplified interface. The primary contribution of our QC pipeline, SHI7 (pronounced “shizen”), is its ease of use. Specifically, SHI7 is trivial to install (either systematically with Conda, or as a portable standalone package with all dependencies included), easy to run from the commandline, and features a learning module which makes data-driven predictions for various QC parameters and presents these predictions for the user to run directly or tweak as desired. Importantly, although we do not posit that SHI7 will outperform any dedicated tool(s) or pipeline(s) on any well-defined short-read sequencing workflow, it is expected to perform reasonably well with little user intervention or expert knowledge across various workflows and data sources, especially when knowledge of sequencing and DNA preparation methodology is scarce or unreliable.

Results.

SHI7 was evaluated on publicly available sequence data from various sources including the Human Microbiome Project (HMP) (9). A random deep shotgun sequencing sample was selected, for simplicity, from the associated HMP data in the Sequence Read Archive for analysis: SRS014271 (Tongue). Without considering specifics of the sequencing platform, chemistry, adaptors, read lengths, library size, or paired-end status, the contents of the sequence archive were extracted into a new folder and the file called “singletons” was removed. SHI7 determined that there was some “TruSeqv2” adaptor contamination, which it removed. It determined these were not amplicon reads, but stitchable shotgun reads (just over 60% of the reads in this sample could be stitched together), which it performed. The distribution of stitched lengths resembled a normal distribution centered around 150, as the command-line debug output shows (Figure 3a). The final trimming removed fewer than 0.1 bases, on average, from either end of the stitched reads, and the average base quality throughout was 36.3 (very high) with an average read length of just under 150bp. Processing time was around 18 minutes for the 24GB pair of FASTQ files on 16 cores of a Xeon E7-4850 server over gigabit network SATA storage.

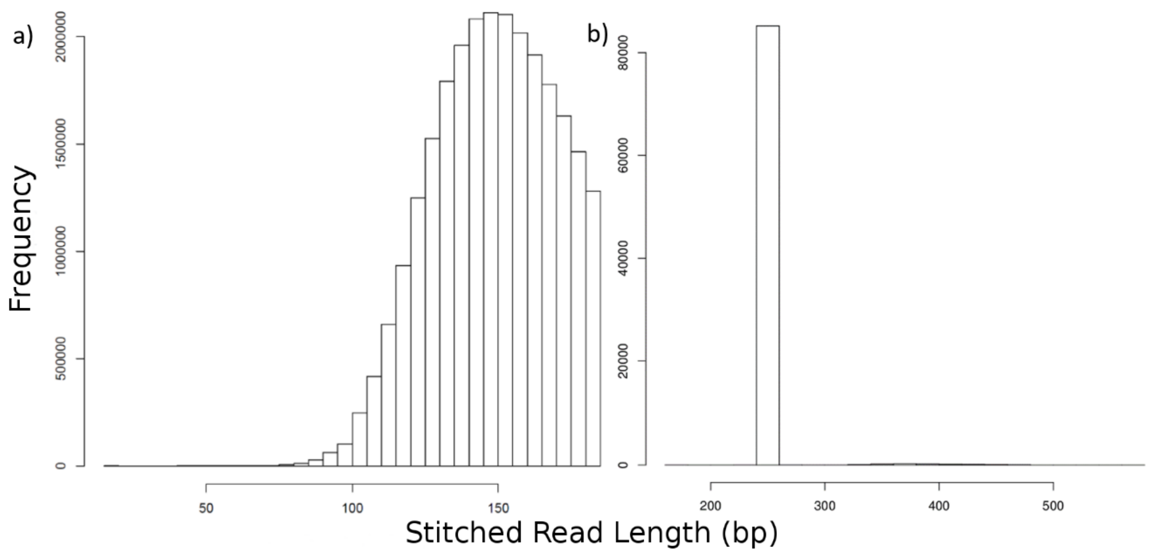
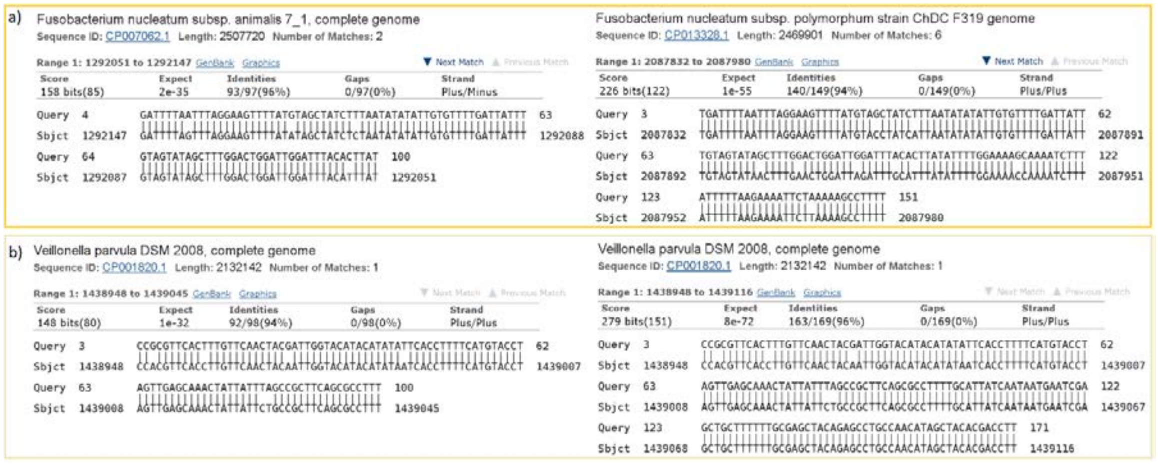


Figure 3. Histogram of stitched reads. These histograms show read length distribution in a single Human Microbiome Project (HMP) metagenomic sample (a) and 16S V4 Primate Microbiome Project (PMP) sample (b). (a) A shotgun metagenomic sample produces stitched contigs spanning range of lengths. The truncation after read lengths of 185bp is due to enforcing a minimum overlap length of 15 base pairs, which in a dataset consisting of 100bp reads is the maximum allowable length (100+100-15). Because the mean of this distribution is 148.6 and its standard deviation is 20.62, the coefficient of variation (CV) is 0.139, above the 0.1 threshold under which the data would be considered amplicon-like by default; the data are hence considered shotgun reads by SHI7. (b) A 16S amplicon sample produces a distinct histogram marked by high representation of certain contig lengths corresponding to target gene size, in this case 252 and 253 base pairs, and a much lower CV (mean = 254.4, SD = 15.7; CV = 0.062). Most residual longer reads match PhiX174, an Illumina control contaminant, and are later removed by SHI7 in "learning mode" by filtering out sequences within mean read length ± SD/2 in amplicon samples.

These results are interesting in that it was not obvious that there was specific adaptor contamination (albeit at low level), nor was it obvious that a majority of these 100bp paired-end shotgun reads overlapped throughout half their length. Downstream analyses also benefit from this procedure in straightforward ways. By way of simple illustration, matched sequences pre- and post-QC from the same HMP sample were submitted to nucleotide BLAST (10, 11) against the "nt" database. As would be expected, comparison between the stitched, quality-controlled sequences produced by SHI7 and the corresponding raw R1 reads shows higher e-values after

QC. In some cases, the post-QC reads received higher match identity for the same query (Figure 4a), and in other cases a different, and presumably more probable, highest scoring match (Figure 4b), although this particular query may still not resolve at the subspecies level with appropriately high similarity, possibly due to the lack of its specific matching reference strain in the database. This illustrates the potential implications of QC for pipelines and analyses relying on such “best-match” alignments. Higher scores (and/or lower e-values) are generally expected following stitching because longer reads have higher information content (it is less likely to match longer series of nucleotides by chance), and consensus quality scores in overlapping regions are likely to result in fewer technical errors, raising match identity. The benefit in using QC to trim adaptor contamination is expected to be higher in end-to-end alignments than in local alignments which implicitly perform soft-clipping.


a) Fusobacterium nucleatum subsp. animalis 7_1, complete genome
Sequence ID: CP007062.1 Length: 2507720 Number of Matches: 2

```
Range 1: 1292051 to 1292147 GenBank Graphics                    ▼ Next Match  ▲ Previous Match
 Score              Expect    Identities         Gaps           Strand
 158 bits(85)       2e-35     93/97(96%)         0/97(0%)       Plus/Minus

Query  4        GATTTTAATTTAGGAAGTTTTATGTAGCTATCTTTAATATATATTGTGTTTTGATTATTT  63
Sbjct  1292147  GATTTTAGTTTAGGAAGTTTTATATAGCTATCTCTAATATATATTGTGTTTTGATTATTT  1292088
Query  64       GTAGTATAGCTTTGGACTGGATTGGATTTACACTTAT  100
Sbjct  1292087  GTAGTATAGCTTTGGACTGGATTGGATTTACATTTAT  1292051
```

Fusobacterium nucleatum subsp. polymorphum strain ChDC F319 genome
Sequence ID: CP013328.1 Length: 2469901 Number of Matches: 6

```
Range 1: 2087832 to 2087980 GenBank Graphics                    ▼ Next Match  ▲ Previous Match
 Score              Expect    Identities         Gaps           Strand
 226 bits(122)      1e-55     140/149(94%)       0/149(0%)      Plus/Plus

Query  3        TGATTTTAATTTAGGAAGTTTTATGTAGCTATCTTTAATATATATTGTGTTTTGATTATT  62
Sbjct  2087832  TGATTTTAATTTAGGAAGTTTTATGTACCTATCATTAATATATATTGTGTTTTGATTATT  2087891
Query  63       TGTAGTATAGCTTTGGACTGGATTGGATTTACACTTATATTTTGGAAAAGCAAAATCTTT  122
Sbjct  2087892  TGTAGTATAACTTTGAACTGGATTAGATTTGCATTTATATTTTGGAAAACCAAAATCTTT  2087951
Query  123      ATTTTTAAGAAAATTCTAAAAAGCCTTTT  151
Sbjct  2087952  ATTTTTAAGAAAATTCTTAAAAGCCTTTT  2087980
```

b) Veillonella parvula DSM 2008, complete genome
Sequence ID: CP001820.1 Length: 2132142 Number of Matches: 1

```
Range 1: 1438948 to 1439045 GenBank Graphics                    ▼ Next Match  ▲ Previous Match
 Score              Expect    Identities         Gaps           Strand
 148 bits(80)       1e-32     92/98(94%)         0/98(0%)       Plus/Plus

Query  3        CCGCGTTCACTTTGTTCAACTACGATTGGTACATACATATATTCACCTTTTCATGTACCT  62
Sbjct  1438948  CCACGTTCACCTTGTTCAACTACAATTGGTACATACATATAATCACCTTTTCATGTACCT  1439007
Query  63       AGTTGAGCAAACTATTATTTAGCCGCTTCAGCGCCTTT  100
Sbjct  1439008  AGTTGAGCAAACTATTATTCTGCCGCTTCAGCGCCTTT  1439045
```

Veillonella parvula DSM 2008, complete genome
Sequence ID: CP001820.1 Length: 2132142 Number of Matches: 1

```
Range 1: 1438948 to 1439116 GenBank Graphics                    ▼ Next Match  ▲ Previous Match
 Score              Expect    Identities         Gaps           Strand
 279 bits(151)      8e-72     163/169(96%)       0/169(0%)      Plus/Plus

Query  3        CCGCGTTCACTTTGTTCAACTACGATTGGTACATACATATATTCACCTTTTCATGTACCT  62
Sbjct  1438948  CCACGTTCACCTTGTTCAACTACAATTGGTACATACATATAATCACCTTTTCATGTACCT  1439007
Query  63       AGTTGAGCAAACTATTATTTAGCCGCTTCAGCGCCTTTTGCATTATCAATAATGAATCGA  122
Sbjct  1439008  AGTTGAGCAAACTATTATTCTGCCGCTTCAGCGCCTTTTGCATTATCAATAATGAATCGA  1439067
Query  123      GCTGCTTTTTTGCGAGCTACAGAGCCTGCCAACATAGCTACACGACCTT  171
Sbjct  1439068  GCTGCTTTTTTGCGAGCTACAGAGCCTGCCAACATAGCTACACGACCTT  1439116
```

Figure 4. Example of SHI7 QC effects on BLAST alignments. This comparison illustrates BLAST alignments before and after SHI7 quality control on the same reads of an HMP shotgun sample. (a) shows the SHI7-QC read (right) achieving a different best-scoring alignment than the non-QC read (left) despite the former’s slightly lower identity (SHI7 alignment: 94% and e-val 1e-55, non-QC: 96% and e-val 2e-35). The same reference as in the non-QC alignment also appears for the SHI7-QC read with the same identity (96%) and 90% coverage, but in third place. (b) shows a different alignment; here the SHI7-QC read (right) finds the same best match as the non-QC read (left), but at higher identity and lower e-value (SHI7: 96% and e-val 8e-72, non-QC: 94% and e-val 1e-32). The case demonstrated by (a) occurs less frequently than (b) for this test data, but may have additional important implications for pipelines relying on "best match" read mapping.

SHI7 was also evaluated on 16S amplicon data. A sample from the Primate Microbiome Project (12) was run through “learning mode,” which determined it contained Nextera adaptor contamination, stitchable reads (over 97% stitched successfully) with combined lengths concentrated around 252 base pairs. SHI7 detected that the sample was an amplicon sample due to the low coefficient of variation in the stitching sizes (<0.03). This in turn informed SHI7 to impose a length minimum and maximum near the peak to reduce contaminant reads (Figure 3b), which we found in this sample to be primarily PhiX174 control sequences that this particular Illumina sequencing technology is known to introduce. The final run took 6 seconds for this sample, consisting of a 56MB pair of input FASTQ files.

The learning module was also tested on a variety of datasets to ascertain whether it was able to discover reasonable QC parameters across sequencing protocols, technologies, and data sources. As summarized in Table 1, SHI7 in each case learns parameters which agree with expectations given background knowledge of the datasets. Some exceptions to our expectations were produced by the learning module, including the ability to stitch reads unexpectedly in the HMP tongue shotgun data, no detection of expected Nextera adaptors in the mouse tutorial data, and detection of TruSeq3-2 adaptors when ScriptSeq adaptors were used for the RNAseq data. However, these decisions were justifiable when each case was investigated carefully: the majority of HMP tongue shotgun reads do indeed stitch appropriately, the mouse tutorial data had its forward adaptors already removed and used longer amplicons without bleed-through of reverse adapters, and the appropriate ScriptSeq adaptors were actually contained within Trimmomatic's version of the TruSeq3-2 adaptor library.

Table 1. Results of SHI7 learning module. SHI7's learning module produces meaningful QC parametrizations on internal and publicly available datasets.

Dataset	Availability	Learned parameters
HMP tongue; shotgun (Illumina HS PE TS2)	Public, SRS014271(9)	--adaptor TruSeq2 --flash True --allow_outies False --filter_qual 36 --trim_qual 36
Immigrant Microbiome Project; amplicon (mixed Illumina PE Nextera)	Internal	--adaptor Nextera --flash True --allow_outies False --filter_qual 34 --trim_qual 32
Small bowel aspirate; amplicon (Illumina PE Nextera)	Internal	--adaptor Nextera --flash True --allow_outies False --filter_qual 36 --trim_qual 34

Primate Microbiome Project stomach; amplicon (Illumina PE TS2)	Internal	--adaptor TruSeq2 --flash True --allow_outies False --filter_qual 36 --trim_qual 33 --min_overlap 239 --max_overlap 269
Longitudinal diet study; shotgun (Illumina HS SE Nextera)	Internal	-SE --adaptor Nextera --flash False --allow_outies False --filter_qual 36 --trim_qual 34
HMP stool; amplicon (454 SE)†	Public(17), stool	-SE --adaptor None --flash False --allow_outies False --filter_qual 34 --trim_qual 31
Mouse tutorial; amplicon (Illumina PE Nextera)	Public(18)	--adaptor None --flash True --allow_outies False --filter_qual 34 --trim_qual 34 --min_overlap 154 --max_overlap 172
Irritable bowel syndrome cohort; shotgun (Illumina HS SE Nextera)	Internal	-SE --adaptor Nextera --flash False --allow_outies False --filter_qual 37 --trim_qual 35
Human microbiome; RNAseq (Illumina HS PE ScriptSeq)	Internal	--adaptor TruSeq3-2 --flash True --allow_outies False --filter_qual 39 --trim_qual 36

†*sff_extract -Q* was used for the initial conversion of .sff to .fastq format (19)

Discussion.

As mentioned previously, these results are not intended to illustrate any advantage in quality or QC performance SHI7 might be expected to achieve over alternative quality control pipelines. This is particularly the case compared to pipelines developed and tested in the context of well-defined workflows utilizing known extraction, amplification, size selection, and sequencing protocols, where each step is carefully translated into the appropriate QC parameters and tested with a variety of tools against mock datasets produced by the same. SHI7 is not intended to replace such highly-specialized workflows. Furthermore, regardless of the validity of the selected parameters, we strongly caution against any blind application of bioinformatics analysis tools without a working comprehension of the concepts underlying biological sequence processing, as this may lead to erroneous analyses.

Instead, the simplicity and convenience afforded by SHI7 is our primary focus, as our results imply that SHI7 is capable of achieving reasonable quality control without need for the user to know or supply any procedural parameters in advance. Further, the resulting merged FASTA file (if this output mode is used) is immediately compatible with OTU picking solutions such as NINJA-OPS and others (3, 13, 14), whose outputs are in turn compatible with statistical analyses in standard metagenomics pipelines including QIIME (15). The intention, then, is for SHI7 to bring users, in most cases, from raw FASTQ data to sequence analysis capability in a single step without needing to know any details of the sequencing procedure or technology, while still providing reasonable quality control.

We believe the ease-of-use, speed, flexibility, and intelligent learning capabilities of SHI7 will be of benefit to novices and experts alike, particularly when dealing with FASTQ data from various sources where the details underlying the sequencing protocol are not well known in advance. For use with both shotgun and 16S sequencing projects, as well as on data from collaborators or online repositories which are often accompanied with sparse methodological detail, we find that

SHI7 reduces time spent adjusting and exploring settings for QC parameters, while providing consistent quality control that mirrors standard practices.

SHI7 is available as free and open-source software under the AGPLv3 license. Dependencies (trimmomatic and FLASH) are distributed with the compatible GPLv3 license. The software is freely available for multiple operating systems on Github (16): https://github.com/knights-lab/shi7 (see release page for portable package). This GitHub page also includes tutorials, example use cases, and a frequently-monitored interface for requesting new features and filing bug reports. SHI7 may be also installed using Anaconda: https://anaconda.org/knights-lab/shi7

Methods.

We have developed a pipeline that integrates standard QC practices with a learning module that automatically tests for and optimizes parameters based on the sequence data itself. Each phase of the pipeline is aware of the parameters selected in other phases and optimized accordingly. This pipeline is applicable across domains, handles a range of short-read sequencing lengths, and can automatically determine whether reads are likely to be paired-end, whether pairs can be stitched and with how much overlap, whether the sequences derive from amplicon or whole-genome shotgun sources, what adaptors were used on the reads (Illumina platform), what aggressiveness to perform quality trimming, and how to transform sample/lane names into standards-compliant FASTA labels if a combined FASTA is desired (or the filenames of the final FASTQ files, if FASTQ is desired as the output).

To accomplish these objectives, SHI7 incorporates just two well-known, lightweight programs: Trimmomatic (5) and FLASH (6), and introduces its own high-performance error-correcting barcode demultiplexer (gotta_split) and quality control (shi7_trimmer) modules. These new modules are written in C, and their performance saturates with the write speeds of modern hard drives. The gotta_split demultiplexer features ambiguous base support in the barcodes (including barcodes beginning with a series of “N” bases), supports staggered barcodes, barcodes occurring elsewhere in the reads than the beginning (disabled by default), and error-correction up to a user-specified number of mismatches, including the ability to report whether the specified number of mismatches could cause one adaptor to be mistaken for another. The shi7_trimmer module implements numerous simple trimming methods, uses different quality cutoffs for trimming either end of the read, and filters for length and average PHRED score.

The primary reason for including the shi7_trimmer module is its variable-length sliding quality floor mode for trimming the ends of reads. This functionality is currently not available in Trimmomatic. Unlike averaged quality scores, which are commonly used in sliding-window-based quality control, the floor function will not tolerate the presence of even a single base of lower quality than a given threshold, regardless of the quality of other bases in the window. Only once all bases in in the sliding window are above this threshold will it stop trimming from the ends of a given read. This behavior is especially useful for removing residual adaptor contamination following full-read-length stitching (as for bacterial 16S V4 amplicon sequencing), where the amplified region is shorter than the technical read length, causing the resulting sequences to contain part of the opposing adaptor. Due to the merging process, these artefactual portions will likely produce poor base matching (as the forward and reverse adaptors are essentially being overlaid), which the FLASH software reports in the form of very low quality scores, allowing shi7_trimmer to remove them from the read.

Interactive mode (manual selection of parameters)

SHI7 can be run without a learning mode, in which case it becomes a simplified wrapper script for standard QC practices with sensible defaults for most paired-end adaptor-free workloads assuming the possibility of stitching. Each stage of the pipeline can be turned off or on with a single command line flag (including disabling stitching, paired end mode, combining results into FASTA, adaptor trimming with specified adaptors, or splitting FASTQ files into samples based on barcodes). Although easy to use, this mode of operation is more suitable for users with a knowledge of their data generation processes, as it depends on the correct assumptions about the data (e.g., are reads paired and if so, are they amenable to stitching? Were adaptors removed and if not, which ones?). It also exposes a few commandline options for each of these stages, allowing for flexible, but not overwhelming, parameter customization. Individual steps in the pipeline can also be run separately without the python wrapper, including the two C modules (gotta_split and shi7_trimmer) for even more flexibility.

Learning mode (automatic)

The SHI7 pipeline in "learning" operation mode first applies heuristics to determine basic features of the data. These include PHRED scale determination, whether reads are paired-end (using filename string pattern checks), and approximate read lengths prior to adaptor removal or quality trimming. The presence of an oligonucleotide file (text file implementing MOTHUR format paired barcodes, typically "oligos.txt") signals to split a single pair of FASTQ files into multiple separate FASTQ files named by corresponding sample ID (8). The software also determines whether reads stitch and if so at what level of overlap if reads are detected to be amplicons. The quality scores at which to filter and trim reads are also learned by profiling the distribution of base qualities in a sampling of reads.

Learning mode operation

In its “learning mode,” SHI7 runs a learning pass on a subsampled selection of the reads across files (up to 1000 reads per fastq file), gathering data by running various combinations of settings, and reports its best estimation for these parameters to the user before proceeding with the full QC pipeline using these options. Specifically, the learning module first subsamples each FASTQ file to 1000 sequences, recording sequence lengths. To determine whether pairs are present, if an even number of FASTQ files exist, these are run through basic pattern recognition to identify if a known pattern exists across all files that successfully distinguishes pairs (“R1/R2”, “.1/.2”, “_1/_2” are checked for among others in a growing list of patterns). If pairs are detected, the subsequent adaptor detection stage proceeds in paired mode (otherwise, unpaired mode is used). The adaptor detection runs a separate instance of trimmomatic using each one of its included repertoire of adapters and picks the adaptor set that produces the smallest output filesize.

Following adaptor detection and removal, if paired-end reads are present, stitching is attempted using generous defaults (minimum overlap of 10 bases and maximum overlap of 700 bases). A histogram of resulting overlaps is generated with FLASH, allowing SHI7 to determine whether a reasonable proportion of reads reliably stitch (25% or more, by default), and further assessing whether coefficient of variation (CV) in stitched read lengths is less than 0.1, signaling significant DNA fragment length uniformity indicative of amplicon or amplicon-like reads. This allows SHI7 to bound the minimum and maximum overlap considered in the stitching process over an expected range (set by default to +/- twice the standard deviation). This “bounded stitching” itself serves as an additional quality control agent by eliminating falsely-stitched reads that result from unexpectedly long or short contaminants, and reducing rare instances of tied equal scoring overlaps by restricting to overlaps in the expected range.

The final trimming quality parameters are determined by scanning the reads again (after all previous QC steps have been completed) to determine average quality as well as “terminus” quality (quality scores averaged over the first and last 10 bases of each read). The learning module recommends a per-read average quality filter equal to the average base quality throughout the dataset, and produces a recommendation for end trimming between this value and the average “terminus” quality calculated previously.

Limitations

Notable limitations of this software include reliance on Trimmomatic’s adaptor collection for detecting explicit adaptor contamination, although any adaptors can be added to this collection by the user if this information is known through the corresponding Trimmomatic interfaces. The pipeline requires both Python 2.7+ (including 3.x, for the wrapper and learning module) and Java SE (for trimmomatic’s adaptor removal). Minimum runtime requirements include a 64-bit operating system (Windows, Linux, or OSX), 4GB RAM (with 1 thread; add 4GB per additional thread used), and free disk space equal to about twice the original size of the data being processed. FASTQ files must not contain entries split across lines (word wrap), and paired ends (if used) must be in split-file format. FASTQ files appearing in interleaved format (both pairs appear in the same file) are not explicitly supported in paired-end mode, but will still be processed normally as though they were single-end reads. Compressed fastq files are not supported; the user must currently extract these files to use them with SHI7 such as with the command “gunzip *”, but support for compressed formats are planned for a future release. If demultiplexing is desired, a text file named “oligos.txt” is required in the input directory in MOTHUR format; there is no automatic detection of barcodes for demultiplexing.

Data Availability

All code used in SHI7 is available in its repository located at https://github.com/knights-lab/shi7. External test datasets are available from their respective citations in Table 1; our own validation datasets are made publically available in the Sequence Read Archive with accession SRP132961.

Acknowledgements

We would like to thank Dr. Ajay Israni and Amutha Muthusamy for providing microbial RNAseq data with technical sequencing details for use in our validations.

(References are provided at the end of this thesis.)

II. NINJA-OPS: fast, accurate marker gene alignment using concatenated ribosomes

(First published: Al-Ghalith, Gabriel A., et al. "NINJA-OPS: fast accurate marker gene alignment using concatenated ribosomes." *PLoS computational biology* 12.1 (2016): e1004658.)

Summary.

The explosion of bioinformatics technologies in the form of next generation sequencing (NGS) has facilitated a massive influx of genomics data in the form of short reads. Short read mapping is therefore a fundamental component of next generation sequencing pipelines which routinely match these short reads against reference genomes for contig assembly. However, such techniques have seldom been applied to microbial marker gene sequencing studies, which have mostly relied on novel heuristic approaches. We propose NINJA Is Not Just Another OTU-Picking Solution (NINJA-OPS, or NINJA for short), a fast and highly accurate novel method enabling reference-based marker gene matching (picking Operational Taxonomic Units, or OTUs). NINJA takes advantage of the Burrows-Wheeler (BW) alignment using an artificial reference chromosome composed of concatenated reference sequences, the "concatesome," as the BW input. Other features include automatic support for paired-end reads with arbitrary insert sizes. NINJA is also free and open source, and implements several pre-filtering methods that elicit substantial speedup when coupled with existing tools. We applied NINJA to several published microbiome studies, obtaining accuracy similar to or better than previous reference-based OTU-picking methods while achieving an order of magnitude or more speedup and using a fraction of the memory footprint. NINJA is a complete pipeline that takes a FASTA-formatted input file and outputs a QIIME-formatted taxonomy-annotated BIOM file for an entire MiSeq run of human gut microbiome 16S genes in under 10 minutes on a dual-core laptop.

Importance.

The analysis of the microbial communities in and around us is a growing field of study, partly because of its major implications for human health, and partly because high-throughput DNA sequencing technology has only recently emerged to enable us to quantitatively study them. One of the most fundamental steps in analyzing these microbial communities is matching the microbial marker genes in environmental samples with existing databases to determine which microbes are present. The current techniques for doing this analysis are either slow or closed-source. We present an alternative technique that takes advantage of a high-speed Burrows-Wheeler alignment procedure combined with rapid filtering and parsing of the data to remove bottlenecks in the pipeline. We achieve an order-of-magnitude speedup over the current state of the art without sacrificing accuracy or memory use, and in some cases improving both significantly. Thus, our method allows more biologists to process their own sequencing data without specialized computing resources, and it obtains more accurate taxonomic annotation for their marker gene sequencing data.

Introduction.

The advent of next-generation sequencing technologies, combined with major advances in molecular and bioinformatics techniques, have enabled rapid growth in the culture-independent sequencing of amplified marker genes (amplicons) from environmental microbial communities. The major benefit of amplicon sequencing is that it allows reasonable resolution of taxonomic composition in these communities at a fraction of the cost of deep metagenomic sequencing. Once these sequences are generated, a common analysis approach is to bin them by sequence identity into operational taxonomic units (OTUs)[1–4]. For environments containing a large fraction of novel taxa, one must rely on unsupervised (“de novo”) clustering of amplicons to convert the raw reads to features representing organisms belonging to distinct evolutionary clades. On the other hand, in habitats with mostly well-characterized microbes, we have the option of matching the generated amplicon sequences to reference databases containing example marker genes from known taxa [5]. A hybrid approach may also be used, where sequences are first compared to a reference database, with subsequent de novo clustering of those that failed to match. As the number of published culture-independent amplicon-based surveys of microbial communities continues to grow, our ability to rely on reference sequences also increases. However, although the crucial analysis step of mapping generated amplicons to reference marker genes has received much attention from the microbial bioinformatics field, with a variety of solutions proposed [6–10], there is much room for improvement in terms of speed, accuracy, memory footprint, and openness of code. NINJA, our portable, open-source OTU picking pipeline, realizes these goals.

Originally conceived as a means to make data more compressible, the Burrows-Wheeler transform (BWT) [11] is a lossless, reversible transformation that effectively positions series of like characters close to each other in a way that can easily be undone to recover the original data. It involves creating a circular suffix array, sorting the final column lexicographically, and storing that column as the transformed data for later compression. This algorithm also has the interesting property of enabling rapid substring search, with O(1) order of growth in finding exact string matches. As long as there is an efficient indexing scheme that stores the indices of the transformed bases into the original string, the BWT can be used for fast database substring search amounting to binary searching (or looking up via rank matrix) the transformed reference string representation and mapping back to the original, and has hence been employed in a number of commonly used DNA alignment tools [11–14]. Although these tools are approximate methods due to the high additional computational cost of performing optimal local or global alignment search when mismatches occur, they are generally fast and widely used in the genome-enabled research community (http://bowtie-bio.sourceforge.net/bowtie2/other_tools.shtml).

Here we demonstrate that BWT-enabled DNA alignment can be effectively used for accurate and fast assignment of marker-gene sequences to a reference database. We present the NINJA-OPS pipeline utilizing several novel contributions to achieve an order of magnitude speedup and higher accuracy when compared to commonly used approaches (or up to two orders of magnitude when combined with denoising). To test the accuracy and efficacy of our approach, we perform closed-reference OTU-picking on a wide range of biological data sets from varied environments.

This is accomplished by the NINJA core tools and an optimal aligner which produces a BLAST-style %ID for each query sequence against the reference sequence chosen by the OTU picking method. Speed was assessed as the elapsed time from parsing the input FASTA file until the alignment (against a pre-generated database) has terminated, but it may be useful to note that NINJA also significantly speeds up the subsequent steps of tallying reads, incorporating taxonomic annotations, and producing an OTU table in sparse BIOM 1.0 format, as well as other steps prior to the alignment such as reverse complementing and trimming reads. Hence, the NINJA pipeline accelerates many stages of the OTU-picking pipeline beyond the alignment step.

Methods.

Pipeline overview.

The pipeline follows three stages: filtering, aligning, and parsing. After forming the concatenated reference string, called the “concatesome,” from the individual references, NINJA applies a powerful filtering step which uses a 3-way radix quicksort on string pointers to rapidly de-duplicate millions of reads, construct a sample dictionary, and output a streamlined filtered FASTA file and sample dictionary (Figure 5). The program implements this lossless filtering approach as well as a lossy variant, making use of singleton filtering as well as statistical profiling over the entire set of reads to exclude reads with a user-defined number of duplicates or rare segments (k-mers) appearing below a user-defined threshold of prevalence. The lossy filtering is intended to independently identify reads with probable read error, and speeds up the resulting alignment by excluding such reads from the BWT aligner. This adds an additional speedup because BWT string matching spends a disproportionate amount of search effort to align erroneous or low-identity reads.

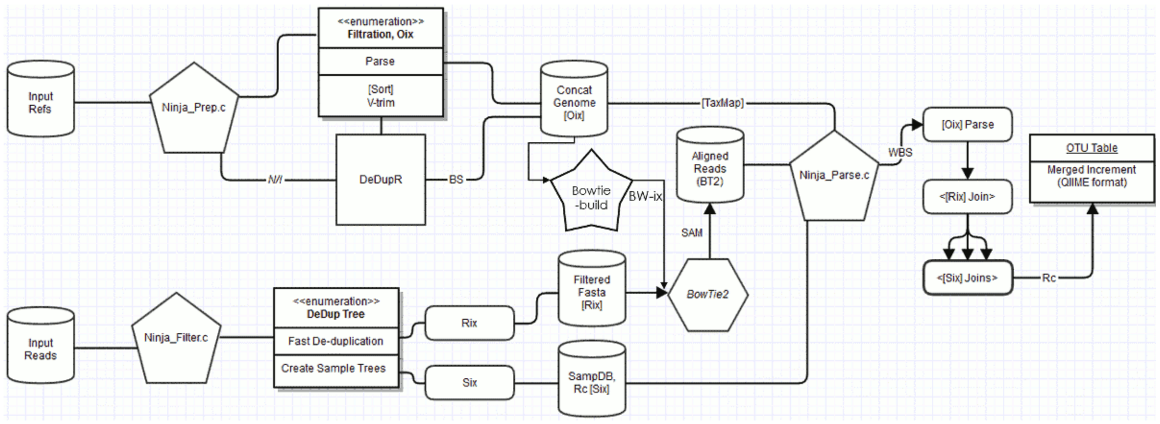


Figure 5. Schematic of the NINJA pipeline. NINJA core programs are represented by pentagons, data files by cylinders, processes within a program as lists, rounded rectangles as index operations, and other swappable programs by other shapes. The entire upper-left branch of the schematic (from input references to bowtie-build and TaxMap) does not need to be performed if using an existing database, such as that supplied with NINJA. The python wrapper encompasses the remaining two branches (bottom and right) for convenience. In general, Ninja_prep prepares the concatesome, Ninja_filter prepares the reads for alignment, bowtie2 (or any BWT-enabled aligner) performs the alignment, and Ninja_parse merges the various pieces into a complete OTU table.

The NINJA filter step also performs reverse-complementing and sequence length trimming at the same time as the other filtering steps. Because of this simultaneous multi-step filtering, no intermediate files are created prior to the alignment stage, and all filtering steps are performed rapidly in optimized C code on data structures already in memory. This takes a fraction of the time used by other filtering pipelines which perform sequential operations often written in general-purpose scripting languages and generate numerous large intermediate files after each step. Using the base NINJA filter parameters, the entire filtering process itself takes approximately 10-20% of the time it takes to align the resulting filtered file when using all optional filtering steps.

Next, the filtered reads are aligned against a reference database containing the (the concatesome) via any BWT-derived short read aligner such as BWA[11], Bowtie[13]/Bowtie2[12], hpg-aligner [15], SOAP2 [16] -- or, more broadly, any read aligner whatsoever capable of outputting to headerless SAM format[17] and suppressing unmatched input reads. Utilizing SAM is much faster than BAM (binary compressed SAM) after deduplication, as the alignment step is not I/O bound and the overhead of BAM's additional compression/decompression step can be significant. We have chosen to standardize NINJA around Bowtie2 for our tests and publish the command line options for Bowtie2 as we have found it to be suitable for the purposes of BLAST-identity-based OTU picking. Following alignment, the resulting SAM file is fed to the NINJA parsing step, which takes in the sample dictionary metadata as well as an optional taxonomy map to rapidly re-assign each de-duplicated read to the biological sample(s) in which it originally occurred, add taxonomy annotation to each picked OTU, bin all reads by their matched OTU into a sample-by-OTU matrix (OTU table), and output the result in sparse BIOM 1.0 format (or a tab-delimited legacy QIIME format that can be read by both human and machine). This can also be incorporated into an open-reference OTU-picking pipeline.

Burrows-Wheeler transform.

The BWT has received a lot of attention in the alignment of short reads to a reference genome, and now enjoys routine use in clinical and other settings as a well-vetted technique for mapping short DNA reads to a longer reference, where it is known as Burrows-Wheeler alignment. The BWT is based on the principle that a long string of text can be reversibly transformed to reduce the complexity of substring queries to effectively two binary searches into the transformed representation of the original string, which is then converted back to indices into the original reference string with a short walk-back (the BW Last-First, or LF walk) or lookup. The efficacy of this approach in matching short reads to a reference database of numerous short reference marker genes has remained largely unexplored [1].

Forming the concatesome.

We concatenate the reference gene sequences to form a single long string. Although this synthetic chromosome is highly repetitive, the BWT effectively enables alignment against all identical reference prefixes simultaneously, which are narrowed down as alignment progresses and differences (or errors) accrue. This enables us to leverage the advantages of the BW technique for amplicon matching and evaluate its quality and speed compared to some of the current heuristic techniques[7][18][19] that have emerged specifically to address the marker-gene matching problem in microbial metagenomics. This step also exists in varying forms and efficiencies in recent versions of some BWT-based read aligners. It is performed by NINJA in a rapid and unified way, enabling drop-in replacement of any BWT-based algorithm, by joining reference sequences them into a single long sequence and recording the location of each OTU in the new longer sequence. The concatesome is output as a new single-line FASTA file, and the original indices are output as an index table of OTU names and IDs. The concatesome serves as the reference for the BWT-enabled aligner to construct its database. For bowtie2, which we have used as the pipeline's BWT aligner for the subsequent analyses, this is accomplished with the bowtie-build command.

k-mer-based denoising.

NINJA's filter step implements a variant of 3-way radix quicksort [20] that achieves high binning and de-duplication performance on amplicon datasets using minimal RAM. This is due to its general high performance on strings combined with a median pivot scheme and increasing performance on data with larger numbers of duplicates, which is especially true for amplicon datasets. Trimming is implemented in *C* and is performed before deduplication during the parsing process by halting the read pointer and skipping to the next sequence, speeding up the parse and using less memory. Trimming at this stage can also speed up the subsequent deduplication by feeding shorter sequences to the sorting algorithm. Additionally, reads that are uniformly trimmed (preferably informed by average quality score prior to running NINJA) are more likely to have identical matches, resulting in a more condensed deduplicated output file and faster subsequent alignment. Reverse complementing is performed after any trimming at the time of output file writing by populating the write buffer in reverse with the result of a lookup performed on each sequential character in the read.

The k-mer based lossy filtering step is performed by incrementing an array of counters indexed by the integer representation of each *k*-mer formed as the read head slides across each base in the read. Formally, each *k*-mer is represented as a binary string with 2*k* bits constructed as follows:

$$x_{2i,2i+1} = \begin{cases} A: 00 \\ C: 01 \\ G: 10 \\ T: 11 \end{cases}$$

The array of counters is of length 2^{2k}, and contains the number of instances observed of a given *k*-mer in the query DNA sequences. We use $k = 8$ by default (adjustable at compile-time with a practical maximum of 14), giving a counter array of length 2^{16}, occupying up to $2^{16} \times 64$ bits or 4 MB of RAM.

The counter is incremented per k-mer per read to represent the number of that k-mer's duplicates in the dataset. The resulting count array after all sequences are parsed forms the empirical *k*-mer frequency distribution across all input reads. For amplicon data, the sorted frequency distribution resembles an exponential curve, with a small number of *k*-mers well represented and others with fewer occurrences (and many with none, depending on k-mer size). With sufficient number of input reads, the likelihood of observing an k-mer of a given rareness can be estimated from the area under the empirical frequency distribution. By checking the number of occurrences of a given k-mer against the empirical duplicate count at that probability threshold, we can determine if a given k-mer is an outlier and hence discard the read. The user can modify the stringency of outlier calling by setting a different observation frequency cutoff. We have found empirically that k-mers (with k=8 to k=14) appearing less often than 0.5%-0.1% (parameter D 0.0001) provide a reasonably safe threshold for low-strength 16S amplicon denoising, but the user is encouraged to evaluate various thresholds in context of the sequencing technology used, as well as read depth, community diversity, and experimental biases.

Singleton-based denoising.

The k-mer-based filtering provides a highly flexible and sensitive approach to denoising, but we have also implemented a very simple denoising step, which is to exclude all singleton (or doubleton, etc.) sequences from the alignment step. This is based on the premise that any sequences that appears twice or more is highly unlikely to have been the result of sequencing error, while sequences that appear exactly once are likely erroneous or so rare that they are just at the detection threshold. In several data sets we have found that this eliminates approximately 10% of the total sequences due to high likelihood that they contain errors. The resulting OTU assignments are generally unchanged when excluding singletons (Figure 6), and yet the runtime achieves an additional 3-10x speedup over the non-denoised approach. We recommend these settings (parameter “D 2”) for most users unless they have very low read counts.

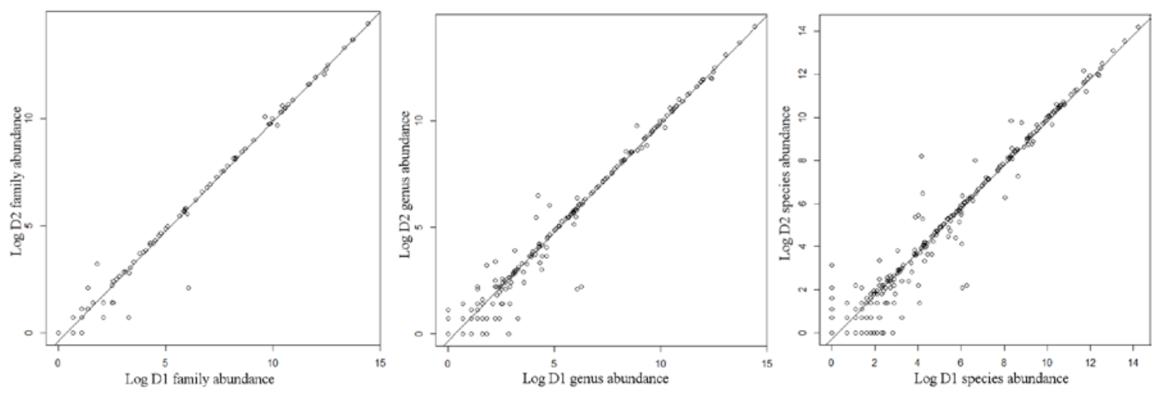


Figure 6. Effects of singleton filtering on taxonomy. These comparisons show taxa abundance correlation between singleton filtering (D2) and no singleton filtering (D1) per taxonomic level. The plots show, from left to right, the scatterplot of log abundances of all matched taxa in a dataset of 6.5 million 225-base-pair sequences at progressively higher taxonomic specificity, along with best fit lines for each. The axes correspond to log abundance within the dataset, and each dot to an arbitrary taxon abundance in the singleton denoised (Y-axis) and non-denoised (X-axis) OTU tables. The left plot shows the family-level concordance (Pearson = 0.9901727, Spearman = 0.9848349), the middle shows genus-level concordance (Pearson = 0.9845869, Spearman = 0.974338), and the right shows species-level concordance (Pearson = 0.9789319, Spearman = 0.9604182).

Alignment.

We have conducted most of our testing using bowtie2 as the BWT-enabled aligner for NINJA. To perform OTU-picking in a manner consistent with current tools, a custom set of command-line parameters was used. We sought to maintain concordance with various USEARCH operating assumptions which have become the standard for marker gene sequencing (as used in the QIIME pipeline). The primary criterion used for matching should be percent identity, which weights all matches in an alignment 1, and mismatches of any type 0. The sum of all such scores across the alignment, spanning the length of the query sequence, is divided by the total length of that alignment. Matches to ambiguous bases are also not penalized, nor are sequential gaps weighted any differently than single gaps (no affine penalty). Further, once a reasonable match has been identified in the database matching these criteria, the search terminates. The following options for bowtie2 target this behavior:

```
--np 0 --mp "1,1" --rdg "0,1" --rfg "0,1" --score-min "L,0,-0.03" -k 1 --norc
```

The percent ID is specified by setting the third argument to the minimum scoring function (--score-min) as %ID/100 - 1. In this case, to match with 97% ID, the parameter is set to 97/100 - 1 = -0.03. It is recommended that users not modify other parameters of bowtie2 pertaining to scoring criteria or output format, to maintain compatibility with the %ID match criterion and downstream NINJA parsing. It is further recommended to use the included python wrapper so that all steps from reading the formatted FASTA to OTU table creation are performed automatically for the user with a single command.

Additionally, for compatibility with NINJA and to save space, reads that fail to align are suppressed and no headers are printed to the output SAM file and the concatesome built with bowtie2-build is used as the reference database. The input sequences are the result of the filtering step. Full examples of the bowtie2 command are given in the online documentation. By default, we include presets for fast or very sensitive matching. Very sensitive matching typically improves the quality of matches noticeably at the expense of a 3-6x longer running time.

Parsing.

The final step in NINJA-OPS combines parsing the alignments, assigning OTU identity and taxonomy, and tallying OTU counts in each sample as an output OTU table (in BIOM 1.0 or **legacy** QIIME format). This is performed by NINJA using the information provided in sample dictionary generated by the filtering step, as well as the alignment data produced by the aligner and (optionally) a user-supplied table of OTU taxonomic assignments (which is also provided pre-compiled with NINJA-OPS). This parsing step is I/O bound and runs in under a second on our test computer and dataset.

Comparison.

To compare NINJA-OPS against USEARCH 8 (an earlier version of USEARCH is used in the QIIME pipeline [1]), we built the USEARCH database using the same multiple reference FASTA used to create the concatesome, the Greengenes 97% OTU representative sequence database. We performed single-strand alignment with the following fast USEARCH settings, which correspond to the stringency used for USEARCH in the QIIME pipeline:

```
-usearch_global input.fna -db db.udb -id 0.97 -blast6out hits.b6 -strand plus -maxaccepts 1 -maxrejects 32 -threads 1
```

Results

Benchmarks for this article were performed on a 2013 MacBook Air with a dual-core Core i7 CPU and 256GB SSD.

Database preparation (ninja_prep).

The runtime performance of the database generation is significantly longer than is practical to perform on-the-fly. This step only needs to be performed once for each reference database. Although ninja_prep performs the concatenation of references rapidly (it is I/O bound on the Macbook's SSD), the BWT program may spend a long time generating the BW index. For bowtie2 on our test machine, this takes over half an hour (with a maximum of one thread) on the Greengenes 97% OTU representative sequence database. For this reason, it is best to store and use pre-compiled databases for all subsequent alignments, and NINJA-OPS is distributed with a number of pre-compiled databases for commonly performed 16S bacterial marker-gene OTU matching.

Short-read filtering (ninja_filter).

NINJA filtering takes approximately 10-20% of the alignment time. For our 1.6 million read 175bp test data, without additional processing, filtering runs in just under 6 seconds and outputs a de-duplicated FASTA file approximately 1/5 the size of the original.

Short-read alignment (bowtie2).

Bowtie2 with the settings mentioned in methods aligns the entire test dataset of 1.6 million 175bp reads in 60 seconds on a single thread of the test machine. Performance for various stringency presets were measured (Figure 7), in addition to the speedup using the standard and fast presets across different datasets (Figure 8). RAM usage during alignment was 205MB in all cases, while that of USearch 8 was 720MB. Using multiple threads during alignment decreases the running time further, but speedup is sublinear, having somewhat more advantage in datasets with longer reads or higher error rates (and hence more difficult alignments). Runtime decreases to 6 seconds when using the singleton-based denoising (k-mer denoising).

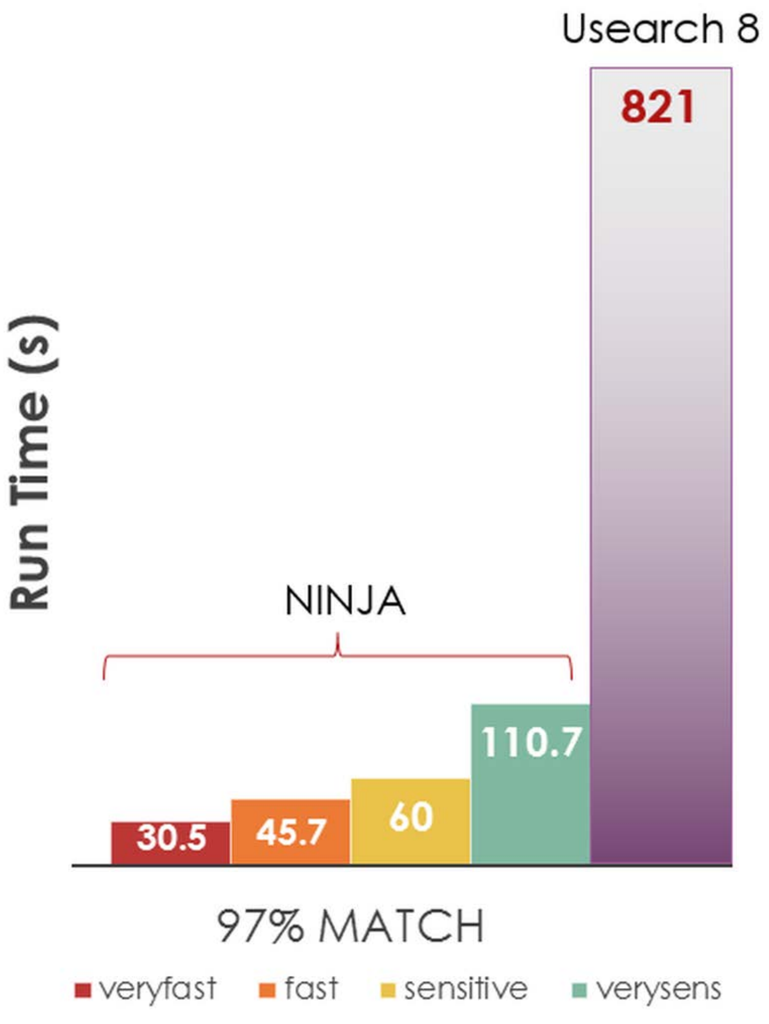


Figure 7. NINJA-OPS vs USearch 8: speed. This benchmark shows runtimes for NINJA (4 bars on the left) compared to USEARCH 8 (the bar on the right) in a single-threaded environment. Multi-threaded alignments are faster (not shown). This represents the entire time from parsing the initial FASTA file to the completion of the OTU table. The

sortMeRNA program took substantially longer than USEARCH 8 (approximately 8000 seconds; bar not shown to preserve scale).

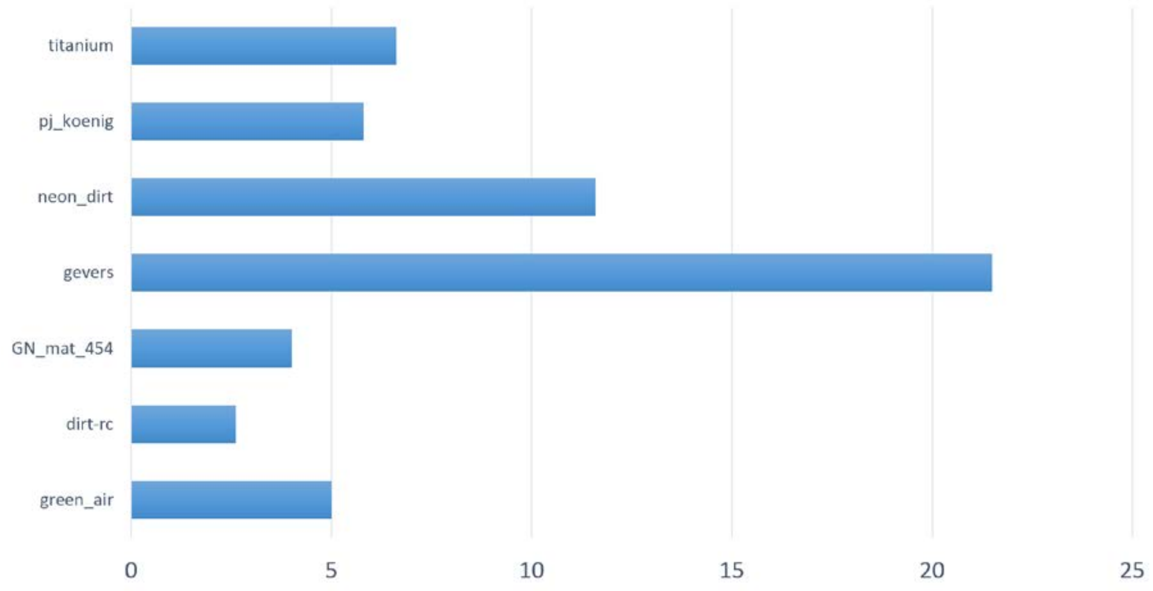


Figure 8. NINJA-OPS speedup without denoising. The effect of denoising on speedup (in multiples, or "X" speedup) over USearch 8 varies by dataset. Note that the Gevers dataset [23], for which the default NINJA preset is over 21 times faster, is fairly representative of human gut communities.

Parsing of filtered alignments (ninja_parse_filtered).

Parsing with ninja_parse takes roughly 0.5-3 seconds on datasets in the size range included here (0.5-2 million sequences). Outputting to legacy tab-delimited format instead of BIOM increases the runtime by a second or two. A Python-based convenience wrapper distributed with NINJA adds additional overhead if the user requests a FASTA file containing the sequences that failed to match the database.

Accuracy.

To assess the accuracy of the alignments found by NINJA, and to compare them to existing tools, we calculated the optimal alignment, using a semi-global version of the Smith-Waterman algorithm, of each query sequence with the reference sequence assigned by a given tool. We found that NINJA generally finds higher-accuracy matches than USEARCH 8 (Mann-Whitney U test p < 2.2e-16) (Figure 9, Figure 10). In a published dataset containing healthy subjects and patients with Crohn's disease the two methods produced the same list of differentiated genera across disease conditions with occasional disagreements about the direction of the association (Figure 11). NINJA produced a comparable percentage of matches (with standard and fast presets) to USEARCH (Figure 12), and generally comparable taxonomic assignments despite some interesting differences (Figure 13).

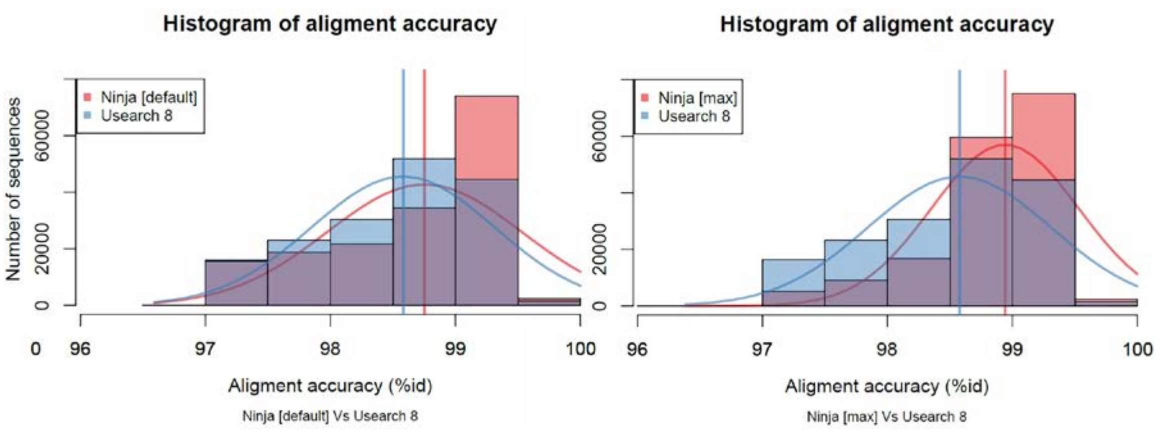


Figure 9. NINJA-OPS vs USearch 8: aggregate accuracy. The upper two red bars are darker for NINJA, showing more reads aligned at higher accuracy. The thick black bar corresponds to the average alignment accuracy, and the thin bars represent the interquartile range. A Student's T-test on the mean alignment accuracy shows NINJA's mean ID was 98.9%, while USEARCH 8 produced 98.7% with p-value < 2.2e-16. For NINJA, default settings are used, and for USEARCH 8, fast settings are used (corresponding to the defaults in the QIIME pipeline).

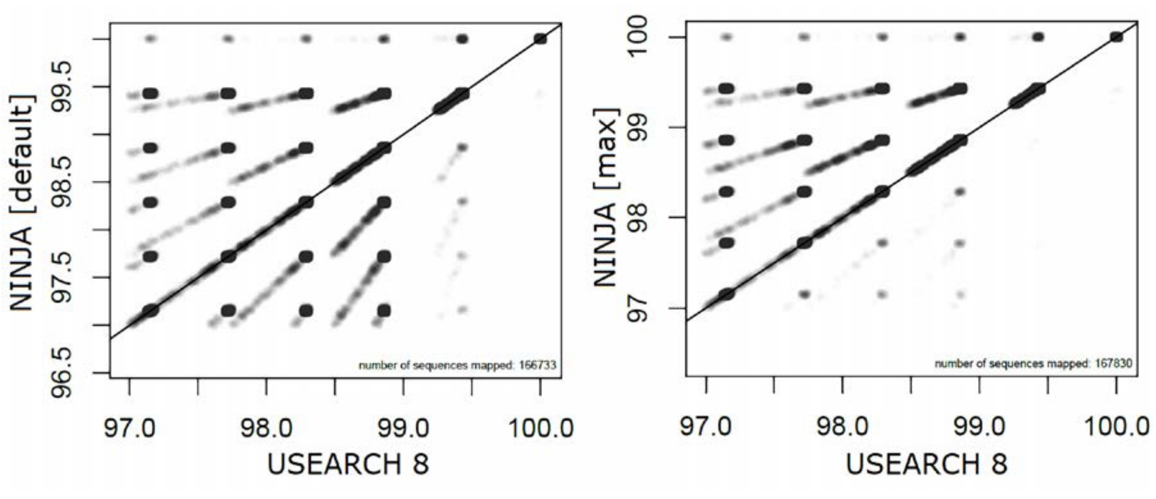


Figure 10. NINJA-OPS vs USearch 8: per-read accuracy. These scatterplots show alignment accuracy of NINJA vs USEARCH 8 for reads where both reported a match. Each point on the graph represents an sequence for which both tools found a valid alignment. A point's position along the X axis corresponds to alignment score (in %ID) for the match chosen by USEARCH 8, and its position on the Y axis corresponds to the alignment score against the match chosen by NINJA. Points along the diagonal represent sequences for which both tools picked the same quality match. Points above the diagonal correspond to sequences for which NINJA produced more accurate hits, and points below the diagonal represent sequences for which USEARCH 8 produced more accurate hits. Note the presence of a line at the top of the graph showing a number of sequences for which NINJA selected a perfect match from the database while USEARCH 8 could not.

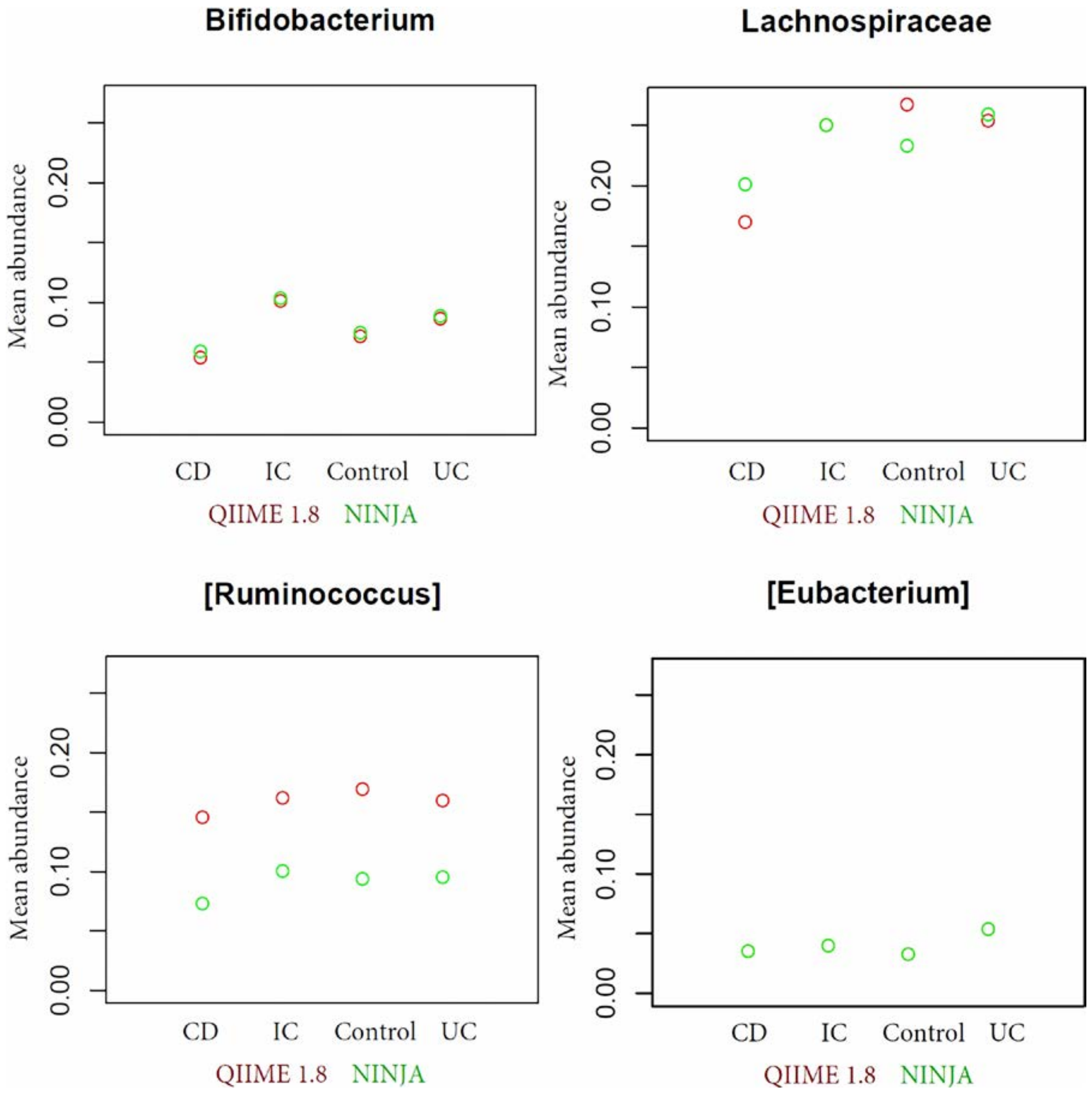


Figure 11. Significantly expressed taxa in an IBD dataset. Top left: shows concordance between NINJA and QIIME 1.8 (USEARCH/UClust). This is the most typical case. Top right: diverging trends between groups. Despite being significantly different across groups, directionality of the trend is inverted between Control and IC/UC groups for the two methods. Bottom left: preservation of general trends but difference in taxonomic abundance of [Ruminococcus]. Bottom right: NINJA reports significance difference in [Eubacterium] expression while QIIME+USEARCH/Uclust does not. Significance was determined by q-values < 0.05.

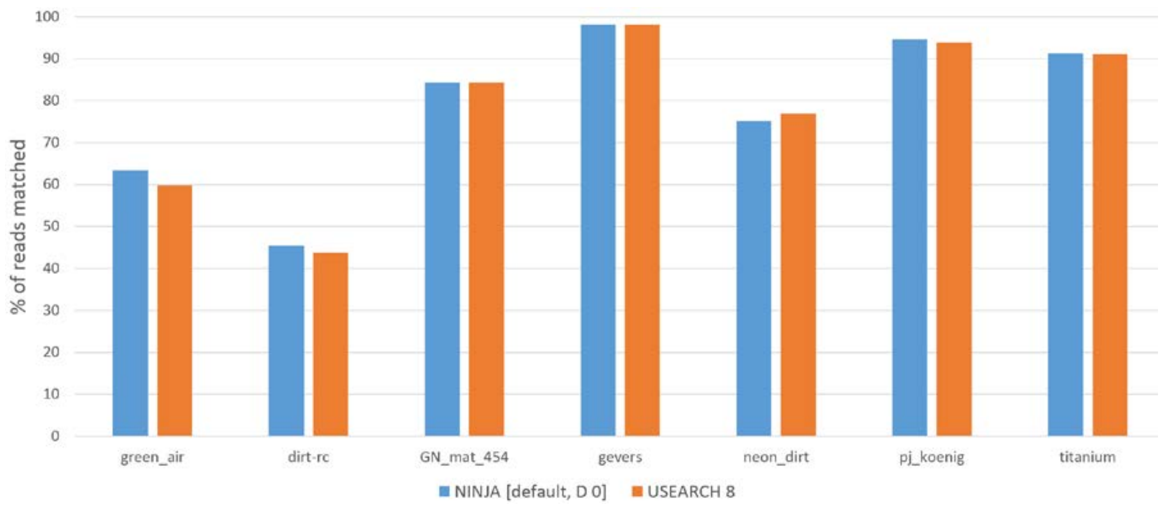


Figure 12. Proportion of successful database matching. The percentage of reads NINJA (default preset, no denoising, "D 0") successfully maps to the database are compared to USEARCH 8. There is very little difference in numbers of reads mapped to the database across datasets.

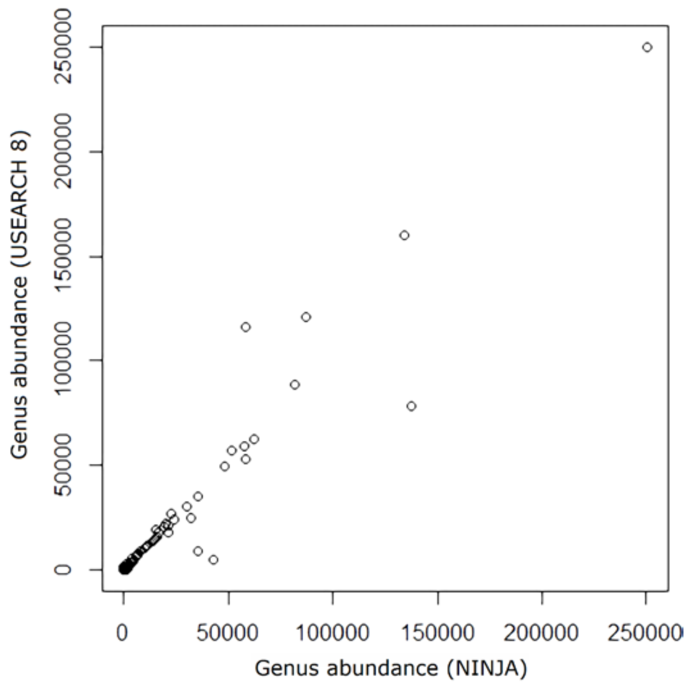
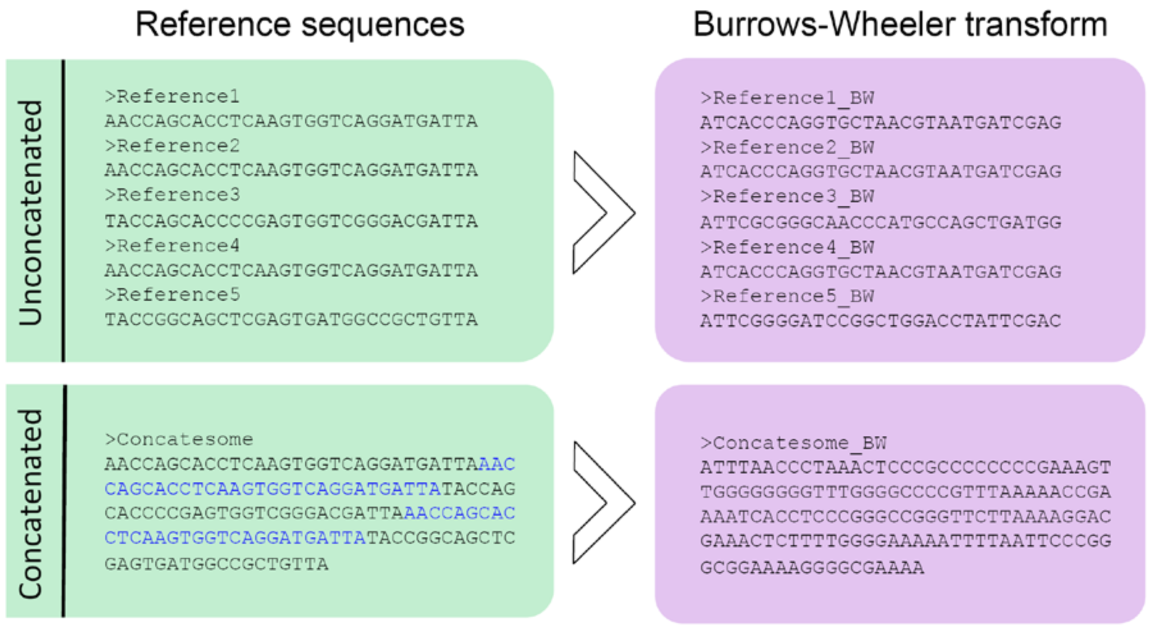


Figure 13. NINJA-OPS vs USearch 8: genus-level concordance. Each point represents a genus-level assignment, with coordinates along each axis corresponding to the total number of reads mapped to that particular genus by either Ninja (X-axis) or USEARCH 8 (Y-axis). Distance from the diagonal represents discordance of taxon calls.

Discussion.

Our tool leverages a combination of several novel approaches to accomplish an order of magnitude speedup over existing methods without compromising accuracy, and in many cases NINJA is more accurate than popular existing tools. In combination with a recommended denoising step, the pipeline achieves up to two orders of magnitude speedup over USEARCH 8. The key innovation of this tool is our use of a single long reference genome, or concatesome, composed of concatenated marker genes. This approach allows NINJA to leverage the benefits of the Burrows-Wheeler transform (Figure 14). The code is available at http://ninja-ops.ninja (or https://github.com/GabeAl/NINJA-OPS).



```
>Reference1
AACCAGCACCTCAAGTGGTCAGGATGATTA
>Reference2
AACCAGCACCTCAAGTGGTCAGGATGATTA
>Reference3
TACCAGCACCCCGAGTGGTCGGGACGATTA
>Reference4
AACCAGCACCTCAAGTGGTCAGGATGATTA
>Reference5
TACCGGCAGCTCGAGTGATGGCCGCTGTTA
```

```
>Reference1_BW
ATCACCCAGGTGCTAACGTAATGATCGAG
>Reference2_BW
ATCACCCAGGTGCTAACGTAATGATCGAG
>Reference3_BW
ATTCGCGGGCAACCCATGCCAGCTGATGG
>Reference4_BW
ATCACCCAGGTGCTAACGTAATGATCGAG
>Reference5_BW
ATTCGGGGATCCGGCTGGACCTATTCGAC
```

```
>Concatesome
AACCAGCACCTCAAGTGGTCAGGATGATTAAAC
CAGCACCTCAAGTGGTCAGGATGATTATACCAG
CACCCCGAGTGGTCGGGACGATTAAACCAGCAC
CTCAAGTGGTCAGGATGATTATACCGGCAGCTC
GAGTGATGGCCGCTGTTA
```

```
>Concatesome_BW
ATTTAACCCTAAACTCCCGCCCCCCCCGAAAGT
TGGGGGGGGTTTGGGGCCCCGTTTAAAAACCGA
AAATCACCTCCCGGGCCGGGTTCTTAAAAGGAC
GAAACTCTTTTGGGGAAAAATTTTAATTCCCGG
GCGGAAAAGGGGCGAAAA
```

Figure 14. Burrows-Wheeler transform on concatenated sequences. Example of the Burrows-Wheeler transform on both small sequences (top) and a concatenated longer sequence (bottom). The concatenated sequence is more easily searchable using the LF walk. NINJA forms a BWT-compatible concatesome that can be used interchangeably among various BWT-based aligners as an artificial reference chromosome. This concatenated sequence serves as a reference against which environmentally-obtained marker sequences are aligned.

Deduplication and NINJA-filter.

Optimizations within NINJA-OPS include tweaks to the parsing and filtering programs to increase the throughput of the processes leading up to the alignment. Deduplication is a viable strategy in marker-gene sequencing contexts because samples usually consist of fewer taxa than there are reads, and in fact are often dominated by a few highly abundant species. This results in a large number of identical reads which can be filtered out to reduce the alignment time. In human gut datasets which are quality-trimmed (or where the marker gene reads are of approximately equal length), this may result in losslessly discarding 80% of the reads as duplicates, depending on the microbial community sampled, which can speed up the downstream alignment step substantially (Figure 15). A sequence-to-sample(s) dictionary keeps track of the abundance of each sequence in each sample to ensure that each original sequence is properly accounted for wherever it was originally found. By default, NINJA-filter also performs read compaction (parameter “D 1"), which normalizes for variation in read lengths within a dataset by treating reads which are subsets of longer reads as copies of the longer reads. This increases consistency of OTU calling as well as decreasing runtime. This behavior can easily be disabled (parameter “D 0”).

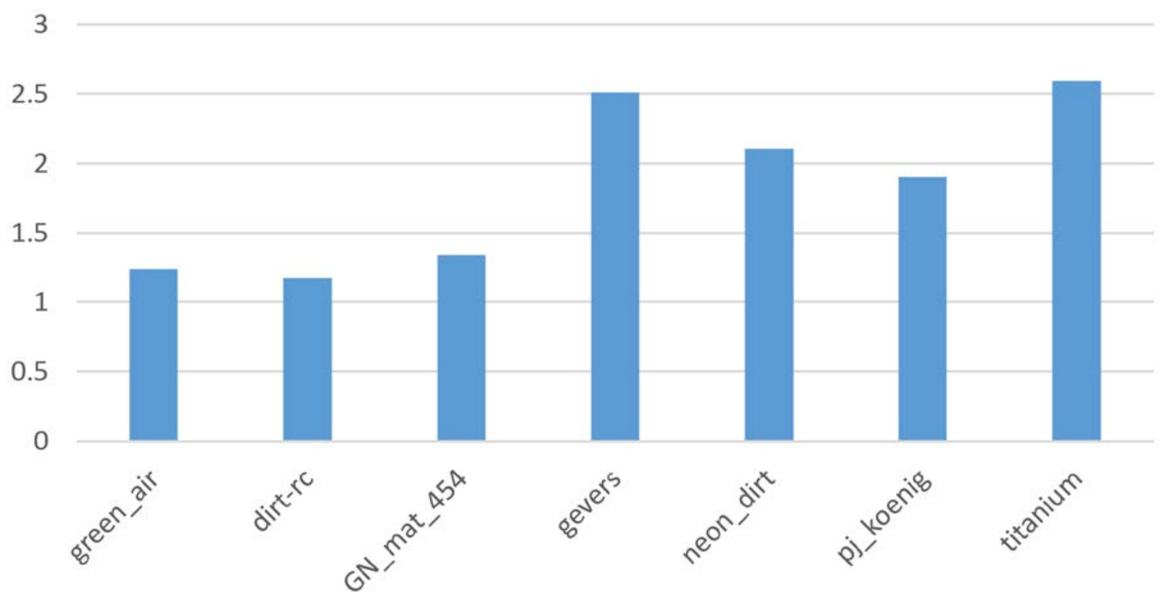


Figure 15. NINJA-OPS speedup per dataset with filtering. This bar graph shows speedup in times (or "X") of the alignment step after filtering (without denoising). Filtering can provide upwards of 2.5x improvement in alignment performance, depending on the dataset. Smaller or more unique datasets see more modest improvements. Datasets with communities of redundant sequences benefit the most. This benchmark was performed without denoising or read compaction (parameter "D 0"). Using NINJA-OPS with denoising or read compaction provides a substantially greater speedup.

An optional beneficial feature during the filter step is the ability to perform lossy denoising. NINJA performs this in two ways. The first and most straightforward for amplicon reads is to discard singleton reads (parameter "D 2"); that is, reads that have no identical match in the entire list of queries, or which are not perfectly contained in a longer read. This can be extended as the user desires from singletons to doublets and so on (parameter "D 3", "D 4", etc.). The second form of denoising is discarding reads judged to be erroneous by breaking each read into its component overlapping k-mers and comparing each of these k-mers to the counts of that k-mer in an empirical distribution of all k-mers in the body of input reads. Reads with k-mers that fail to meet user-defined criteria for support (appearing under a certain % in the dataset) are discarded completely from subsequent analysis. The resulting speedup for the downstream alignment is often much greater than the proportion of reads discarded, because Burrows-Wheeler alignment programs expend a disproportionately large amount of effort attempting to align erroneous reads that will not match the database compared to non-erroneous reads which will often find perfect (or near-perfect) matches in a well-populated database. The BW substring search is designed for perfect substring searches, so it performs most efficiently in aligning reads that have few to no mismatches with a subsequence of the database. This is also why NINJA and BWT tools perform most effectively when the alignment identity is high (%ID in the mid-to upper-90's, with taxonomic resolutions at the level of genus or finer). Performance of BWT-based tools is expected to increase as the diversity of available reference sequences increases, because the probability of finding a perfect match likewise increases.

One early concern as we were considering how to most effectively construct the concatesome was that some reads would align by chance to the boundary between two concatenated marker genes, which would produce a meaningless mapping. However, in practice, such an occurrence is exceedingly unlikely to occur in end-to-end marker gene alignments at genus-level or greater resolution due to the high identity expected over the entire length of the input read. This is even more true of marker gene alignment, where reads are much more similar to each other than in shotgun data, and the possible sites of alignment seeding are likewise similar, with significantly less randomness than would produce alignments with the boundary region by chance. The prevalence of such reads in our 16S test data is accordingly less than 1 in 1,000,000 reads aligned. Furthermore, in the unlikely event that such an alignment does occur, it is trivial to discard it in the final parsing step by testing whether the index at the end of the alignment is equal to or greater than the starting index of the subsequent marker gene. NINJA-parse automatically discards reads that map to junctions between concatenated marker genes.

An interesting finding that corroborates past findings [21] is that the commonly used bowtie, bowtie2, and BWA alignment tools do not scale linearly with increasing read length. However, due to the ability to substitute alternative BWT-based alignment programs for the alignment step, it is possible to use alternatively optimized variants such as HPG Aligner, which uses uncompressed suffix arrays instead of “traditional” BWT but shares many of the same characteristics with the added benefit of better scaling for longer reads. GPU-accelerated variants of the original algorithm are also available [22]. Additionally, NINJA-OPS is not restricted to the domain of 16S OTU picking, although it is distributed with a pre-built 16S database. Marker genes such as ITS for fungal identification [24], bacterial *rpoB* [25], and the recently proposed *Cpn*60 universal bacterial barcode [26] are easily incorporated into NINJA-OPS simply by compiling the included “ninja_prep.c” and running it on an appropriately-formatted FASTA file containing the desired marker sequences, followed by the BWT-based aligner’s database generation step. Further, NINJA-OPS can be incorporated as a preliminary step in another pipeline; for instance, NINJA-OPS can be used to group reads prior to de novo assembly [27]. This flexibility of the pipeline in allowing for substituting the aligner itself, as well as the marker gene database used, makes NINJA-OPS applicable for situations and optimizations beyond what were envisioned at the time of writing, and enable the pipeline to keep pace with emerging technologies in the sequencing and computing spheres alike.

Acknowledgments.

This work was supported by the Minnesota Partnership grant 13.28, and a University of Minnesota Supercomputing Institute travel grant.

(References are provided at the end of this thesis.)

III. BURST enables mathematically optimal short-read alignment for big data

(In submission by Gabriel Al-Ghalith and Dan Knights)

Summary.

One of the fundamental tasks in analyzing next-generation sequencing data is genome database search, in which DNA sequences are compared to known reference genomes for identification or annotation. Although algorithms exist for optimal database search with perfect sensitivity and specificity, these have largely been abandoned for next-generation sequencing (NGS) data in favor of faster heuristic algorithms that sacrifice alignment quality. This problem is compounded in metagenomics where many similar genomes (or marker genes) must be searched, leading to multiple best alignments. Here we introduce BURST, a mathematically optimal high-throughput DNA short-read aligner that enables provably optimal alignment faster than existing optimal alignment algorithms by relying on several key novel optimizations. Moreover, BURST guarantees to find all equally good matches in the database above a specified identity threshold and can either report all of them, pick the most likely among tied matches for a query given all query alignments, or interpolate taxonomic annotation with a user-specified confidence level for sequences that match multiple genomes. BURST can align, disambiguate, and assign taxonomy at a rate of 1,000,000 query sequences per minute against the RefSeq v82 representative prokaryotic genome database (5,500 microbial genomes, 19GB) at 98% identity on a 32-core computer, representing a speedup of up to 20,000-fold over current optimal alignment techniques. This may have broader implications for clinical applications, strain tracking, and other situations where fast, exact metagenomic alignment is desired.

Introduction.

As the amount of next-generation DNA sequencing data increases at a higher rate than computational power[1], approximate heuristic solutions to the fundamental DNA alignment/mapping problem are increasingly used[2]. Paradoxically, it seems, the more data made available through advances in sequencing throughput, the less accurate the alignment algorithms used to analyze it. Algorithms with maximal theoretical sensitivity and specificity under mismatch constraints exist but have largely been eclipsed, for high-throughput alignment, by techniques providing faster alignment rate at the cost of absolute alignment quality in terms of precision and/or recall.

History and background

The problem of short read mapping can be recast as one of fuzzy database search. A database is formed from numerous nucleotide sequences about which some information is known (the so-called “references”); using this database, lookups are then performed for shorter substrings about which little or no information is known (“queries”). A successful lookup assigns meaning to the query, be it match identity, relative position, functional content, taxonomy, or other information as conferred though association with the matched reference(s). It is these assigned meanings themselves that are then treated, often in aggregate, with statistical theory to arrive at some biological insight.

The role of the lookup function in this fuzzy database search problem is simply to provide meaningful mappings from queries to references. As such, the precise technical underpinnings are arguably irrelevant in cases where any such mapping, regardless of the potential existence of closer matches or the number and distribution of other valid matches, provides sufficient information to annotate queries with the desired meaning(s),. Thus, the simpler the information required of a match, the simpler the mapping function can be to provide adequate information. A trivial example is the taxonomic assignment of a single-organism sample to one of two highly divergent taxonomies; an approximate match of a query to any genomic information in one or the other suffices to reasonably infer the taxonomic origin of that query, and the sample's final assignment can be inferred by tallying the votes of all queries in the sample.

The converse also holds, in that when more complex information is desired, the mapping function must likewise increase in sophistication or accuracy. For instance, if information such as the uniqueness of a match is required, or if the best match is required among many similar matches to resolve biologically minute differences, a lookup function sufficiently sensitive to consider and discriminate among many close alignments to select the correct match would be preferable to one that would select an arbitrary "close" match.

As mentioned, the technical underpinnings of the mapping function can determine these properties. Examples of simple mapping functions are those using exact match (k-mers, substring search) and other alignment free approaches; by contrast, rigorous mapping functions are often implemented via dynamic programming[3,4], as this technique is the foundation for non-heuristic alignment, as well as other fuzzy matching techniques such as bitap[5] or edit distance kernels. Hybrid functions include popular "seed-and-extend" heuristics[6] which combine a simple initial search (seed) with dynamic programming (extension) to produce mappings of intermediate stringencies.

Previous work

Efforts to accelerate optimal alignment have largely focused on using efficient data structures[7] and low-level hardware capabilities[8]. Some efforts have also focused on pre-processing a database of reference sequences[9,10]. To our knowledge, however, there have been no previous efforts to bring optimal short-read mapping to metagenome-scale datasets with the ability to elucidate all ties and perform exhaustive inline taxonomic interpolation and/or minimum reference set computation to disambiguate tied matches deterministically.

Challenges and opportunities in short-read optimal alignment

In comparison to general-case alignment, short-read alignment, particularly of the metagenomic variety, presents distinct challenges[11]. Perhaps the most obvious among these challenges is the requirement to map millions or billions of short reads to databases containing thousands of organisms with various degrees of interrelatedness. Less obvious is the need for meaningful disambiguation when a read maps to numerous genomes or chromosomal locations, or the need to aggregate reads into taxonomic bins with a certain level of confidence (the "taxonomic binning problem" in metagenomics). The size of the databases required is large, often containing numerous redundancies (tandem repeats, multiple genomes from a single bacterial species, etc). The theoretical number of alignments required to perform an exhaustive search is staggering even at seemingly modest scales[12]; naively aligning one million short reads against a database of one million genes would require one trillion alignments, each of which necessitating a number of calculations proportional to the product of the lengths of query and reference.

However, these challenges of short-read alignment are not without attendant opportunities from a computational perspective. There are numerous restrictions embedded in the problem definition, which afford opportunities to reduce computational complexity. Since the alignments are typically DNA (or reverse-transcribed RNA), the effective alphabet size is small (usually 4 letters, but up to 16 with IUPAC ambiguity codes[13]). Since “mapping” implies the presence of close matches in the database, tools for exploring more remote evolutionary homologies (local alignment, affine gaps, and evolutionary scoring matrices) may not be entirely applicable. Indeed, allowing for some degree of mismatch, each query sequence is expected to be fully contained within some reference in order for a mapping to qualify. Further, particularly in metagenomics, a uniform selection criterion such as sequence identity (BLAST[14] percent id), or some number of mismatches, is often imposed by which to filter out potentially spurious alignments. Such a criterion also naturally gives rise to a simplified scoring framework by which to compare alignments across queries and references, and such a scoring scheme (percent identity) is used in the literature to define taxonomic homology thresholds for marker genes.

There are other advantageous restrictions from a computational standpoint. Databases are relatively immutable; that is, it is reasonable to expect that multiple query datasets will be aligned against a single reference database, allowing pre-processing or indexing of this database to facilitate future alignments. Because of the nature of short-read datasets, query sequences produced by a given sequencing technology have a well-defined maximum length. This is particularly applicable when the database contains numerous repeats, as this maximum length can define a "duplication window" with which to abstract away duplicated portions of reference sequences during database pre-processing, espousing concepts from compressive genomics and allowing database size to scale sub-linearly with the number of reference sequences therein. In addition to such database redundancy, query sequences of a limited alphabet, when sorted, contain trivial prefix redundancy which increases with the number of queries considered at a time, even for randomly generated queries.

Further advantageous restrictions can be found in the alignments themselves, as researchers often provide a "minimum acceptable" alignment score in the form of a minimum percent identity or maximum number of mismatches. Further, alignments other than the best alignment(s) may be suppressed, depending on use-case. Certain combinations of these restrictions naturally imply others; for instance, when no minimum acceptable score is provided but only the best alignments are desired in the alignment report, the program can internally infer such a score for each query as more alignments are seen. This is useful if the program is capable of disqualifying alignments known to be worse than the current best without needing to calculate the entire alignment. Alternatively, if the user specifies a threshold and requires all alignments be reported above it regardless of which are best, the program can still disqualify some alignments based on the provided minimum score.

For each of the aforementioned restrictions (and others discussed in later sections), an implementation is possible that can guarantee that the resulting alignment report (after any user-defined filters are applied) will be exactly identical to that produced by a full exhaustive alignment of all queries against all full-length references. This holds even in the presence of ambiguous bases (which, except for bases 'X' and 'N', default to allowing unpenalized matches).

The reporting of alignments themselves, however, is often of limited use at the metagenomic scale, as information about the likely taxonomic identity (or some interpolation of other hierarchical metadata) may be desired instead. Alternatively, some useful disambiguation of multiply matched queries is often warranted by taking into account relationships among the queries themselves post-alignment. A simple use-case is picking a single reference for each query that will minimize the total number of references chosen by all queries, in order to produce the most likely minimal set of references that sufficiently matches all queries. The latter may be of particular use for determining which individuals among closely related reference sequences are represented in a complex community.

Results

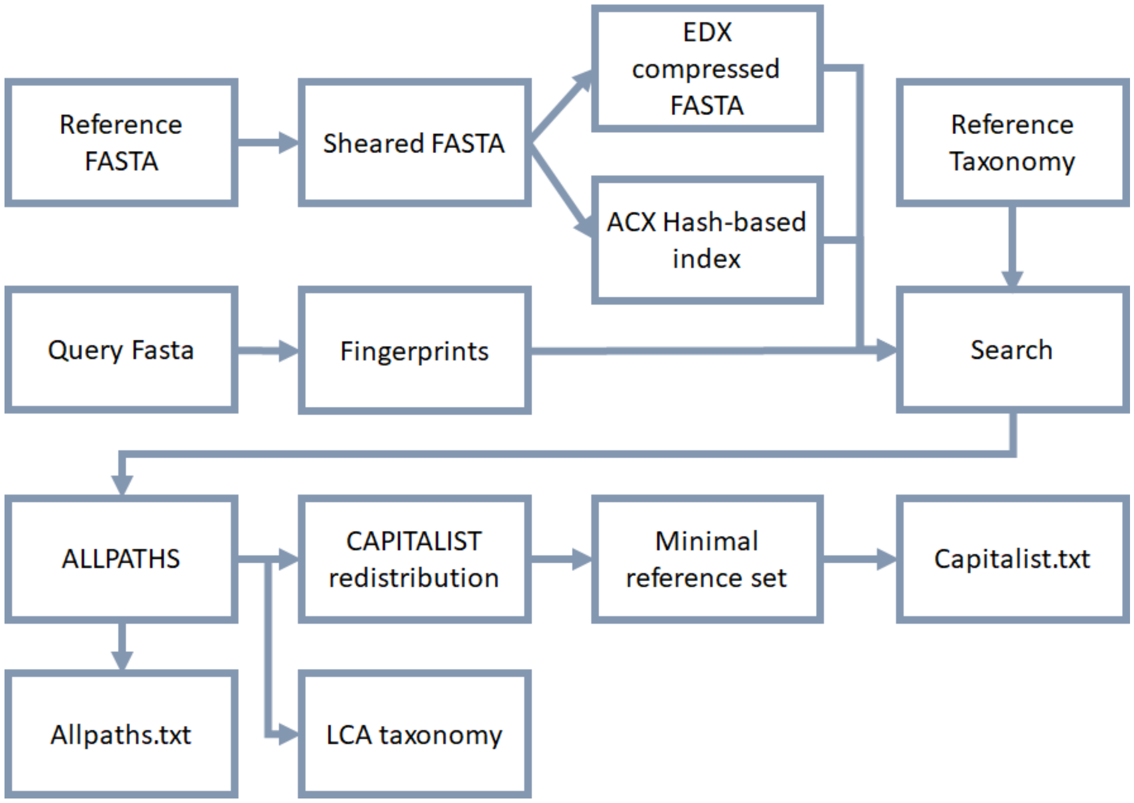


Figure 16. Schematic of BURST. This schematic depicts both the database-generation and alignment arms of BURST. In a common workflow, the user begins by creating a database out of a FASTA of reference sequences, then using that database to align a FASTA of queries and produce alignments with taxonomy according to the desired reporting mode.

Here we present BURST, a metagenomic-scale alignment software which performs exhaustive, optimal alignment to map short reads to a (meta-)genomic database, guaranteeing the recovery of any or all matches of query sequences to references in the database above a user-specified alignment similarity score (percent identity). A schematic is presented in Figure 16. Various output modes exist (with similar runtime performance) to output all tied best matches, select a single best match among ties that minimizes the number of unique references chosen, and assign interpolated taxonomy (or other semicolon-delimited hierarchical metadata) that satisfies a user-specified confidence threshold.

To accomplish this with feasible runtime performance, a series of optimizations have been performed as shown in Table 2. It is important to note that only optimizations that do not reduce sensitivity of specificity of alignment are included; they increase speed at no cost of accuracy.

Table 2. Overview of algorithmic optimizations in BURST.BURST uses multiple optimizations to achieve speedup without affecting alignment quality. "+" indicates further speedup is possible in tandem with other items.

Optimization	**Speedup**
Error-aware pruning	>2x[1]
Lexicographic table caching	>2x[1]
Eliminate matrix tracebacks	2x
Vectorization along reference clusters, SSE4.1 instructions	10x
2-pass search: edit distance, BLAST identity	2x
Code micro-optimizations	2x
Multithreading: thread-local data with global sync	+0.9x/thread
Optional high-identity map filter ("accelerator")	2-20x[2]
Optional database support with low-identity fingerprint filter	1-5x[3]
Lossless Heuristics: Ordering, pre-pass, best-hit prioritizing	1-10x[3]

[1] Up to 100x when both caching and pruning are used together
[2] Lower estimate for amplicon databases; higher for shotgun databases
[3] Depends on read length, shearing factor, database uniqueness, and desired identity cutoff

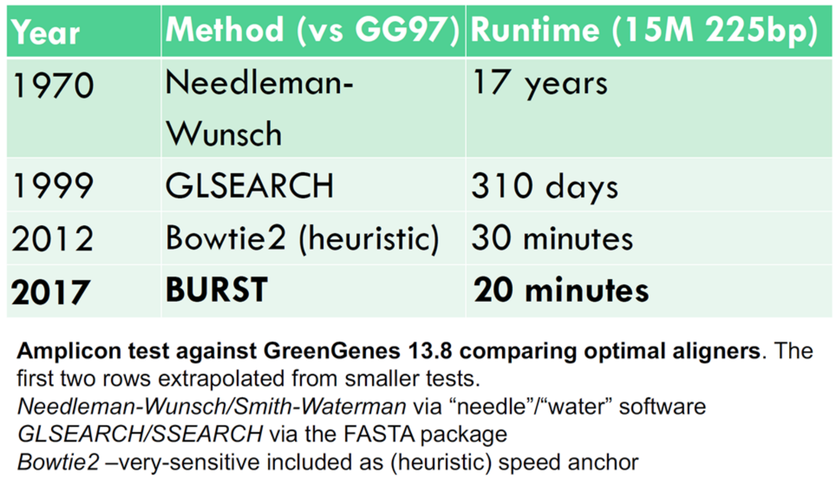
Performance and system requirements

Generally, the memory and disk requirements of BURST are low with only a primary database, but high when an additional accelerator is produced. The primary database alone occupies space and memory of 0.5-1x the size of the original FASTA file of references. The accelerator, if produced, further occupies 2-7x the size of the original FASTA file of references in RAM and disk space.

BURST transparently distinguishes between two types of accelerator formats, "small" and "large." Large format accelerators are produced when the number of reference sequences after shearing and deduplication exceeds 1 million. The RAM and space requirements of both the database and accelerator tends to be just over 7x the size of the original FASTA file of references used to produce them, assuming little redundancy in the original sequences. An example is the "reference and representative genome collection" from NCBI RefSeq[15] v82, which contains just over 5000 representative prokaryotic genomes and very little redundancy. With increasing redundancy, this factor can drop considerably, such as to 2.5x with over 51,000 prokaryotic genomes, such as those drawn from NCBI RefSeq v84 (all complete, chromosome-, and scaffold-level genomes plus all "representative" contig-level genomes). This factor is expected to drop further with the addition of more genomes as more redundancy is found.

Importantly, BURST demonstrates dramatic speedup over existing optimal algorithms in terms of alignment speed on a real gut amplicon dataset (Table 3). In practical metagenomic alignment, BURST can align, disambiguate, and assign taxonomy at a rate of 1,000,000 150-base query sequences per minute against the RefSeq[15] v82 representative prokaryotic genome database (5,500 microbial genomes, 19GB) at 98% identity on a 32-core computer.

Table 3. Alignment through the ages. BURST, an exact method, shows runtime performance comparable to heuristic algorithms on human gut amplicon data using the Greengenes 97% OTU database.



Year	Method (vs GG97)	Runtime (15M 225bp)
1970	Needleman-Wunsch	17 years
1999	GLSEARCH	310 days
2012	Bowtie2 (heuristic)	30 minutes
2017	**BURST**	**20 minutes**

Amplicon test against GreenGenes 13.8 comparing optimal aligners. The first two rows extrapolated from smaller tests.
Needleman-Wunsch/Smith-Waterman via “needle”/“water” software
GLSEARCH/SSEARCH via the FASTA package
Bowtie2 –very-sensitive included as (heuristic) speed anchor

Availability

BURST is available under the AGPLv3 license. Precompiled static binaries (with no dependencies or installation required other than simply downloading and running BURST). It is available on GitHub at github.com/knights-lab/BURST (see the releases page under "Downloads" for the precompiled binaries: github.com/knights-lab/BURST/releases).

Methods

Alignment

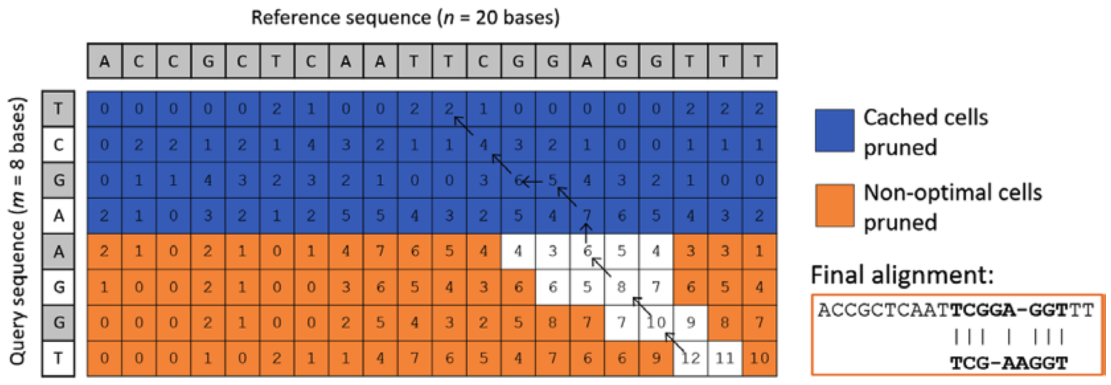
The alignment scoring function proposed and used in this work is fundamentally a dynamic-programming based similarity measure with a two-component objective function, the first of which is Levenshtein distance[16] and the second being alignment identity (sometimes called "BLAST id"). Creation of this hybrid metric was informed by a desire to score internal alignments using criteria consistent with those commonly used later to filter them. The advantage in using the hybrid measure over either component alone is that the Levenshtein distance alone is subject to producing multiple different but identically-scoring alignments from a single short-read alignment, and the BLAST id may produce a slightly higher-scoring alignment that is substantially longer and more gap-ridden, and hence of dubious biological merit. Using BLAST id to pick from among those alignments produced by Levenshtein distance provides a natural resolution. Another commonly used metric for screening and evaluating alignments, the e-value, was not considered as BURST performs end-to-end alignments only, and in context of short-read mapping, the query sequences are often of similar or identical lengths as one another, reducing the discriminative utility of the e-value in this context.

Although the omission of alternative scoring criteria such as variable gap penalties or nucleotide-specific scores may be unreasonable for general-purpose alignments[17,18], their utility in the high-identity short-read mapping problem is unclear. The objective of short-read mapping, rather than characterizing local evolutionary homology as with comparison of two full-length protein sequences, is instead tantamount to fuzzy substring search where the relatively few allowable mismatches can be contributed by artifactual sequencing noise. Hence, the unbiased measurement of absolute sequence similarity was deemed appropriate in this context. Further, such an approach may have contributed to the high relative sensitivity and specificity of the bowtie2-based NINJA-OPS[19] software in short-read amplicon mapping despite bowtie2's reliance on randomization and seed-and-extend heuristics[10].

"Synergistic" caching and pruning

Within the alignment matrix, two major optimizations are used which in tandem increase the rate of alignment more than either alone. The first such optimization is lexicographic caching of previous query alignments to the same reference, eliminating the need to recalculate this matrix for a number of rows equal to the length of the shared prefix. Importantly, for such an optimization to be useful, query sequences are sorted using a fast multi-threaded sorting method to maximize the length of the shared prefix between adjacent query sequences. For 1 million random sequences with 4-letter alphabet (DNA), it is easy to observe that the length of a shared prefix of length 10 will be observed, on average, 61.5% of the time between any two adjacent queries: 1-(1-(1/4)^10)^1000000. For amplicon data, the shared prefix length is typically much higher as the selection of genomic region in this case is non-random. Figure 17 (blue cells) shows an example of this optimization (we can assume the previous query sequence is "TCGAACAA").

Another optimization used in tandem with this caching step is opportunistic pruning of guaranteed "dead end" paths through the dynamic programming matrix (Figure 17, orange cells). For each cell, a threshold score calculated by summing that cell's current alignment score with the maximum possible score for all remaining nucleotides in the query, assuming all receive perfect matches from this query position onward. The cell is marked inactive if this threshold score is lower than the minimum user-defined score (or the best score among all previous alignments of this query, if in a "best-hit" reporting mode). Elimination of traceback is also implemented so as not to require final traversal through the best-scoring path at the end of alignment; this is accomplished by storing an auxiliary matrix that tracks gap counts directly rather than noting the direction of the best-scoring path at each cell (Figure 17, black arrows).


Reference sequence (n = 20 bases)
Query sequence (m = 8 bases)
Cached cells pruned
Non-optimal cells pruned
Final alignment:

```
ACCGCTCAATTCGGA-GGTTT
          ||| | |||
          TCG-AAGGT
```

Figure 17. Simplified example DNA alignment accelerated by BURST. Optimal alignments usually require filling a dynamic programming matrix from top-left to bottom-right and finding the highest-scoring path. BURST skips

calculating all blue and orange cells while guaranteeing the best alignment (saving 146/160 = 91.25% of the computation). The previous query's alignment (not shown) is cached until the point of prefix divergence from the current (blue). Also, cells that cannot form future valid alignments given current scores (orange) are not pursued. Path traceback (arrows) is also eliminated.

Another strategy for improving runtime performance is the use of a 2-pass scoring criterion for determination of best matches. Because the Levenshtein distance calculation is prerequisite for the secondary BLAST id selection, the former alone can be applied as an initial screening pass; if an alignment fails the first criterion, it will by definition fail the second and be rejected, sparing the need to compute both for each alignment.

BURST enables arbitrarily confident taxonomy assignment per read with thresholded LCA.

In one of its modes of operation, BURST can assign taxonomy (or any other hierarchical, semicolon-delimited feature) to each read using lowest common ancestor (LCA).

Discussion

BURST is a high-performance standalone program with no dependencies that enables exhaustive, high-throughput metagenomic alignment without heuristics or approximations. This makes BURST uniquely suitable for a variety of tasks where accuracy is of paramount importance, such as part of a pathogen screening or strain detection pipeline. As shown in Figure 18a, the heuristic aligner bowtie2[10] increasingly fails to discover optimal matches with higher sequence divergence from known references, whereas BURST's best hit always returns an optimal match. Not only does BURST provide more accurate "best hits" with decreasing sequence similarity, it also enables some novel use-cases as well.

BURST can be used to report conservative taxonomy on a per-read basis by finding the lowest common ancestor among all tied alignments against all references for each query. Using this approach on an example human amplicon dataset[20] with the Greengenes[21] 13_8 reference database, we find that using a naïve “best-hit” approach for assigning taxonomy results in drastic levels of overconfidence in taxonomy assignments (Figure 18b). This overconfidence can lead to split signals when, for instance, multiple different species of a genus are reported when they do not actually exist in a sample. In this case, BURST would report higher amounts of the genus alone without assigning arbitrary species labels at random.

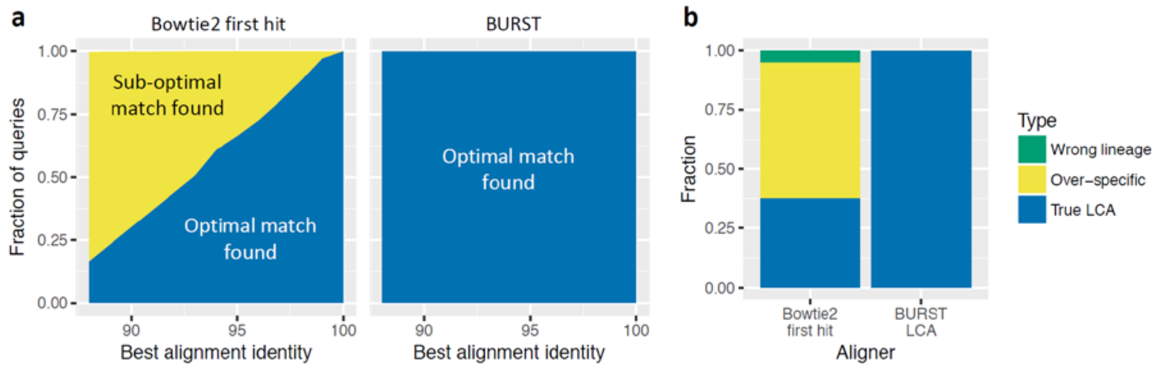


Figure 18. BURST vs Bowtie2: best hit and LCA taxonomy (a) A comparison of BURST in best-hit mode compared to the default runtime configuration of bowtie2 on an example gut amplicon dataset against the Greengenes 97% OTU database. BURST recovers an optimal best-scoring match for all queries at all alignment similarities, while bowtie2 returns more suboptimal matches with decreasing alignment identity. (b) A comparison of BURST in LCA mode vs bowtie2 default settings in terms of reported taxonomy. When multiple identically-scoring best hits exist, BURST will return the lowest common ancestor (LCA) as the taxonomy, while best-hit methods pick an arbitrary representative which is often over-specific or, due to heuristics, sometimes the wrong lineage entirely.

When used in “Forage” mode, BURST can also be used as an exhaustive means to determine where a set of primers may align in a database of many genomes, including the ability to specify primers with ambiguous bases and any number of mismatches, insertions, or deletions. Additionally, using BURST’s ability to report all tied best alignments (“allpaths” mode) can be used in interesting ways such as in Figure 19, which shows a Cytoscape visualization of the relatedness of 21 strains within three species of the bacterium *Enterococcus*. For *E. faecalis* and *E. faecium*, 10 strains each were randomly selected from RefSeq [15] v86 assemblies, and a single strain of *E. aquimarinus* was used. As expected from a phylogenetic analysis[22], *E. faecalis* and *E. faecium* are substantially more closely related than either is to *E. aquimarinus*. Ecologically, the former two microbes are also common gut commensals, whereas *E. aquimarinus* was isolated from sea water[23]. It may also be inferred that the randomly selected E. faecalis strains are more dissimilar to one another than the E. faecium strains. This may have important implications for creation of a reference database, as intelligent strain selection may be beneficial to capture a species' full pan-genomic content without inflating the size of the database with highly similar strains. More fundamentally, it visually highlights the fact that not all species of a genus are equidistant from one another, nor are strains of a species.

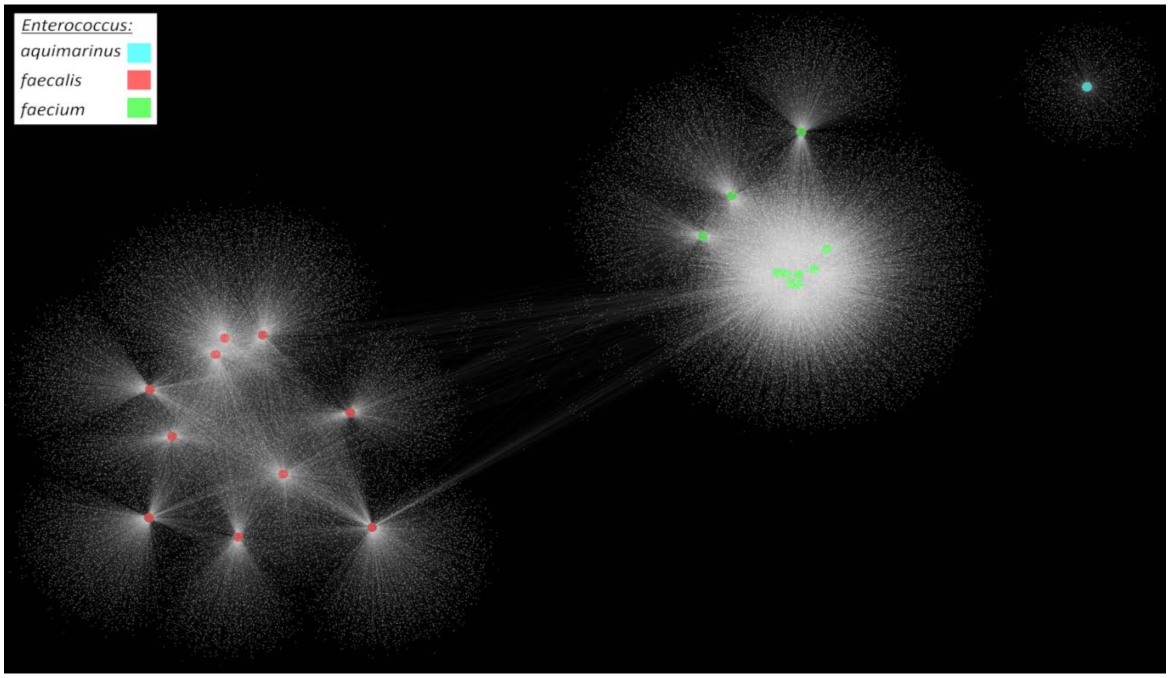


Figure 19. BURST reveals species relatedness through shared genes. All genes from 21 strains of Enterococcus were aligned with BURST in "allpaths" mode against a database of the same strains' genomes. Large nodes indicate individual genomes, colored by species, and white dots represent individual genes. Lines indicate alignments between genes and genomes. Force-directed layout was used to automatically orient the network. The randomly selected E.

faecalis and E. faecium share more genes with themselves than with each other, and the E. aquimarinus shares little with either other species.

Another reporting mode in BURST that may be especially useful for whole-genome metagenomics is the “capitalist” reporting mode, a Minpath-like[24] parsimony-based disambiguation scheme the workings of which are highlighted in Figure 20a in comparison to the conservative LCA method (Figure 20b) also demonstrated above. An example of running this mode on a simulated dataset is shown in Figure 21.

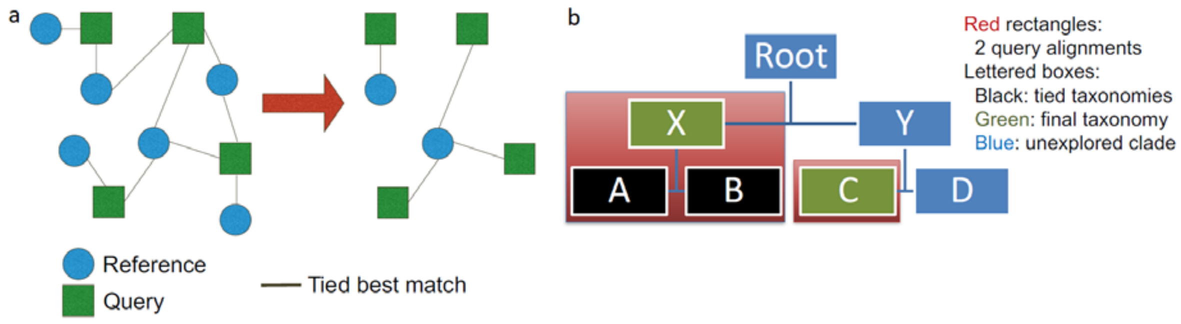


Figure 20. Illustration of BURST "Capitalist" disambiguation vs LCA. (a) Tied best matches arise due to repetition in multi-organism reference databases. To resolve this, BURST computes the minimal set of references necessary to satisfy all queries. This reduces split statistical signals present in most metagenomic data. (b) Lowest common ancestor (LCA) is an alternative, more conservative approach that independently assigns each read a taxonomy by backtracking from leaf to root (most specific to least specific). BURST implements both methods.

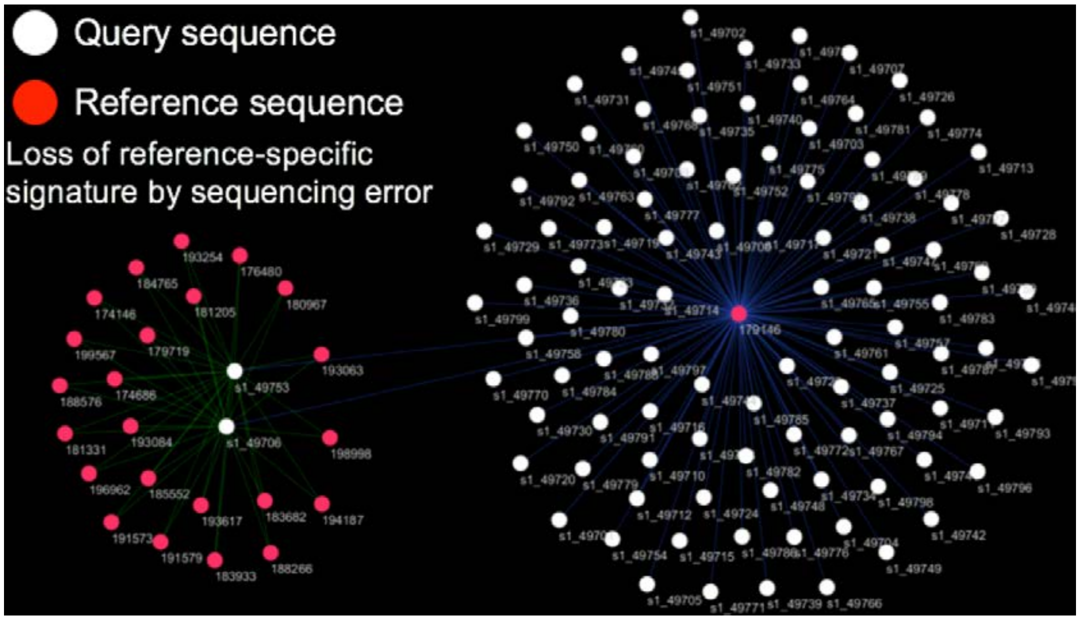


Figure 21. Practical example of CAPITALIST. In this example run on a simulated dataset, all of the queries (white dots) originate from the same reference (red dot in right-center). However, in two of these queries (two white dots on the left-center), the nucleotides necessary to confer unambiguous best alignments to their reference of origin were mutated, making them align equally well to a number of other references as well. Lines represent all possible tied best alignments; blue lines represent capitalist picks (minimizing alignments), which would eliminate all extraneous references on the left and properly assign the two left-most queries to the same reference as the other queries.

Conclusion.

With its favorable performance attributes and guarantee of alignment optimality, BURST may be a reasonable alternative to other aligners in the metagenome space. Further, BURST incorporates a number of features useful for metagenomic analysis including methods to disambiguate ties, assign conservative taxonomy annotations, and reduce database size with increasing numbers of similar reference genomes. Because it is not splice-aware, nor does it support human-centric alignment formats such as SAM [25], it may not be immediately suitable for human genomic applications reliant on these features.

Further improvements to BURST may include the ability to align protein sequences (including local alignments within a specified minimum length and support for distance matrices) as well as further improvements to the compressive database engine and other optimizations to increase speed and reduce memory usage. Improvements to the reporting of alignments may also include incorporation of a “coverage”-based statistic, the goal of which would be to disambiguate reads based not only on aggregate parsimony but also on evenness of read coverage throughout the genome. Future work may characterize the extent to which the various output modes of BURST impact the performance of statistical and machine learning models.

(References are provided at the end of this thesis.)

IV. Other contributions to comparative genomics and database generation/search.

(Not submitted for publication as of this thesis.)

Summary.

DNA sequence alignment is a foundational centerpiece of modern genome-enabled disciplines. Despite its ubiquity, however, sequence alignment is employed in vastly different ways depending on the biological objective desired. For strain detection and clinical applications, sensitivity and specificity are paramount; for broad ecological analyses of uncharacterized communities, a rough taxonomic synopsis of an environment is often sufficient. Accordingly, where databases are used (i.e., for alignment, taxonomy profiling, or phylogenetics), their use-cases likewise differ by purpose. The various extant “biological objective functions” have driven the development of numerous bioinformatics tools for alignment, profiling, and both database-dependent and database-independent comparison methods for sequences and communities of sequences. I have developed and implemented a variety of tools to tackle problems across metagenomic sequence analysis, including specialized databases, algorithms, utilities, and pipelines for various applications. Specifically treated here are: marker gene and whole-genome metagenomics sequence databases and phylogenies, a lossless data compression engine, a fast heuristic aligner, automated database update protocol, an alignment-free phylogenetic tree algorithm, a database-free metagenomic sample comparison algorithm, a fast *de novo* amplicon clustering pipeline, a non-parametric statistical comparison technique for groups with differing baselines and sample numbers, a toolset for metagenomic alignment summarization including coverage analysis, and a microbial coalition-based feature generator. Each of these tools improves upon existing standards in various ways, and some provide entirely new capabilities for generating biological insights.

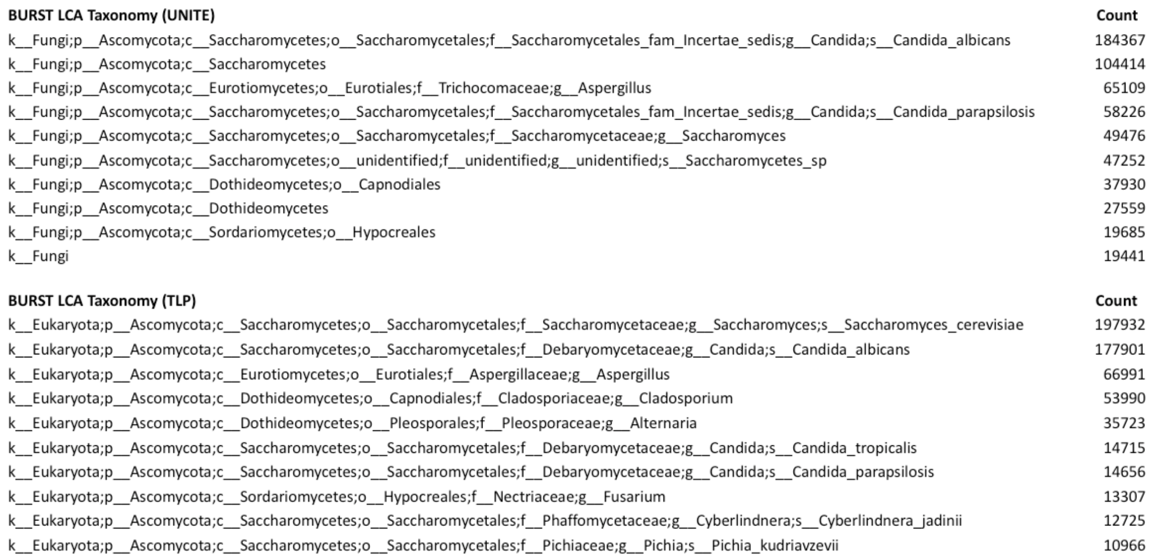
Microbial genomic, taxonomic, and phylogenetic marker databases.

I have produced a database for 16S microbial marker gene analysis based on the NCBI RefSeq Targeted Loci Project (TLP) [1], as well as an ITS microbial marker gene database based on a related curation project for fungi [2]. A primary motivation for this project is a consistent curation system as well as gold-standard (TYPE-strain) taxonomic information. The 16S records each contain a full-length or near-full-length 16S region selected by a small number of universal primers. Each ITS record contains full-length, contiguous ITS1, 5.8S, and ITS2 regions.

Phylogenetic trees were created from these full-length records using rigorous multiple-sequence alignment (MSA) via structural global sequence alignment with the rigorous and computationally-intensive MAFFT v.7.310 Q-INS-i algorithm [3] on Minnesota Supercomputing Institute servers, followed by maximum-likelihood tree creation with RAXML-ng v0.5.0 [4].

These databases are particularly useful with an exhaustive optimal aligner such as BURST [Chapter 3] because taxonomy assignment can occur through conservative lowest-common-ancestor approach at lower alignment identity, which is not feasible with other applications which require heuristically pre-clustered databases such as Greengenes [5]. Particularly for fungal analysis, the UNITE [6] database is not as useful with BURST LCA due to many records lacking meaningful taxonomic annotation, leading to many annotations being overly conservative.

Table 4. BURST LCA with UNITE vs LTP. The top 10 taxonomic annotations resultant from BURST LCA are shown for both the UNITE 7 database and my new TLP-based database.



BURST LCA Taxonomy (UNITE)	**Count**
k__Fungi;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Saccharomycetales_fam_Incertae_sedis;g__Candida;s__Candida_albicans	184367
k__Fungi;p__Ascomycota;c__Saccharomycetes	104414
k__Fungi;p__Ascomycota;c__Eurotiomycetes;o__Eurotiales;f__Trichocomaceae;g__Aspergillus	65109
k__Fungi;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Saccharomycetales_fam_Incertae_sedis;g__Candida;s__Candida_parapsilosis	58226
k__Fungi;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Saccharomycetaceae;g__Saccharomyces	49476
k__Fungi;p__Ascomycota;c__Saccharomycetes;o__unidentified;f__unidentified;g__unidentified;s__Saccharomycetes_sp	47252
k__Fungi;p__Ascomycota;c__Dothideomycetes;o__Capnodiales	37930
k__Fungi;p__Ascomycota;c__Dothideomycetes	27559
k__Fungi;p__Ascomycota;c__Sordariomycetes;o__Hypocreales	19685
k__Fungi	19441

BURST LCA Taxonomy (TLP)	**Count**
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Saccharomycetaceae;g__Saccharomyces;s__Saccharomyces_cerevisiae	197932
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Debaryomycetaceae;g__Candida;s__Candida_albicans	177901
k__Eukaryota;p__Ascomycota;c__Eurotiomycetes;o__Eurotiales;f__Aspergillaceae;g__Aspergillus	66991
k__Eukaryota;p__Ascomycota;c__Dothideomycetes;o__Capnodiales;f__Cladosporiaceae;g__Cladosporium	53990
k__Eukaryota;p__Ascomycota;c__Dothideomycetes;o__Pleosporales;f__Pleosporaceae;g__Alternaria	35723
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Debaryomycetaceae;g__Candida;s__Candida_tropicalis	14715
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Debaryomycetaceae;g__Candida;s__Candida_parapsilosis	14656
k__Eukaryota;p__Ascomycota;c__Sordariomycetes;o__Hypocreales;f__Nectriaceae;g__Fusarium	13307
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Phaffomycetaceae;g__Cyberlindnera;s__Cyberlindnera_jadinii	12725
k__Eukaryota;p__Ascomycota;c__Saccharomycetes;o__Saccharomycetales;f__Pichiaceae;g__Pichia;s__Pichia_kudriavzevii	10966

A meaningful fungal ITS marker tree created using modern rigorous methods has not been previously attempted. The closest analog, Ghost-tree [7], uses a two-step approach which requires additional matching records using another ribosomal subunits which may not be available for all TYPE strains. Additionally, the alignment and tree generation methods used in that publication are dated, and the databases used do not include all up-to-date TYPE strain information from the TLP. In contrast, my tree can be regenerated directly from the full TLP ITS data and has tips for all fungi included therein (Figure 22). Furthermore, I have found this ITS tree performs comparably well (or slightly better) in separating fungi by environmental body sites than Ghost-tree in a real human mycobiome dataset [8] as shown in Figure 23.

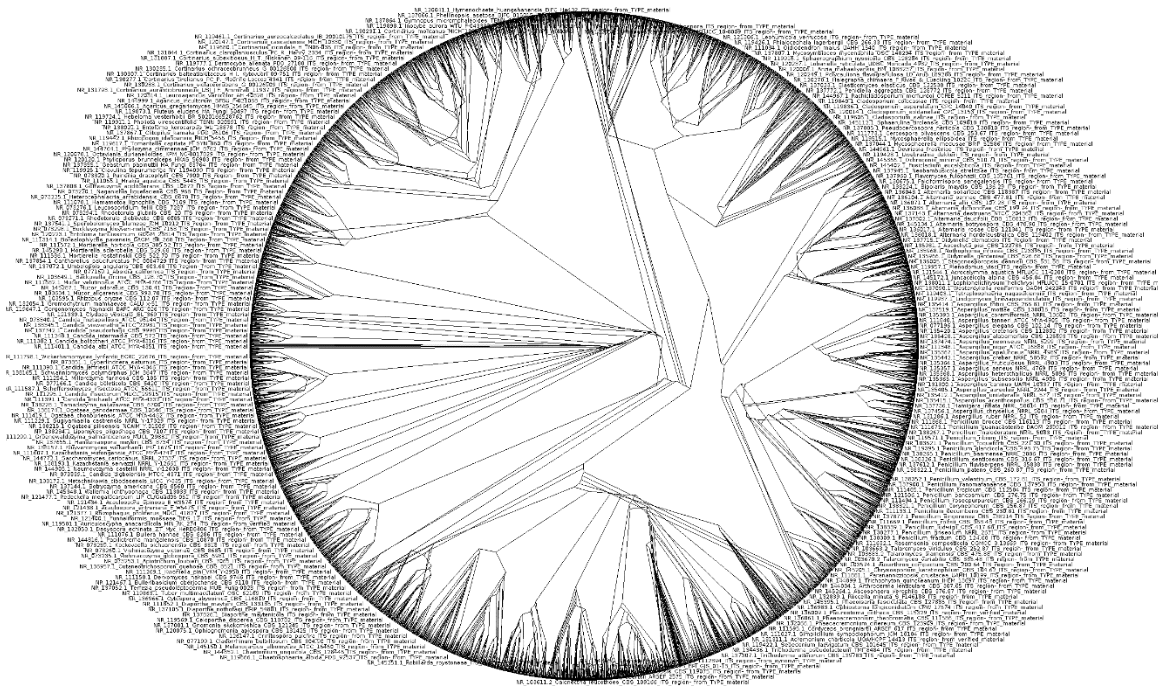


Figure 22. Phylogenetic ITS marker tree.

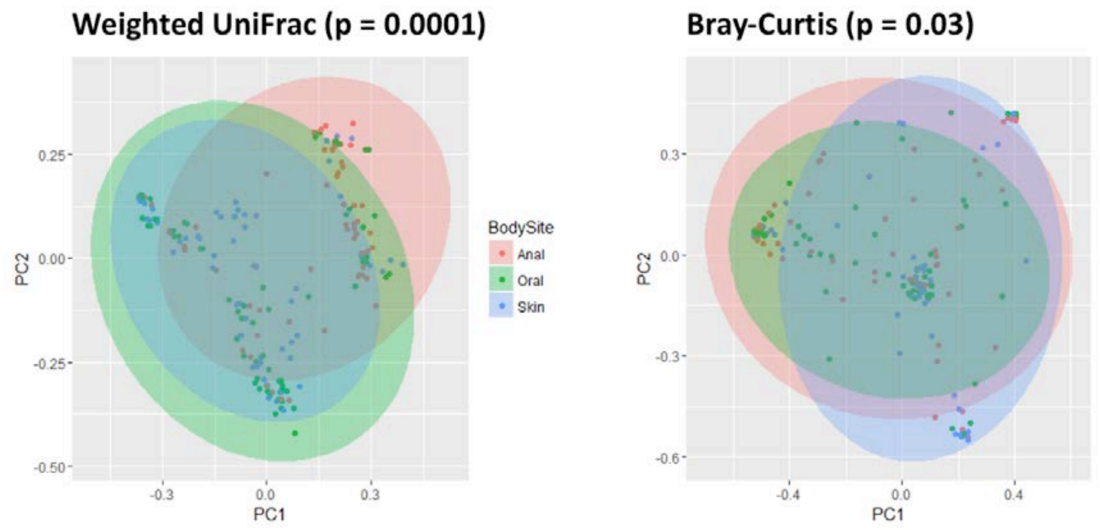


Figure 23. ITS tree distinguishes communities by body site. (left) ITS marker tree-based UniFrac dstance. (right) Naïve taxonomic analyis without phylogeny using Bray-Curtis disance metric.

Prokaryotic metagenomic full-genome databases.

I have also produced a number of full-genome databases based on NCBI RefSeq microbial assemblies. Using the “representative” and “reference” genome labels provided by RefSeq for its microbial databases, I have produced a streamlined whole-genome and gene-separated database containing approximately 6,000 distinct representative microbes spanning the tree of life from RefSeq v87. I have also created a more comprehensive database of over 50,000 prokaryotes from all microbes except those with genomes labeled to be of the lowest level of assembly by NCBI, the “contig” classification. For genomes of “contig” level completeness, I only included genomes marked “representative” or “reference” as contained in the streamlined database, making the comprehensive database a perfect superset of the representative database.

I have also produced a specialized human and murine-associated microbial reference database containing over 10,000 microbes selected as follows. First, a list of candidate microbial species was obtained by aligning human and murine reads from in-house as well as publicly available human metagenomic datasets from around the world, as well as murine samples from multiple strains and laboratory mice, against a comprehensive database. All available RefSeq strains for each matched bacterial species, subsampled down to 100 random strains if more were available for a given species, were selected for inclusion. Additionally, all records for all matched genera present in the “representative” or “reference” RefSeq genome were also added, in addition to 3 representative archaea.

Both the streamlined representative database as well as the denser but human-specific reference databases based on RefSeq v87 occupy less than 110GB of RAM when formatted for the BURST aligner, making them suitable for performing exhaustive metagenomic alignment on increasingly common 128GB RAM workstations. Typical human and mouse datasets show 50-80% of metagenomics reads aligning at 95% sequence identity or higher.

Databases are available on S3 upon request and also on a Google Drive public folder:

https://drive.google.com/drive/folders/0B_nJId0uAM7oYnJJTTBndl9zTGs?usp=sharing

Lossless nucleotide sequence compression with kafan.

I have developed a compression codec called “kafan” based on a heavily modified and extended version of the LZ78 algorithm [9]. An early edition of kafan appeared in NINJA-OPS [10] using the TCF wrapper. My initial motivation its development was to integrate extremely high-throughput and compact compression code for use as the concatesome format, as distribution of NINJA-OPS databases required small files due to hosting limitations at the time. Indeed, the entirety of the original implementation in NINJA-OPS consists of a single function, where the entire compliable code for concatesome creation, preprocessing, and compression is 160 lines of C including comments, debug code, and user help strings. Decompression is comparably straightforward.

Kafan was later extended into a general-purpose blockwise binary compression codec with multiple compression speed presets (Figure 24). The default speed preset is an order of magnitude faster than default gzip compression/decompression (Table 4) yet also offers a superior compression ratio for such sequence databases (along with extreme code portability).

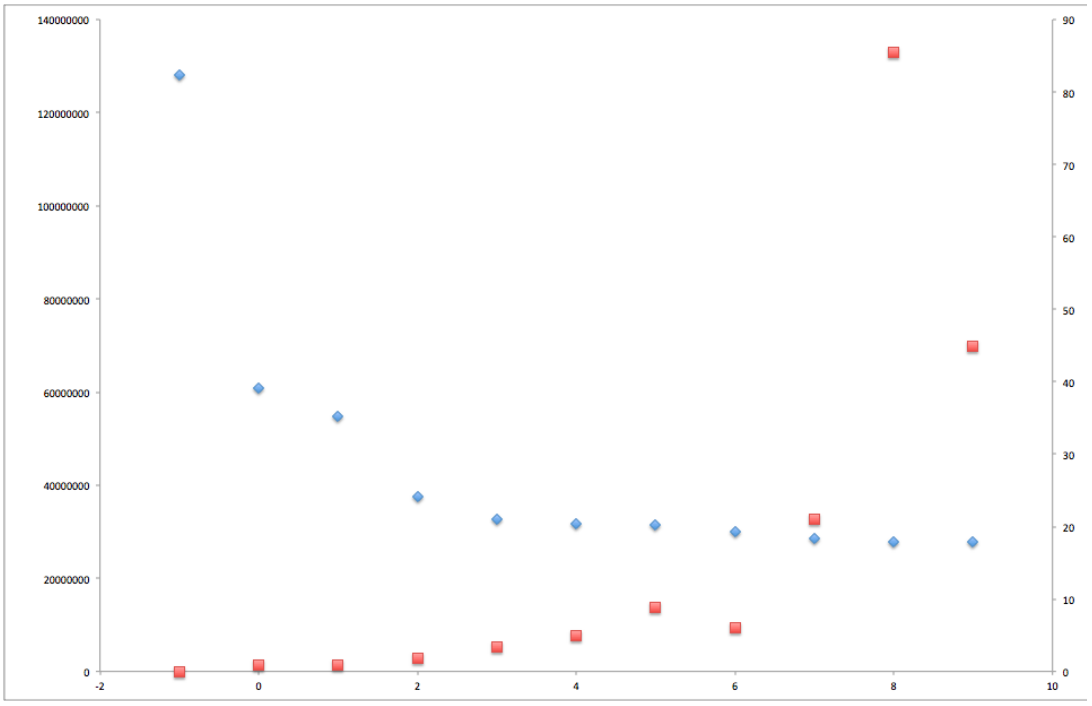


Figure 24. Kafan speed vs compression. The blue series represents filesize (left axis, in bytes) and the red represents compression speed (right axis, in seconds). The x axis shows compression speed preset, with -1 being no compression (raw file).

Table 5. Comparison of kafan and gzip (GZ) modes with no preprocessing.

Mode	Compression (s)	Decompression (s)	Size
0	500	96	44.0GB
1	570	175	42.4GB
2	800	144	35.7GB
3	1170	200	33.3GB
GZ1	2500	1250	36.8GB
GZ5	13,100	Not measured	27.7GB

The primary algorithmic developments of kafan over vanilla LZ78 are support for block-wise compression, dynamic multi-pass dictionary pruning and regeneration (implemented in mode >

7), and modified dictionary initialization and no-expansion guarantee for text/sequence compression (implemented in mode 0). Pre-filtering options such as implemented in NINJA-OPS TCF also improve compression performance for sequence data by separating the binary data for sequence headers from sequences, and sorting to allow common prefixes to occur together (similar to an effect associated with Burrows-Wheeler compression [11]).

There are widespread implications for an algorithm that is both tremendously faster and somewhat more efficient than gzip on nucleotide data, including more efficient storage, retrieval, and transmission of sequence data, which is growing exponentially [12].

Source and binaries are available on GitHub: https://github.com/knights-lab/kafan

Fast heuristic alignment, classification, and taxonomy assignment with UTree.

UTree is a general-purpose, ultra-fast k-mer mapper and DNA sequence classifier. It is the fastest known mapper/classifier known to the author at the time of writing, outperforming the current widely-used k-mer-based metagenomic taxonomy classifier kraken [] by approximately two-fold in alignment performance, 10-fold in database creation speed, and 10-fold in memory usage. It supports inline reverse-complementing of the query sequences, and two voting schemes including plurality vote and LCA (Figure 25).

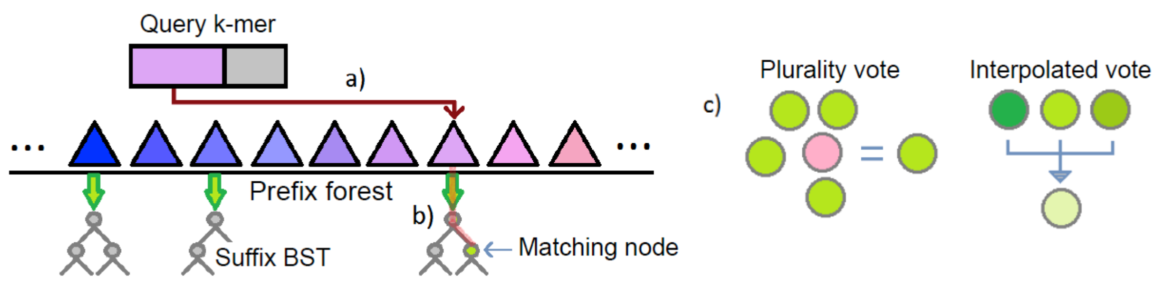


Figure 25. Illustration of UTree's operation. A prefix forest of predefined size is allocated, followed by the creation of suffix trees at the tips representing unique k-mers in a given species. Querying the database involves using a k-mer's prefix to map directly into the prefix forest (a), followed by traversing the suffix tree in search of its suffix (b). If multiple k-mers match within a read, the algorithm votes using different possible methods (c) to determine what taxonomic assignment can confidently be made.

There are numerous advantages of UTree over other aligners, such as my BURST aligner [Chapter 3]. For the simpler biological objective of per-read taxonomic classification, the accuracy remains sufficiently high, if a bit more conservative (lower specificity). Yet the speed of operation is orders of magnitudes faster, and the memory usage an order of magnitude less. For high-throuhgput metagenomic profiling on a personal laptop computer in minutes, UTree is the only viable solution known to date. Additionally, UTree is capable of producing a reference database of nearly 100,000 prokaryotes (spanning over 10,000 unique species) – the entirety of RefSeq's assembly records – in a memory and disk footprint of approximately 30GB while retaining high sensitivity and specificity at the species level.

Further, UTree allows customizable amounts of lossy database compression. The previous statistics presented were for the default compression level of 2, but compression can be tuned from 0 to 4. Each increasing compression level affords a 4-fold reduction in database size over the previous at a cost of less than 50% reduction of specificity per-read (or approximately 10% when aggregated over an entire sample). Higher compression levels may allow metagenomic profiling to operate with reasonable performance on a mobile phone or tablet.

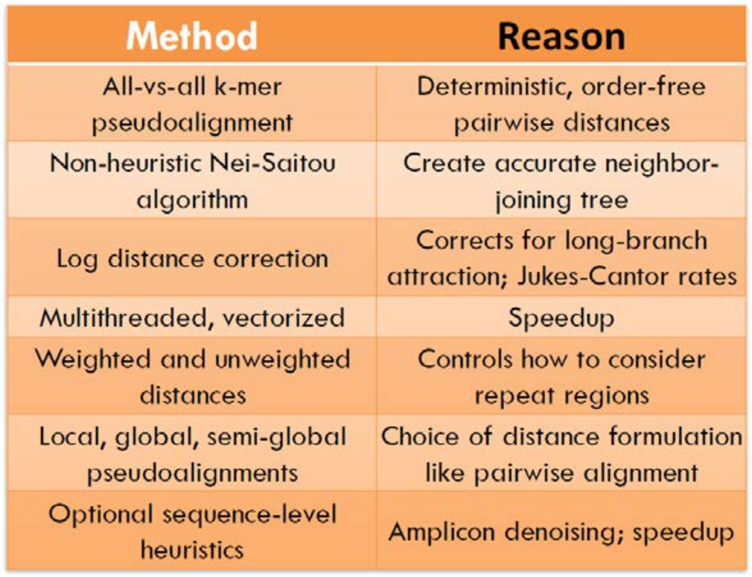
Unlike BURST and other short-read methods considered here, UTree is capable of classifying long reads and even entire genomes. It can simulate database alignment of long-reads simply by building a database out of reference genomes and providing a taxonomy for each genome including a specific genome identifier as the strain label. UTree is also capable of handling arbitrary ranks in hierarchical taxonomic data, which can be used to novel and powerful effect. For instance, the user can provide it root-to-leaf node traversal data in lieu of traditional taxonomy, including phylogenetic tree splits, functional annotations,

UTree is used by the SHOGUN engine [13] and is available on GitHub: https://github.com/knights-lab/UTree

Alignment-free phylogenetics and metagenomic sample comparison with aKronyMer.

aKronyMer is a dependency-free, standalone software that dramatically enhances and accelerates both pairwise sequence comparison and phylogenetic tree construction using a variety of methods (Table 6).

Table 6. Overview of algorithmic optimizations in aKronyMer. aKronyMer derives its accuracy and performance characteristics from a combination of algorithm and feature optimizations. aKronyMer can operate on individual sequence data (such as within a sample) or on entire samples within a dataset.



Method	Reason
All-vs-all k-mer pseudoalignment	Deterministic, order-free pairwise distances
Non-heuristic Nei-Saitou algorithm	Create accurate neighbor-joining tree
Log distance correction	Corrects for long-branch attraction; Jukes-Cantor rates
Multithreaded, vectorized	Speedup
Weighted and unweighted distances	Controls how to consider repeat regions
Local, global, semi-global pseudoalignments	Choice of distance formulation like pairwise alignment
Optional sequence-level heuristics	Amplicon denoising; speedup

Huge phylogeny creation

A critical component in the study of microbial ecology, taxonomy, and evolution, including all fields that utilize principles of these, is the phylogenetic tree. This tree models the evolutionary distances (and in essence, the similarities) between microbes hierarchically. Such trees are also used for metagenomic sample comparison through the use of kernels such as UniFrac [14].

However, producing phylogenetic trees on large numbers (>10,000) organisms is complex and time-consuming, and quickly becomes intractable for conventional methods to produce for NGS-scale data. Some methods exist that attempt to divide the phylogeny creation problem into smaller subsets [15] but most require creation of multiple-sequence alignments (MSA) to produce an initial alignment, followed by a dedicated phylogenetic tree creation software [4] to produce the final tree.

However, it has been found that pairwise distance-based methods for phylogenetic tree creation such as Nei-Saitou [16] perform almost equally well as more complicated methods such as minimum evolution, maximum parsimony, or maximum likelihood as the number of sequences considered increases, especially when read lengths are comparatively short [17]. Therefore, there are 2 primary bottlenecks to forming massive phylogenetic trees: the creation of the multiple sequence alignment, and the formation of the tree itself, both of which are computationally intensive. Furthermore, rearrangements and other structural changes within genomes or marker genes may cause alignment-based techniques to fail to accurately form multiple-sequence alignments, and consideration of these events would further slow the alignment process. Additionally, the use of longer sequences with which to create a phylogeny require exponentially more time in both the alignment and tree construction stages.

To address all of these issues, aKronyMer is a k-mer-based pseudoalignment method that can take a single FASTA file containing marker genes or even entire genomes, and produce a non-heuristic evolution-corrected Nei-Saitou phylogenetic tree in a single step. aKronyMer supports pseudo-local and pseudo-global sequence comparison, adjusts for sequence dissimilarity due to differential sequence lengths, and implements a Jukes-Cantor-like distance correction [18] shown to improve phylogenic accuracy as well as diminish long-branch-attraction artifacts in the resulting tree.

In our test datasets, the Robinson-Foulds distance [19] between an aKronyMer-produced tree of 500 randomly selected 16S sequences reaches over 75% (a high similarity) with a rigorous standard tree using MAFFT G-INS-i and raxml-ng, and this number is expected to increase with the addition of more sequences. From initiation to completion of the process took aKronyMer less than 200ms, and the standard method over 2 hours on the same dual-core laptop.

This level of performance allows a completely *de novo* phylogenetic amplicon analysis given just the denoised starting amplicons and is suitable for use in UniFrac and other diversity analysis (Figure 26). Furthermore, the speed and performance of aKronyMer allows the creation of genome-scale alignments, demonstrated by a phylogenomic tree of approximately 100,000 RefSeq prokaryotes (whole genomes) in under 6 hours on a workstation computer (Figure 27).

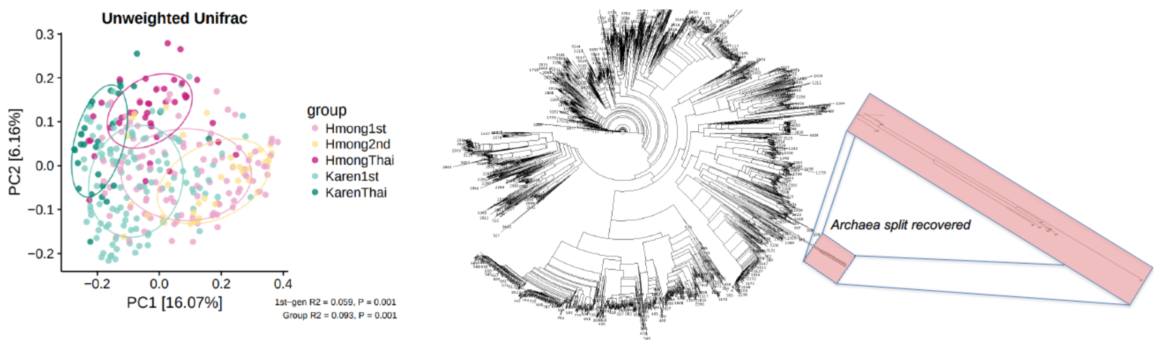


Figure 26. Rapid de novo amplicon phylogeny. Samples collected in the Immigrant Microbiome Project cluster by group in a phylogenetic ordination. Left: Unweighted UniFrac principle components ordination; p-values show clustering is significant by PERMANOVA. Middle: aKronyMer phylogeny of representative amplicons (no database). Right (truncated): single clade exclusively contains all archaea in dataset (validated independently by exhaustive database alignment of these amplicons).

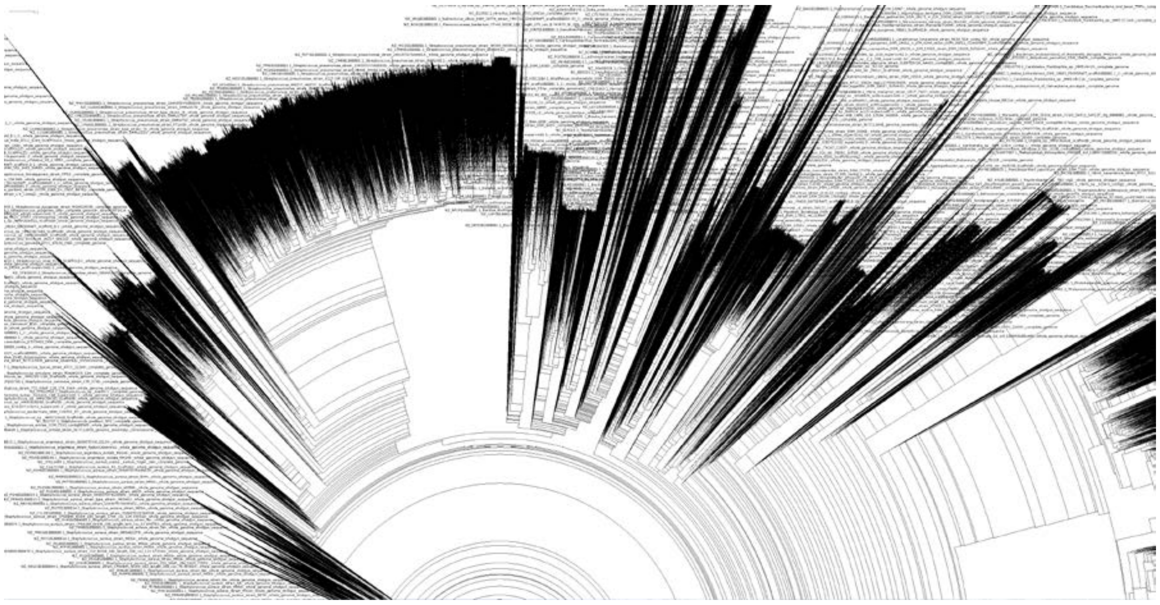


Figure 27. Snapshot of aKronyMer phylogeny from ~100,000 full genomes.

Database- and phylogeny-free metagenomic comparison

Metagenomes are high-dimensional mixtures of microorganisms playing complex ecological roles. Because of the gaps of knowledge reflected in our incomplete databases of microbes, there is great difficulty characterizing the precise strain composition of a community. Yet such a breakdown (a “census” or “tally” or “profile”) is often considered essential for performing other comparative analyses between microbiomes. These downstream analyses are often characterized by diversity and/or sample similarity calculations, which are then used to determine whether two given microbiomes are likely from the same group (geographic location, treatment status, etc), or otherwise assess aggregate diversity statistics within or between groups or samples. Another usage for individual microbes is as features for machine learning models.

However, for many of these use-cases, the determination of the abundances of which known (or assumed) microbes is unnecessary if a method existed which could directly produce distance matrices (beta-diversity), as well as alpha-diversity measures, for all samples. For use in downstream statistical or machine learning, the distance matrix itself can be used as the input features as a *de facto* kernel, and such techniques have been shown to be comparable to raw feature-based techniques [20].

aKronyMer is capable of producing highly accurate sample comparisons, and outputs alpha diversity scores per sample, beta diversity matrices, and sample comparison trees representing inter-sample distances. The raw sample data (both amplicon and whole-genome shotgun data is supported) is simply provided in FASTA format (a tool exists to convert sample sequence files to the required merged records), and results are output such as in Figure 28.

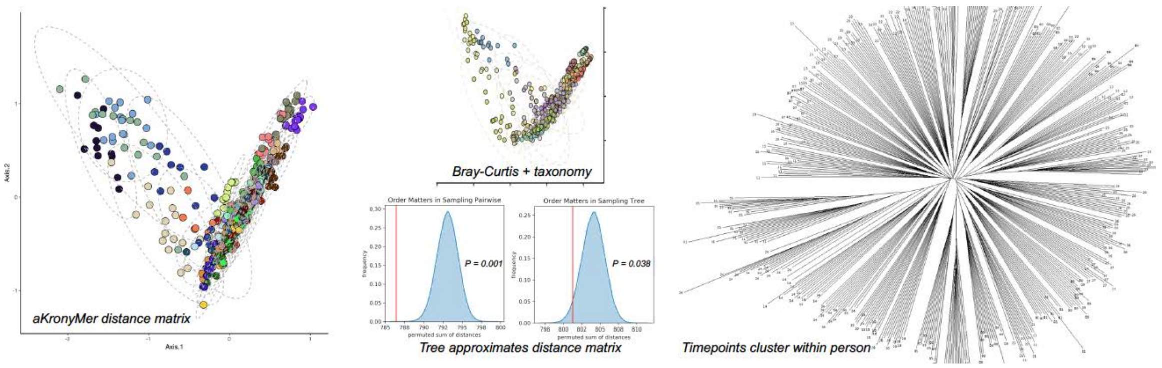
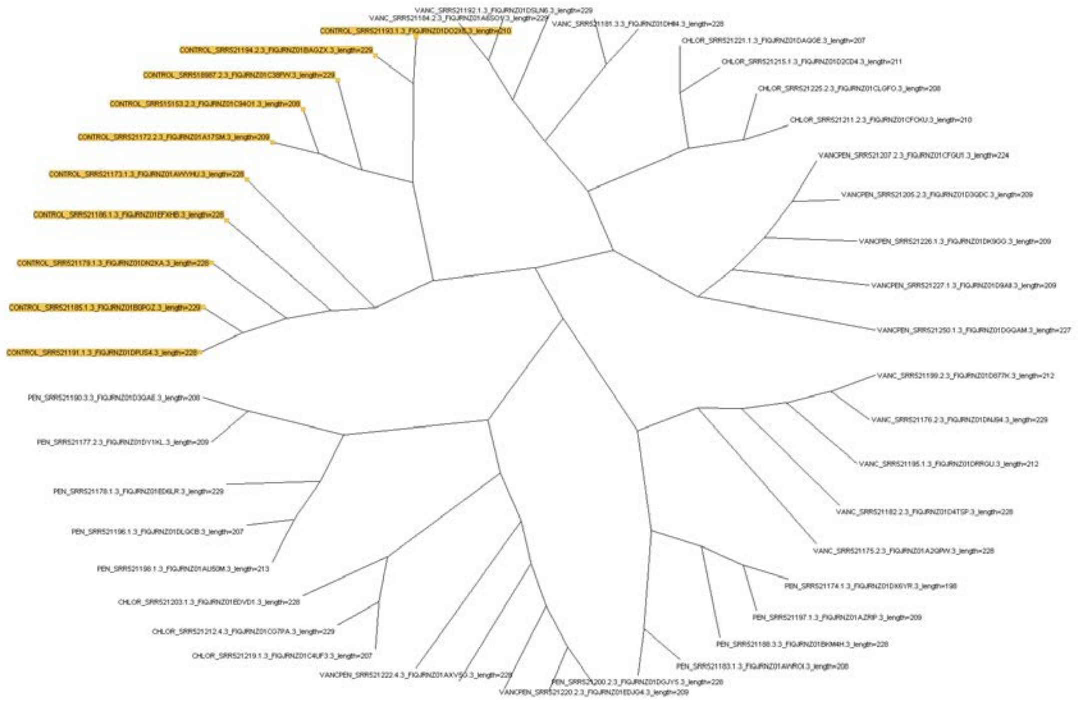


Figure 28. aKronyMer analysis of gut timeseries. aKronyMer can output a distance matrix, which can be ordinated directly with principle coordinate analysis (left). This distance matrix is nearly identical to one produced from exhaustive database alignment and creation of features followed by Bray-Curtis ordination of features (top-middle). aKronyMer also outputs a sample tree output (right) that may better model relationships among samples, evidenced by the pizza-like slices grouping together each person's multiple timepoints. Both the distance matrix and tree are capable of recapitulating the chronological ordering of the timepoints (bottom-middle).

To demonstrate the capabilities of aKronyMer for database-free sample comparison, I evaluated whether aKronyMer could recover known trends in existing datasets. In a public 16S amplicon dataset containing samples from patients on antibiotics and healthy controls [21], aKronyMer successfully separates the controls from the antibiotic classes (Figure 29). The algorithm also correctly separates by body site, exemplified by both amplicon and whole-genome shotgun sequencing (Figure 30).



VANC_SRR521192.1.3_FIQJRNZ01DSLN6.3_length=229
VANC_SRR521184.2.3_FIQJRNZ01A6SOV.3_length=229
VANC_SRR521181.3.3_FIQJRNZ01DHI4.3_length=228
CONTROL_SRR521193.1.3_FIQJRNZ01DO2X8.3_length=210
CONTROL_SRR521194.2.3_FIQJRNZ01BAGZX.3_length=229
CONTROL_SRR518987.2.3_FIQJRNZ01C38PW.3_length=229
CONTROL_SRR515153.2.3_FIQJRNZ01C94O1.3_length=208
CONTROL_SRR521172.2.3_FIQJRNZ01A17SM.3_length=209
CONTROL_SRR521173.1.3_FIQJRNZ01AWYHU.3_length=228
CONTROL_SRR521186.1.3_FIQJRNZ01EFXHB.3_length=228
CONTROL_SRR521179.1.3_FIQJRNZ01DN2KA.3_length=228
CONTROL_SRR521185.1.3_FIQJRNZ01B0PGZ.3_length=229
CONTROL_SRR521191.1.3_FIQJRNZ01DPUS4.3_length=228
CHLOR_SRR521221.1.3_FIQJRNZ01DAQGE.3_length=207
CHLOR_SRR521215.1.3_FIQJRNZ01D2CD4.3_length=211
CHLOR_SRR521225.2.3_FIQJRNZ01CLGFO.3_length=208
CHLOR_SRR521211.2.3_FIQJRNZ01CFCKU.3_length=210
VANCPEN_SRR521207.2.3_FIQJRNZ01CFGUI.3_length=224
VANCPEN_SRR521205.2.3_FIQJRNZ01D3GDC.3_length=209
VANCPEN_SRR521226.1.3_FIQJRNZ01DK9GO.3_length=209
VANCPEN_SRR521227.1.3_FIQJRNZ01D9A8.3_length=209
VANCPEN_SRR521250.1.3_FIQJRNZ01DGQAM.3_length=227
VANC_SRR521199.2.3_FIQJRNZ01D677K.3_length=212
VANC_SRR521176.2.3_FIQJRNZ01DNJ94.3_length=229
VANC_SRR521195.1.3_FIQJRNZ01DRRGU.3_length=212
VANC_SRR521182.2.3_FIQJRNZ01D4TSP.3_length=228
VANC_SRR521175.2.3_FIQJRNZ01A2GPW.3_length=228
PEN_SRR521174.1.3_FIQJRNZ01DX6YR.3_length=198
PEN_SRR521197.1.3_FIQJRNZ01AZRIP.3_length=209
PEN_SRR521188.3.3_FIQJRNZ01BKM4H.3_length=228
PEN_SRR521183.1.3_FIQJRNZ01AWROI.3_length=208
PEN_SRR521200.2.3_FIQJRNZ01DGJY5.3_length=228
VANCPEN_SRR521220.2.3_FIQJRNZ01EDJG4.3_length=209
VANCPEN_SRR521222.4.3_FIQJRNZ01AXV50.3_length=228
CHLOR_SRR521219.1.3_FIQJRNZ01C4UF3.3_length=207
CHLOR_SRR521212.4.3_FIQJRNZ01CO7PA.3_length=229
CHLOR_SRR521203.1.3_FIQJRNZ01EDVD1.3_length=228
PEN_SRR521198.1.3_FIQJRNZ01AU50M.3_length=213
PEN_SRR521196.1.3_FIQJRNZ01DLQCB.3_length=207
PEN_SRR521178.1.3_FIQJRNZ01ED6LR.3_length=229
PEN_SRR521177.2.3_FIQJRNZ01DY1KL.3_length=209
PEN_SRR521190.3.3_FIQJRNZ01D3QAE.3_length=208

Figure 29. aKronyMer groups controls distinct from antibiotic users. In an example amplicon dataset [21], control patients (yellow) cluster separately from various antibiotic classes.

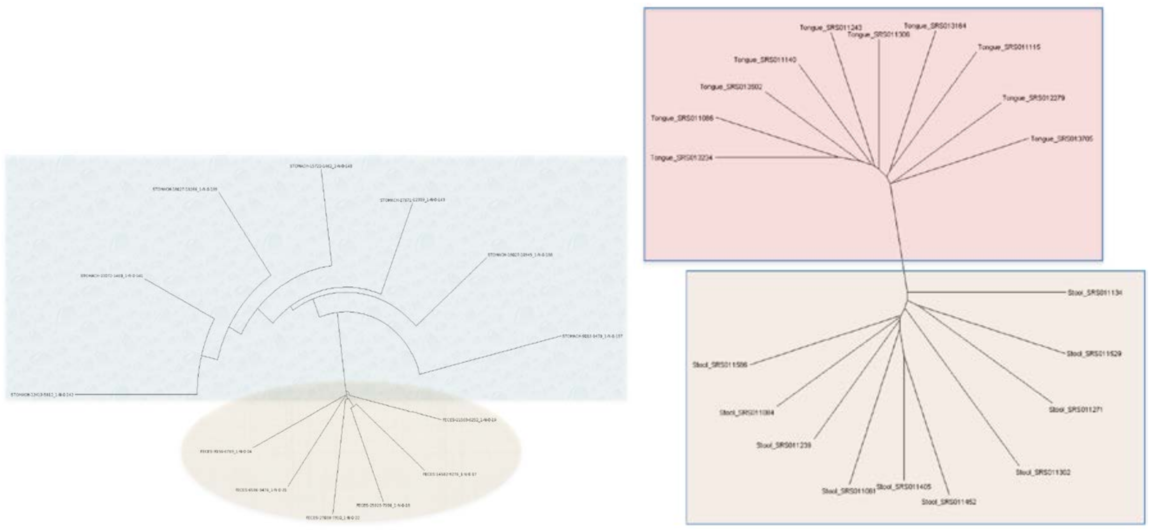


Figure 30. aKronyMer distinguishes between body sites. Primate Microbiome Project amplicon samples (left) is shown to separate by stool (brown) and stomach (teal) samples. Alpha diversity was also significantly different between the stomach and stool groups (p < 0.03). Human Microbiome Project [22] whole-genome shotgun samples (right) subset to 250,000 reads per sample also separate by stool (brown) and tongue (red) samples.

aKronyMer is available on GitHub: https://github.com/knights-lab/aKronyMer

Pipeline for *de novo* amplicon sequencing.

I have also developed a *de novo* amplicon clustering / denoising pipeline exclusively utilizing my other tools. This pipeline operates by performing basic quality control using the SHI7 pipeline [23; Chapter 1]. It then de-duplicates amplicon sequences according to an exact (with allowable subsets/overhangs) deduplication using the ninja_filter module of NINJA-OPS [10; Chapter 2] according to expected number of errors in a read given read length and sequencing platform. These deduplicated sequences become the “representative set” of amplicons in the dataset. The pipeline then re-aligns the entire dataset against the deduplicated “representative set” at a lower identity (usually 98% or greater identity, or dynamically chosen so that 95% of the raw reads align, assuming 5% outliers) using the “capitalist” mode of BURST [Chapter 3] and aggregated into an OTU table. A phylogenetic tree is then created from the representative set using aKronyMer [Chapter 4]. Optionally, if a database is available, BURST is also used to assign taxonomy using its lowest common ancestor mode [LCA] to all sequences in the representative set. The alignment file, phylogenetic tree, and (optional) taxonomy file are sufficient for all amplicon analyses. Runtime performance of the entire de novo pipeline is from 60 seconds to 10 minutes on typical amplicon datasets following quality control.

Non-parametric baseline-independent statistical comparator.

I have developed a permutation-based technique for assessing the statistical significance between two groups with pre- and post- sampling groups and multiple (potentially varying in number) repeated samples per group. The premise is to first calculate the sum of the differences between all microbial abundances between the two timepoints for each group, and normalize by the number of all repeated samplings per group. The group labels are shuffled within each person, preserving the number of repeated samples in the original groupings, and the former calculation is repeated and compared with the original distances. This permutation and re-calculation step are repeated many times (such as 1 million), and the proportion of times the permuted distances are greater than or equal to the originals is reported as the empirical p-value.

Because this technique uses linear algebra techniques with hardware accelerated manipulation in R, it is capable of calculating the permuted distances of each group to its baseline simultaneously, speeding up the process hundreds-fold and allowing hundreds of microbial taxa to be compared with one million iterations on a desktop computer in under an hour. This also allows p-value resolutions sufficient for multiple-hypothesis correction, a weakness of other permutation-based tests that are only capable of running few permutations in a reasonable amount of time.

This code is available on GitHub: https://github.com/knights-lab/megaperm

Toolset for metagenomic alignment summarization.

I have created a variety of convenience utilities to streamline and integrate various tools I have created. These are referred to as “embalmlets” and perform a variety of tasks such as linearizing genomes comprised of multiple contigs while separating out plasmids (“lingenome”), creating text-format semi-colon delimited taxonomic annotations for arbitrary NCBI data in a memory and time-efficient way (“a2gg”), and performing aggregation of alignment results from BURST.

This aggregation can be simple, such as creating a relative abundance table (“embalmulate”), or more complex, utilizing position information in the alignment table to perform coverage analysis (“bcov”). The former enables conventional microbiome analyses such as amplicon studies; the latter is more useful in determining whether a given genome is actually present in a community by assessing the fraction of its genome is actually spanned by query reads. This coverage analysis is important for database-dependent strain detection, and performs well as a classifier for strain presence/absence (Figure 31).

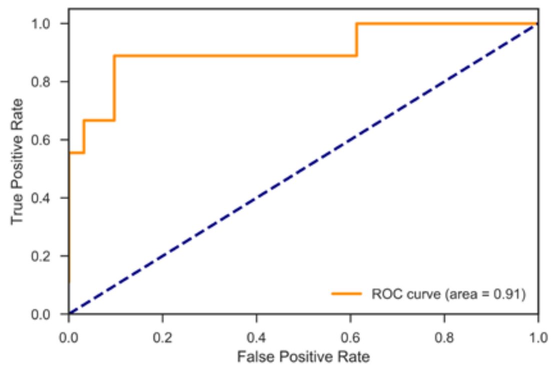


Figure 31. Performance of bcov for strain detection in an example dataset. Area under the curve (“AUC”) is a measure of classifier performance in machine learning. 0.91 is a high score.

These tools are available on GitHub: https://github.com/knights-lab/BURST/tree/master/embalmlets

Microbial coalition-based feature generator.

Metagenomic data analysis is characterized by highly multidimensional data such as the genomes present in a microbial community. However, treating each microbe as an independent variable may be untenable given the ecological dynamics (cooperation, competition) that exist between microbes. This is a particularly important consideration when performing pairwise statistical tests or machine learning, where either multiple hypothesis correction is performed or feature selection is used. This is because these strategies often do not account specifically for community dynamics and instead only reduce or account for number of features, but not the higher-order associations between them. I have produced a software called WRANGLr, which uses greedy feature recombination and Nei-Saitou-inspired clustering by correlation with an outcome variable, to facilitate feature creation through aggregation instead of individual microbes.

By considering microbes as coalitions, groups of microbes are found to form cooperative or competitive units through mutual correlation or anti-correlation with respect to an outcome variable such as treatment group. Preliminary analyses indicate that use of these aggregates instead of individual microbes results in modest but consistent performance improvements in statistical and machine learning tasks. Additionally, studying the members of these microbial aggregates may lead to new biological insights (i.e. through enrichment analysis for factors such as shared ecology, functional capabilities, or phylogenetic relatedness).

Bibliography

INTRODUCTION.

1. Martin M. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet. journal. 2011 May 2;17(1):pp-10.

2. Magoč T, Salzberg SL. FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics. 2011 Sep 7;27(21):2957-63.

3. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, Fierer N, Pena AG, Goodrich JK, Gordon JI, Huttley GA. QIIME allows analysis of high-throughput community sequencing data. Nature methods. 2010 May;7(5):335.

4. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014 Apr 1;30(15):2114-20.

5. Al-Ghalith GA, Montassier E, Ward HN, Knights D. NINJA-OPS: fast accurate marker gene alignment using concatenated ribosomes. PLoS computational biology. 2016 Jan 28;12(1):e1004658.

6. Edgar RC. Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010 Aug 12;26(19):2460-1.

7. Kopylova E, Noé L, Touzet H. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. Bioinformatics. 2012 Oct 15;28(24):3211-7.

8. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nature methods. 2012 Apr;9(4):357.

9. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. Journal of molecular biology. 1990 Oct 5;215(3):403-10.

10. Lozupone C, Knight R. UniFrac: a new phylogenetic method for comparing microbial communities. Applied and environmental microbiology. 2005 Dec 1;71(12):8228-35.

I. SHI7: A self-learning pipeline for multi-purpose short-read DNA quality control.

1. Kahvejian A, Quackenbush J, Thompson JF. 2008. What would you do if you could sequence everything? Nat Biotechnol 26:1125–1133.

2. Kuczynski J, Lauber CL, Walters WA, Parfrey LW, Clemente JC, Gevers D, Knight R. 2012. Experimental and analytical tools for studying the human microbiome. Nat Rev Genet 13:47–58.

3. Langmead B, Salzberg SL. 2012. Fast gapped-read alignment with Bowtie 2. Nat Methods 9:357–359.

4. Hamady M, Walker JJ, Harris JK, Gold NJ, Knight R. 2008. Error-correcting barcoded primers allow hundreds of samples to be pyrosequenced in multiplex. Nat Methods 5:235–237.

5. Bolger AM, Lohse M, Usadel B. 2014. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30:2114–2120.

6. Magoč T, Salzberg SL. 2011. FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics 27:2957–2963.

7. Kuczynski J, Stombaugh J, Walters WA, González A, Caporaso JG, Knight R. 2012. Using QIIME to analyze 16S rRNA gene sequences from microbial communities. Curr Protoc Microbiol 1E–5.

8. Oligos File - mothur.

9. Human Microbiome Project Consortium. 2012. Structure, function and diversity of the healthy human microbiome. Nature 486:207–214.

10. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. 1990. Basic local alignment search tool. J Mol Biol 215:403–410.

11. Johnson M, Zaretskaya I, Raytselis Y, Merezhuk Y, McGinnis S, Madden TL. 2008. NCBI BLAST: a better web interface. Nucleic Acids Res 36:W5–W9.

12. Clayton JB, Vangay P, Huang H, Ward T, Hillmann BM, Al-Ghalith GA, Travis DA, Long HT, Tuan BV, Minh VV, Cabana F, Nadler T, Toddes B, Murphy T, Glander KE, Johnson TJ, Knights D. 2016. Captivity humanizes the primate microbiome. Proc Natl Acad Sci 113:10376–10381.

13. Al-Ghalith GA, Montassier E, Ward HN, Knights D. 2016. NINJA-OPS: Fast Accurate Marker Gene Alignment Using Concatenated Ribosomes. PLOS Comput Biol 12:e1004658.

14. Edgar RC. 2010. Search and clustering orders of magnitude faster than BLAST. Bioinformatics 26:2460–2461.

15. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, Fierer N, Pena AG, Goodrich JK, Gordon JI. 2010. QIIME allows analysis of high-throughput community sequencing data. Nat Methods 7:335–336.

16. Al-Ghalith G, Hillmann B, Ang K, Shields-Cutler R, Knights D. 2017. SHI7: A Streamlined short-read iterative trimming pipeline. Zenodo.

17. Turnbaugh PJ, Ley RE, Hamady M, Fraser-Liggett C, Knight R, Gordon JI. 2007. The human microbiome project: exploring the microbial part of ourselves in a changing world. Nature 449:804.

18. Comeau AM, Douglas GM, Langille MG. 2017. Microbiome Helper: a Custom and Streamlined Workflow for Microbiome Research. mSystems 2:e00127-16.

19. Blanca J. 2017. seq_crumbs: Little sequence file utilities meant to work within Unix pipelines. Python.

II. NINJA-OPS: fast, accurate marker gene alignment using concatenated ribosomes.

1. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, et al. QIIME allows analysis of high-throughput community sequencing data. Nat Methods. 2010;7: 335–336.

2. Huttenhower C, Gevers D, Knight R, Abubucker S, Badger JH, Chinwalla AT, et al. Structure, function and diversity of the healthy human microbiome. Nature. 2012;486: 207–214.

3. Sunagawa S, Mende DR, Zeller G, Izquierdo-Carrasco F, Berger SA, Kultima JR, et al. Metagenomic species profiling using universal phylogenetic marker genes. Nat Methods. 2013;10: 1196–1199.

4. Stahringer SS, Clemente JC, Corley RP, Hewitt J, Knights D, Walters WA, et al. Nurture trumps nature in a longitudinal survey of salivary bacterial communities in twins from early adolescence to early adulthood. Genome Res. Cold Spring Harbor Lab; 2012;22: 2146–2152.

5. Caporaso JG, Lauber CL, Costello EK, Berg-Lyons D, Gonzalez A, Stombaugh J, et al. Moving pictures of the human microbiome. Genome Biol. BioMed Central Ltd; 2011;12: R50.

6. Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, et al. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol. 2009;75: 7537–7541.

7. Edgar RC. Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010;26: 2460–2461.

8. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22: 1658–1659.

9. Wang X, Yao J, Sun Y, Mai V. M-pick, a modularity-based method for OTU picking of 16S rRNA sequences. BMC Bioinformatics. 2013;14: 43.

10. Schloss PD, Westcott SL. Assessing and improving methods used in operational taxonomic unit-based approaches for 16S rRNA gene sequence analysis. Appl Environ Microbiol. 2011;77: 3219–3226.

11. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009;25: 1754–1760.

12. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9: 357–359.

13. Langmead B. Aligning short sequencing reads with Bowtie. Curr Protoc Bioinformatics. 2010;Chapter 11: Unit 11.7.

14. Martínez H, Tárraga J, Medina I, Barrachina S, Castillo M, Dopazo J, et al. A Dynamic Pipeline for RNA Sequencing on Multicore Processors. Proceedings of the 20th European MPI Users' Group Meeting. New York, NY, USA: ACM; 2013. pp. 235–240.

15. Martínez H, Tárraga J, Medina I, Barrachina S, Castillo M, Dopazo J, et al. Concurrent and Accurate RNA Sequencing on Multicore Platforms. arXiv [q-bio.GN]. 2013.

16. Li R, Yu C, Li Y, Lam T-W, Yiu S-M, Kristiansen K, et al. SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics. 2009;25: 1966–1967.

17. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009;25: 2078–2079.

18. Hildebrand F, Tito R, Voigt A, Bork P, Raes J. Correction: LotuS: an efficient and user-friendly OTU processing pipeline. Microbiome. 2014;2: 37.

19. Kopylova E, Noé L, Touzet H. SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. Bioinformatics. 2012;28: 3211–3217.

20. Bentley JL, Sedgewick R. Fast algorithms for sorting and searching strings. SODA. 1997. pp. 360–369.

21. Tárraga J, Arnau V, Martínez H, Moreno R, Cazorla D, Salavert-Torres J, et al. Acceleration of short and long DNA read mapping without loss of accuracy using suffix array. Bioinformatics. 2014;30: 3396–3398.

22. Zhao Z, Yin J, Xiong W. GPU-accelerated Burrow-Wheeler Transform for genomic data. Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on. 2012. pp. 889–892.

23. Gevers D, Kugathasan S, Denson LA, Vázquez-Baeza Y, Van Treuren W, Ren B, et al. The treatment-naive microbiome in new-onset Crohn's disease. Cell Host Microbe. 2014;15: 382–392.

24. Schoch CL, Seifert KA, Huhndorf S, Robert V, Spouge JL, Levesque CA, et al. Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi. Proc Natl Acad Sci U S A. 2012;109: 6241–6246.

25. Mollet C, Drancourt M, Raoult D. rpoB sequence analysis as a novel basis for bacterial identification. Mol Microbiol. 1997;26: 1005–1011.

26. Links MG, Dumonceaux TJ, Hemmingsen SM, Hill JE. The chaperonin-60 universal target is a barcode for bacteria that enables de novo assembly of metagenomic sequence data. PLoS One. 2012;7: e49755.

27. Links MG, Chaban B, Hemmingsen SM, Muirhead K, Hill JE. mPUMA: a computational approach to microbiota analysis by de novo assembly of operational taxonomic units based on protein-coding barcode sequences. Microbiome. 2013;1: 23.

III. BURST enables mathematically optimal short-read alignment for big data.

1. Fritz MH-Y, Leinonen R, Cochrane G, Birney E. Efficient storage of high throughput DNA sequencing data using reference-based compression. Genome Res. 2011;

2. Patro R, Mount SM, Kingsford C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. Nat Biotechnol. 2014;32: 462–464. doi:10.1038/nbt.2862

3. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48: 443–453. doi:10.1016/0022-2836(70)90057-4

4. Gotoh O. An improved algorithm for matching biological sequences. J Mol Biol. 1982;162: 705–708. doi:10.1016/0022-2836(82)90398-9

5. Baeza-Yates R, Gonnet GH. A new approach to text searching. Commun ACM. 1992;35: 74–82.

6. Li H, Homer N. A survey of sequence alignment algorithms for next-generation sequencing. Brief Bioinform. 2010;11: 473–483.

7. Chakraborty A, Bandyopadhyay S. FOGSAA: Fast Optimal Global Sequence Alignment Algorithm. Sci Rep. 2013;3. doi:10.1038/srep01746

8. Farrar M. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. Bioinformatics. 2007;23: 156–161. doi:10.1093/bioinformatics/btl582

9. Lam TW, Sung WK, Tam SL, Wong CK, Yiu SM. Compressed indexing and local alignment of DNA. Bioinformatics. 2008;24: 791–797. doi:10.1093/bioinformatics/btn032

10. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9: 357–359.

11. Mande SS, Mohammed MH, Ghosh TS. Classification of metagenomic sequences: methods and challenges. Brief Bioinform. 2012;13: 669–681.

12. Slater GSC, Birney E. Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics. 2005;6: 31.

13. Librado P, Rozas J. DnaSP v5: a software for comprehensive analysis of DNA polymorphism data. Bioinformatics. 2009;25: 1451–1452.

14. Johnson M, Zaretskaya I, Raytselis Y, Merezhuk Y, McGinnis S, Madden TL. NCBI BLAST: a better web interface. Nucleic Acids Res. 2008;36: W5–W9.

15. Pruitt KD, Tatusova T, Maglott DR. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Res. 2006;35: D61–D65.

16. Yujian L, Bo L. A normalized Levenshtein distance metric. IEEE Trans Pattern Anal Mach Intell. 2007;29: 1091–1095.

17. Altschul SF, Erickson BW. Optimal sequence alignment using affine gap costs. Bull Math Biol. 1986;48: 603–616. doi:10.1007/BF02462326

18. Zachariah MA, Crooks GE, Holbrook SR, Brenner SE. A generalized affine gap model significantly improves protein sequence alignment accuracy. Proteins Struct Funct Bioinforma. 2005;58: 329–338.

19. Al-Ghalith GA, Montassier E, Ward HN, Knights D. NINJA-OPS: Fast Accurate Marker Gene Alignment Using Concatenated Ribosomes. PLOS Comput Biol. 2016;12: e1004658. doi:10.1371/journal.pcbi.1004658

20. Knights D, Silverberg MS, Weersma RK, Gevers D, Dijkstra G, Huang H, et al. Complex host genetics influence the microbiome in inflammatory bowel disease. Genome Med. 2014;6: 1–11. doi:10.1186/s13073-014-0107-1

21. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, et al. Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl Environ Microbiol. 2006;72: 5069–5072. doi:10.1128/AEM.03006-05

22. Zhong Z, Zhang W, Song Y, Liu W, Xu H, Xi X, et al. Comparative genomic analysis of the genus Enterococcus. Microbiol Res. 2017;196: 95–105. doi:10.1016/j.micres.2016.12.009

23. Švec P, Vancanneyt M, Devriese LA, Naser SM, Snauwaert C, Lefebvre K, et al. Enterococcus aquimarinus sp. nov., isolated from sea water. Int J Syst Evol Microbiol. 2005;55: 2183–2187. doi:10.1099/ijs.0.63722-0

24. Ye Y, Doak TG. A Parsimony Approach to Biological Pathway Reconstruction/Inference for Genomes and Metagenomes. PLOS Comput Biol. 2009;5: e1000465. doi:10.1371/journal.pcbi.1000465

25. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The sequence alignment/map format and SAMtools. Bioinformatics. 2009;25: 2078–2079.

IV. Other contributions to comparative genomics and database generation/search.

1. Pruitt KD, Tatusova T, Brown GR, Maglott DR. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. Nucleic acids research. 2011 Nov 24;40(D1):D130-5.

2. Schoch CL, Robbertse B, Robert V, Vu D, Cardinali G, Irinyi L, Meyer W, Nilsson RH, Hughes K, Miller AN, Kirk PM. Finding needles in haystacks: linking scientific names, reference specimens and molecular data for Fungi. Database. 2014 Jan 1;2014.

3. Katoh K, Toh H. Improved accuracy of multiple ncRNA alignment by incorporating structural information into a MAFFT-based framework. BMC bioinformatics. 2008 Dec;9(1):212.

4. Alexey Kozlov. (2018, June 16). amkozlov/raxml-ng: RAxML-NG v0.6.0 BETA (Version 0.6.0). Zenodo. http://doi.org/10.5281/zenodo.1291478

5. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, Huber T, Dalevi D, Hu P, Andersen GL. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. Applied and environmental microbiology. 2006 Jul 1;72(7):5069-72.

6. Abarenkov K, Henrik Nilsson R, Larsson KH, Alexander IJ, Eberhardt U, Erland S, Høiland K, Kjøller R, Larsson E, Pennanen T, Sen R. The UNITE database for molecular identification of fungi–recent updates and future perspectives. New Phytologist. 2010 Apr;186(2):281-5.

7. Fouquier J, Rideout JR, Bolyen E, Chase J, Shiffer A, McDonald D, Knight R, Caporaso JG, Kelley ST. ghost-tree: creating hybrid-gene phylogenetic trees for diversity analyses. Microbiome. 2016 Dec;4(1):11.

8. Ward TL, Dominguez-Bello MG, Heisel T, Al-Ghalith G, Knights D, Gale CA. Development of the Human Mycobiome over the First Month of Life and across Body Sites. MSystems. 2018 Jun 26;3(3):e00140-17.

9. Ziv J, Lempel A. Compression of individual sequences via variable-rate coding. IEEE transactions on Information Theory. 1978 Sep;24(5):530-6.

10. Al-Ghalith GA, Montassier E, Ward HN, Knights D. NINJA-OPS: fast accurate marker gene alignment using concatenated ribosomes. PLoS computational biology. 2016 Jan 28;12(1):e1004658.

11. Burrows M, Wheeler DJ. A block-sorting lossless data compression algorithm. 1994.

12. Cook CE, Bergman MT, Finn RD, Cochrane G, Birney E, Apweiler R. The European Bioinformatics Institute in 2016: data growth and integration. Nucleic acids research. 2015 Dec 15;44(D1):D20-6.

13. Benjamin Hillmann & Dan Knights. (2018, March 5). knights-lab/SHOGUN: SHOGUN: Accurate, scalable metagenomic quantification with shallow shotgun sequencing (Version v1.0.5). Zenodo. http://doi.org/10.5281/zenodo.1188956

14. Lozupone C, Knight R. UniFrac: a new phylogenetic method for comparing microbial communities. Applied and environmental microbiology. 2005 Dec 1;71(12):8228-35.

15. Mirarab S, Nguyen N, Warnow T. PASTA: ultra-large multiple sequence alignment. InInternational Conference on Research in Computational Molecular Biology 2014 Apr 2 (pp. 177-191). Springer, Cham.

16. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular biology and evolution. 1987 Jul 1;4(4):406-25.

17. Tamura K, Nei M, Kumar S. Prospects for inferring very large phylogenies by using the neighbor-joining method. Proceedings of the National Academy of Sciences. 2004 Jul 27;101(30):11030-5.

18. Jukes TH, Cantor CR. Evolution of protein molecules. Mammalian protein metabolism. 1969;3(21):132.

19. Robinson DF, Foulds LR. Comparison of phylogenetic trees. Mathematical biosciences. 1981 Feb 1;53(1-2):131-47.

20. Hofmann T, Schölkopf B, Smola AJ. Kernel methods in machine learning. The annals of statistics. 2008 Jun 1:1171-220.

21. Cho I, Yamanishi S, Cox L, Methé BA, Zavadil J, Li K, Gao Z, Mahana D, Raju K, Teitler I, Li H. Antibiotics in early life alter the murine colonic microbiome and adiposity. Nature. 2012 Aug;488(7413):621.

22. Human Microbiome Project Consortium. 2012. Structure, function and diversity of the healthy human microbiome. Nature 486:207–214.

23. Al-Ghalith GA, Hillmann B, Ang K, Shields-Cutler R, Knights D. SHI7 Is a Self-Learning Pipeline for Multipurpose Short-Read DNA Quality Control. MSystems. 2018 Jun 26;3(3):e00202-17.