One-Pass Diversified Sampling with Application to Terabyte-Scale Genomic Sequence Streams

Benjamin Coleman [*1] **Benito Geordie** [*2] **Li Chou** [3] **R. A. Leo Elworth** [2] **Todd J. Treangen** [2]
Anshumali Shrivastava [2 4]

Abstract

A popular approach to reduce the size of a massive dataset is to apply efficient online sampling to the stream of data as it is read or generated. Online sampling routines are currently restricted to variations of reservoir sampling, where each sample is selected uniformly and independently of other samples. This renders them unsuitable for large-scale applications in computational biology, such as metagenomic community profiling and protein function annotation, which suffer from severe class imbalance. To maintain a representative and diverse sample, we must identify and preferentially select data that are likely to belong to rare classes. We argue that existing schemes for diversity sampling have prohibitive overhead for large-scale problems and high-throughput streams. We propose an efficient sampling routine that uses an online representation of the data distribution as a prefilter to retain elements from rare groups. We apply this method to several genomic data analysis tasks and demonstrate significant speedup in downstream analysis without sacrificing the quality of the results. Because our algorithm is 2x faster and uses 1000x less memory than coreset, reservoir and sketch-based alternatives, we anticipate that it will become a useful preprocessing step for applications with large-scale streaming data.

Whole-genome shotgun sequencing (WGS) has inspired numerous breakthroughs over the past decade in large-scale comparative genomics research. The total amount of WGS data doubles approximately every seven months (Stephens et al., 2015), far exceeding even the pace of computing capabilities predicted by Moore's law. Genomic data facilitates many advances in human health, including pathogen surveillance (Timme et al., 2019), antibiotic resistance detection (Köser et al., 2014), and cancer genomics (Meyerson et al., 2010). Labs are thus motivated to sequence as many organisms as possible, culminating in repositories like the the European Nucleotide Archive (ENA) (Leinonen et al., 2010a), which contains one-fifth of a petabyte of bacterial and viral DNA alone, and the NCBI Short Read Archive (SRA)(Leinonen et al., 2010b). At the time of writing, the SRA contains over 43 petabytes of publicly-hosted data on the AWS cloud. The scale of genomic data rivals that of the largest web-scale companies and government agencies, and the data collection rate is likely to increase thanks to ambitious plans such as the NIH *All of Us* program (Sankar & Parker, 2017), which will sequence 1 million human genomes as part of its personalized medicine initiative. Recent developments in sequencing hardware also threaten to inflate the exponential growth rate. The PromethION system from Oxford Nanopore Technologies has a data rate of over 4 terabytes per run, with a theoretical maximum yield of 15 TB.

This 'data deluge' requires continued innovation for data processing and analysis (Schatz & Langmead, 2013). For example, hardware vendors are actively developing "Read Until" technology – a *real-time selective sequencing* approach that ejects abundant background sequences in favor of scarce diverse sequences present in a sample (Loose et al., 2016; Edwards et al., 2019). When run on fast GPUs, recent base calling algorithms, such as Guppy, can keep up with a single MinION or PromethION data stream by offering 1.5-6 Mbps throughput (Wick et al., 2019). However, it is increasingly challenging to store the data off of the device on local storage, and many downstream methods require storing all k-mers from a dataset in memory. This is untenable for terabyte-scale datasets. To cope with the onslaught of data, the algorithms community must develop streaming algorithms that can both retain the diversity

[*]Equal contribution [1]Department of Electrical and Computer Engineering, Rice University, Houston TX, USA [2]Department of Computer Science, Rice University, Houston, TX, USA [3]Department of Engineering and Computer Science, West Texas A&M University, Canyon TX, USA [4]Third AI, Houston TX, USA. Correspondence to: Benjamin Coleman <brc7@rice.edu>, Benito Geordie <bg31@rice.edu>, Todd J. Treangen <treangen@rice.edu>, Anshumali Shrivastava <anshumali@rice.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

of data contained in massive genomic and metagenomic datasets and also keep pace with the sequencing devices. At this scale, sketching and streaming algorithms are the only way to process massive sequence datasets without a corresponding increase in computational resources (Rowe, 2019).

Diversity Sampling: Inspired by these developments, we consider the streaming diversity sampling problem. In this problem, we are given a data stream $\mathcal{D}$ of elements (or sequences) $x_1, x_2, \ldots, x_N$, which we see one element at a time. The task is to construct a diverse sample $\mathcal{S}$ of $\mathcal{D}$ while fulfilling three properties. (1) *One Pass.* Once we see x_i, we must immediately decide whether to accept or reject it without seeing future samples. (2) *Efficient.* The accept/reject decision must be fast enough to keep pace with the data generation rate. (3) *Low-Memory.* The memory of the algorithm should fit into RAM and scale well with N. These criteria have immediate consequences for algorithm design. Due to latency and RAM limitations, it is infeasible to store lookup structures for similarity search. For petabyte-sized streams with long sequences, it is even difficult to store buffers of sampled elements. Our aim is to perform diverse sampling while achieving the above three properties.

Diversity maximization problems have been explored by the computational geometry community in the context of diverse coresets (Indyk et al., 2014) and diverse near-neighbor search (Abbar et al., 2013b), but diversity is usually defined geometrically (e.g. the minimum pairwise distance between the points of the sample set). However, other diversity measures are more useful and well-known for genomics problems. In this paper, we focus on *biodiversity measures* such as the Simpson index and Shannon index, which roughly measure the fraction of species (classes) that are represented in the sample.

Related Work: One can always obtain a random sample of the dataset via reservoir sampling (Vitter, 1985). Reservoir sampling satisfies the three aforementioned properties, but random selection is undesirable because it may discard important information. We consider several alternatives.

Composable Coresets: A *coreset* is a subset of samples from the dataset $\mathcal{D}$ that approximately preserves the properties of the dataset. Indyk et al. consider *composable* (or mergeable) coresets for diversity maximization, under several geometric notions of diversity (Indyk et al., 2014). Their algorithm breaks the stream into buffers of M elements and constructs a diverse coreset within each buffer. To get the final sample, we merge the coresets from each buffer. The method has strong theoretical guarantees and good performance on news data (Abbar et al., 2013a).

Buffer algorithms: Algorithms such as *deterministic lossy counting* and *randomized sticky sampling* use a buffer of samples to identify frequent elements (Manku & Motwani, 2002). Lossy counting records the frequency of the top M unique items in an array, while sticky sampling allocates a new counter for every new item seen so far. Both algorithms periodically purge the buffer of low-count items, to avoid maintaining counts for infrequent elements.

Sketching Algorithms: The Count-Min sketch (CMS) is an array of counters that can efficiently approximate how frequently an element occurs in a data stream (Cormode & Muthukrishnan, 2005). Diginorm (Brown et al., 2012), Bignorm (Wedemeyer et al., 2017) and NeatFreq (McCorrison et al., 2014) are CMS-based sketching algorithms which attempt to remove redundant, duplicate sequences from the stream, a process referred to as *genetic data normalization*. At a high level, such methods downsample high-frequency k-mers while attempting to retain rare information.

Why are existing methods insufficient? Two key factors limit the utility of existing methods for diversity sampling. First, the focus is often on removing exact duplicates in the stream, which lacks robustness to the slight perturbations introduced by hardware-level read errors. A CMS or lossy counting buffer can answer whether we have previously seen a particular sequence, but not whether we have seen a similar one. Second, the memory and computation often scales poorly with the sample size. Composable coresets can reject similar sequences, but require costly distance computations, while the memory of CMS-based methods scales poorly with the stream size. This is reflected in our experiments, where Diginorm (Brown et al., 2012) requires progressively larger amounts of memory for larger datasets and coresets cannot run on datasets with more than 1 million reads.

Our Contribution: In this paper, we propose a novel sampling algorithm to select elements from a stream while preserving diversity. Inspired by recent advances in the field of density estimation and locality-sensitive hashing (LSH), we use a sketch to quickly estimate the probability of observing each stream item. This diversity score enables a robust inverse propensity sampling process that preserves diversity. The memory depends on the diversity of the stream, rather than the size, and the algorithm is fast enough to run on terabyte-scale data.

1. Background

We present a fast diversity sampling routine that handles high-throughput streams of sequence data. Our method constructs a diverse sample set by rejecting sequences that are similar to the samples which have been collected so far. The critical part of our approach is that we do not perform pairwise sequence similarity comparisons. Instead, we estimate the data distribution from the stream, which

we use to approximate the likelihood of the current element given the past stream elements. This likelihood acts as an inverse diversity score - if the likelihood is low, then the element does not resemble previous data and should be included in the sample.

Density Estimation: To model the distribution of elements in the stream, we require a fast method for density estimation. If the parametric form of the density is known (e.g. Gaussian mixture), then one can use standard methods to fit a distribution to the previous stream observations. However, we often do not know the form of genomics data distributions. WGS sequence datasets are most naturally modeled using string distances that make it difficult to construct and learn a parameterized distribution. Even with an embedding into a more traditional metric space, it is expensive to fit and update a distribution each time the stream outputs new data.

Therefore, we use the kernel density estimate (KDE) to model the streaming distribution. The KDE is a well-established non-parametric statistical method to estimate the data distribution from a collection of points. Given a dataset $\mathcal{D}$, a distance function $d(x, q)$ and a kernel function $\kappa : d(x, q) \to [0, 1]$, the radial KDE is the quantity:

$$\text{KDE}(q) = \sum_{x \in \mathcal{D}} \kappa(d(x, q))$$

Although the naive calculation of the KDE is expensive ($O(N)$), we leverage recent work towards fast kernel sum calculations. We introduce these methods below.

Locality-Sensitive Hashing: (LSH) An LSH family (Indyk & Motwani, 1998) is a family of functions where similar points have a high probability of having the same hash value under the hash mapping.

Definition 1.1. (R, cR, α, β)-sensitive hash family
A family $\mathcal{H}$ is called (R, cR, α, β)-sensitive with respect to a distance function $d(x, y)$ if the following two properties hold for any hash function $h \in \mathcal{H}$ and any points x and y:

$$d(x, y) \leq R \implies \Pr_{\mathcal{H}}[h(x) = h(y)] \geq \alpha$$

$$d(x, y) \geq cR \implies \Pr_{\mathcal{H}}[h(x) = h(y)] \leq \beta$$

LSH Notation: If two points x and y have the same LSH function value ($h(x) = h(y)$), we say that these points *collide*. We use the notation $\rho(x, y)$ to refer to the collision probability $\Pr_{\mathcal{H}}[h(x) = h(y); x, y]$. We may improve the selectivity of an LSH function through a process known as *concatenation*, which packages together several hash values into one output. Suppose we are given an LSH family with collision probability ρ, and we compute n hash functions. The concatenated hash output is the set $[h_1(x), h_2(x)...h_n(x)] \in \mathbb{Z}^n$, and the collision probability of the resulting hash is ρ^n. To transform the values in $\mathbb{Z}^n$ to an integer in the fixed range $[1, B]$, we may *rehash* the LSH output using a universal hash function (Carter & Wegman, 1979). The collision probability becomes $\rho^n \frac{B-1}{B} + \frac{1}{B}$.

MinHash and Genomics: MinHash is sensitive to the Jaccard similarity between sets and is widely used in bioinformatics (Broder, 1997b). In genomics, the sets are sequences, represented as bags of k-mers, and the Jaccard similarity is related to the average nucleotide identity (ANI) (Ondov et al., 2016). Figure 1 shows how to hash a sequence with MinHash.

RACE Sketch: The RACE sketch (Repeated Array of Count Estimators) is a technique to approximate the KDE when the kernel κ is the collision probability of an LSH function (Luo & Shrivastava, 2018; Coleman et al., 2020; Coleman & Shrivastava, 2020). The sketch consists of a 2D integer array $A \in \mathbb{Z}^{B \times R}$ that is indexed using an LSH function. When a new data element x arrives from the stream, we hash x using R different LSH functions and increment the counters in A at the corresponding locations. To approximate $\text{KDE}(q)$, we take the average of the R count values at the indices corresponding to $h(q)$.

Theorem 1.2. *(Coleman & Shrivastava, 2020) Given a dataset $\mathcal{D}$ and an LSH family $\mathcal{H}$ with collision probability $\rho = \kappa(d(x, y))$, create a RACE array A of the dataset with $R = 1$ using an LSH function $h(q)$ sampled uniformly from $\mathcal{H}$. Then for any query q, $\mathbb{E}[A[h(q)]] = \text{KDE}(q)$ with bounded variance.*

$$\text{var}(A[h(q)]) \leq \left(\sum_{x \in \mathcal{D}} \sqrt{\kappa(d(x, q))} \right)^2$$

2. Diversified Sampling via Inverse Propensity

We propose a form of inverse propensity sampling to perform diversified sampling on data streams. Inverse propensity sampling is a type of importance sampling where points are weighted based on the inverse probability of their inclusion in the dataset. The technique is a well-established method to remove selection bias arising from the assignment of examples to treatment groups in statistical studies (Rosenbaum & Rubin, 1983).

Intuition: We observe that propensity estimates can be used to implement diversified sampling. To understand why, consider a mixture model where we first draw a class C with probability p_C from a set of classes and then generate a random element $x \in C$. To maximize traditional measures of biodiversity, we wish to select an equal number of elements from each class. To de-bias the non-uniform probability of selecting class C in the first level of the mixture model, we can use inverse propensity sampling to select elements from C based on the importance weight p_C^{-1}. The result is a set of samples that are uniformly distributed among the classes.

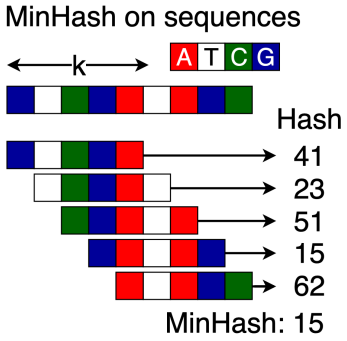

Figure 1. MinHash on gene sequence data. Each k-mer in the sequence is hashed using the same universal (random) hash function, resulting in a set of hash values. The MinHash output value is the smallest of these hash values, or the *minimum hash*.

We would like to use this idea to select points from the data stream, but there are two challenges. First, the propensity p_C is unknown and may even change over time. Second, there is no fast way to determine the class from which each stream element was drawn. We use efficient density estimation to address these problems. If we suppose that the classes are well-separated within the metric space – a reasonable assumption for many metagenomic problems – we may use the KDE of x as our estimate of p_C. If we use RACE as the density estimator and eliminate the computational overhead due to probabilistic sampling, we obtain the sampling method Algorithm 1.

There are several valid string distances (and corresponding kernels) that are appropriate for this situation, including the Hamming distance, Jaccard distance, and edit distance (McCauley, 2019). We consider the Jaccard kernel because it can be implemented very efficiently via the MinHash LSH family (Broder, 1997a) for both genomic and metagenomic applications (Ondov et al., 2016). However, our results and algorithms generally apply to any metric space that supports efficient KDE.

Computation and Memory Cost: Consider the computation and memory requirements of Algorithm 1. Each element in the stream requires R MinHash computations, followed by R counter look-ups in the sketch and R scalar additions to the count value. Our experiments show that R can be as small as 10, leading to a negligible computation cost. Similarly, the memory footprint is also small. Unlike reservoir sampling, frequency sampling or coresets, diversified sampling does not use buffers of sequence data. Instead, we store a $B \times R$ array of integers in RAM. We will later see that the value of B depends on the diversity of the stream, but even with large and diverse datasets we only need $B \leq 1M$ and $R = 10$, or a few megabytes of RAM.

Algorithm 1 Diversified Sampling

Input: Data stream $\mathcal{D}$, threshold τ, repetitions R, array width B, number of LSH concatenations n, size of sequence k-mers k
Output: Diverse sample $\mathcal{S}$
Initialize: R independent LSH functions $\{h_1, \ldots, h_R\}$ with range B and n concatenations
Sample $\mathcal{S} \leftarrow 1$ and $A \leftarrow \mathbf{0}^{B \times R}$
for $x \in \mathcal{D}$ **do**
  Score $s \leftarrow 0$
  for r in 1 to R **do**
    $s = s + A[r, h_r(x)]$
    Increment $A[l, h_l(x)]$
  end for
  $s = s/R$
  if $s < \tau$ **then**
    Append x to $\mathcal{S}$
  end if
end for

3. Theoretical Results

In this section we show that, under reasonable modeling assumptions, Algorithm 1 maximizes the diversity index of the sample. It is well-known that inverse propensity sampling undoes the data generation bias, and there is rich theory corresponding to the use of estimated propensity scores. However, Algorithm 1 is a de-randomized version of the algorithm with approximate scoring, and this requires a specialized analysis.

Our proof sketch is as follows. We start with the assumption that the sequence classes are separable. A sequence class is an abstract term that can refer to a species, Operational Taxonomic Unit (OTU), protein functional group, or any other group with unique gene-level information. Using this assumption, we show that each of the counters within A for our sample $\mathcal{S}$ converges to a constant under Algorithm 1. That is, if we were to construct a RACE sketch using only the sequences in our sample, all of the counters in the array would have the same value. We conclude our analysis with a proof that if a sample has uniform counters, then it has an optimal diversity index. We begin by formally defining the diversity sampling problem.

Definition 3.1. Given a streaming dataset $\mathcal{D}$, consisting of elements $x_1, x_2, \ldots, x_N$ where each element x_i belongs to one of Z different classes $C_1, C_2, \ldots, C_Z$, and a diversity measure $D(\mathcal{S})$, construct a subset $\mathcal{S} \subset \mathcal{D}$ of M samples, where $M \ll N$, such that $D(\mathcal{S})$ is maximized. Namely, find

$$\mathcal{S}^\star = \underset{|\mathcal{S}|=M}{\arg\max}\, D(\mathcal{S})$$

Definition 3.1 is also given in (Indyk et al., 2014) and the problem has been analyzed for a dozen different diversity

measures $D(\mathcal{S})$. However, these measures are low-level geometric functions that do not consider the class labels, with the exception of topic diversity, where we wish to solve a set coverage problem on the topics covered by a sample of news articles. Hence, we adopt more traditional biodiversity measures such as the Simpson, Shannon, and Berger–Parker indices, which demand a subset that has equal representation of as many classes as possible. In practice, our classes may be operational taxonomic units (OTU), species, genera, protein classes, or any other classification unit. We assume that the classes are Δ-*separable*, or distinguishable based on sequence-level features.

Definition 3.2. Δ-separable classes: We say that two classes C_1 and C_2 are Δ-separable if $d(x, y) \geq \Delta$ for all $x \in C_1$ and $y \in C_2$

Under this assumption, we can select parameters that force each labeled class to map to a unique set of counters in the array because the minimum distance Δ bounds the collision probability β from Definition 1.1. Lemma 3.3 is a consequence of standard LSH amplification techniques (Datar et al., 2004).

Lemma 3.3. *Suppose that two classes C_1 and C_2 are Δ-separable and that there exists an $(R, \Delta, \alpha, \beta)$-sensitive hash family $\mathcal{H}$. Then given $\delta \in [0, 1]$, there exists an LSH function $h(x)$ such that $h(x) \neq h(y)$ for all $x \in C_1$ and $y \in C_2$ with probability $1 - \delta$.*

Note that the hash function from Lemma 3.3 sends classes to distinct buckets with probability $1 - \delta$. To obtain a true non-colliding hash, we can verify and retry the construction process if necessary (Las Vegas style). We can also simply allow our subsequent guarantees to hold with probability $1 - \delta$. We will assume the former. Now we consider the discrete set of count values in A. Since the count values are non-decreasing, it is straightforward to see that Algorithm 1 causes them to converge to $\lceil \tau \rceil$.

Lemma 3.4. *Construct a single RACE array with B buckets using an LSH function $h(\cdot)$ on the elements of the sample set $\mathcal{S}$ obtained using Algorithm 1. Then Algorithm 1 causes each of the RACE counters in this array to converge to $\lceil \tau \rceil$.*

Diversity Indices: We measure sample diversity using the Shannon index (1), inverse Simpson index (2), and Berger–Parker index (3).

$$H' = -\sum_{i=1}^{Z} s_i \log\ z_i \tag{1}$$

$$\lambda^{-1} = \left(\sum_{i=1}^{Z} z_i^2 \right)^{-1} \tag{2}$$

$$D_B = \max_{1 \leq i \leq Z} z_i \tag{3}$$

Z is the number of sequence classes in the sample $\mathcal{S}$. The proportion z_i is the ratio n_i/M, where n_i is the number of times that class i appears in the sample and M is the sample size. Assume the classes are Δ-separable and use Lemma 3.3 to ensure that each class maps to a unique set of buckets. We are now ready to express the diversity index in terms of the count values.

Theorem 3.5. *Assume that classes $C_1, \ldots, C_S$ are Δ-separable and use the hash function from Lemma 3.3 to construct a count sketch for the output of Algorithm 1. Then the ratio z_i converges to a constant:*

$$z_i = \frac{n_i}{m} \to \frac{\lceil \tau \rceil B_i}{\lceil \tau \rceil B} = \frac{B_i}{B}$$

B_i is the number of buckets in the hash range of class C_i.

This theorem allows us to prove useful statements about the diversity index. Consider the simple case where all B_i are equal. The following statement is a straightforward consequence of the fact that $z_i \to 1/Z$ and the observation that the diversity indices attain their maximum when $z_i = 1/Z$.

Corollary 3.6. *Suppose each class maps to exactly k buckets (i.e. $B_i = k$). With $B = Zk$ buckets, the Shannon, Simpson, and Berger-Parker diversity measures converge to their maximum value.*

Limitations: In practice, classes do not map to the same number of buckets. However, a number of useful mixture models *approximately* have this property, resulting in a sample with a diversity index that is approximately optimal. For instance, consider a dataset where class C_i consists of a uniform sample on the ball $\mathcal{B}(c_i, r_i)$, which is the set of all sequences y such that $d(y, c_i) < r_i$. If the ball radii $\{r_1, \ldots r_Z\}$ are all the same, then all the classes map to approximately the same number of buckets. This model assumes that the sequences in each class are highly similar to each other, but one can construct other models with the same convergence behavior that do not require this assumption.

It should also be noted that our assumptions about the class k-mer distribution are not necessarily required for good practical performance. Because the diversity index is a precision metric over all classes, we require an equal number of samples from each class to guarantee an optimal diversity index. This leads to the assumptions made by Corollary 3.6. However, many applications only demand high recall, where all classes are represented. For this behavior, Definition 3.2 is sufficient. We empirically verify that Definition 3.2 holds in real datasets in Figure 2.

Practical Implications: Theorem 3.5 indicates that we will oversample classes with high levels of sequence variation. For example, species with large pan-genome sizes will cover a larger number of buckets in the array than those with

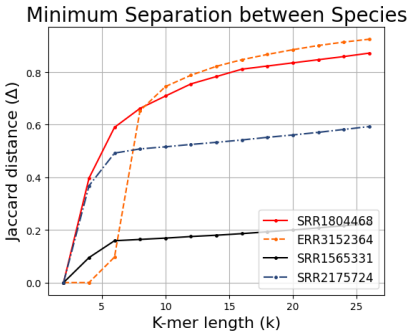

Figure 2. Values of Δ for real-world datasets. Note that the separability of two classes depends on the k-mer length.

small pan-genomes. This means that diversified sampling would likely retain many samples belonging to *Escherichia coli* bacteria in metagenomics applications, because it has an open pan-genome with an incredibly high degree of sequence-level variation (Gordienko et al., 2013). Therefore, it will cover a larger number of buckets and have a higher B_i than a species with a smaller, closed pan-genome, such as *Mycobacterium tuberculosis* (Kavvas et al., 2018). The practical takeaway is that we oversample classes based on the amount of space covered by the class, while other methods sample classes based on the number of sequences belonging to the class.

Relationship to k-mer Abundance: Methods such as Diginorm, Bignorm and NeatFreq estimate the redundancy based on the median abundance of k-mers that appear in a read (Brown et al., 2012; Wedemeyer et al., 2017; McCorrison et al., 2014). If we adopt an analysis based on k-mer abundance rather than class diversity, we can show that our method estimates the redundancy based on the *average* k-mer abundance. Given a read x, let $s(x)$ be the score assigned by our method to the read. The expected value $\mathbb{E}[s(x)]$ is the average abundance over the k-mers that compose x. Observe that MinHash uniformly selects a random k-mer from x, which we hash into a count-min sketch to get $s(x)$. The expectation over MinHash seeds is the average value of the CMS cells that correspond to the k-mers in x. With R repetitions, we get a sharp estimate of $\mathbb{E}[s(x)]$ which we compare to a threshold τ.

Memory Performance: Coresets require roughly ML bytes in RAM, where M is the number of samples and L the average length of each read. For most applications, M must increase with the dataset size (N), often growing at the rate $M = O(N)$. Our method uses RB integers, where R and B do not depend linearly on N. Diginorm, Bignorm and NeatFreq also use an $R \times B$ integer array, but these methods require $R = O(\log N)$ to accurately estimate individual k-mer frequencies (to report the median). Because we use the average instead of the median, we can set $R = O(1)$ while still maintaining constant error (Theorem 1.2).

4. Experiments

To show that diversified sampling is a viable and efficient way to downsample sequence datasets, we conducted four sets of experiments on a broad range of problems. We downloaded real datasets directly from the SRA, ENA, and UniRef archives. We briefly describe the tasks here, with a longer explanation in the appendix.

Terminology: We use the term *dataset* to refer to a set of sequences. A *read* is a sequence of variable length that is output by the low-level sequencing hardware and basecalling software. Protein sequences describe the structure of a protein and have an alphabet size of 20 characters, while DNA sequences have an alphabet of 4 characters {A,T,C,G}. *Metagenomic* datasets contain DNA sequences belonging to multiple organisms, while *genomes* are full, assembled sequences belonging to one species. The *Average Nucleotide Identity* (ANI) is a similarity measure that decides whether two sequences belong to the same species. Table 2 shows the run accession numbers and properties of the datasets used in our evaluation.

Species-Preserving Downsampling: In metagenomic community profiling, we wish to identify low-abundance classes in a dataset (Qin et al., 2010; Segata et al., 2012). Thus, our set of samples must represent as many species as possible if it is to be useful. We evaluate our method by counting the number of species preserved after downsampling short-read (Illumina) and long-read (PromethION / GridION) metagenomes. To obtain the ground truth reference labels, we used the Kraken2 tool (Wood et al., 2019) to annotate each sequence.

While Kraken2 labels have previously been shown to be highly accurate, the process can induce bias if applied to microbiomes with poor representation in the Kraken2 database (Nasko et al., 2018). To address this, we focused on human-host associated microbiomes from the HMP2 project (Peterson et al., 2009), where we found that over 70-80% of the microbial species in these samples were contained in the database. Before performing the classification and sub-sampling, we used Trimmomatic (Bolger et al., 2014) to remove errors in the read data for our short-read sequences. We did not pre-process the PromethION data.

Accelerating Containment Queries: The *containment* problem is to determine whether sequences from a particular organism are present in a dataset. Containment queries are crucial for exploratory analysis, but they can take up to an hour for large datasets, even with efficient techniques based on MinHash and Bloom filters (Koslicki & Zabeti, 2019). We measure the end-to-end speedup of Mash Screen (Ondov

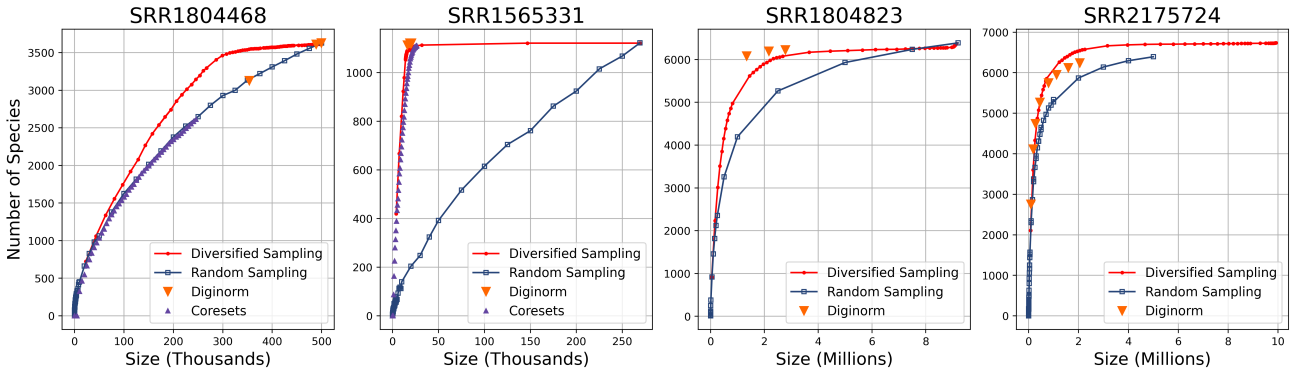

Figure 3. Sample size vs. species count for short-read Illumina datasets. Up and to the left is better. Note that we were unable to run coresets on million-read datasets. Also, we were unable to specify the full range of sample sizes with Diginorm due to algorithm limitations that prevent small outputs. We abbreviate random sampling (RS) and our diversified sampling algorithm (DS).

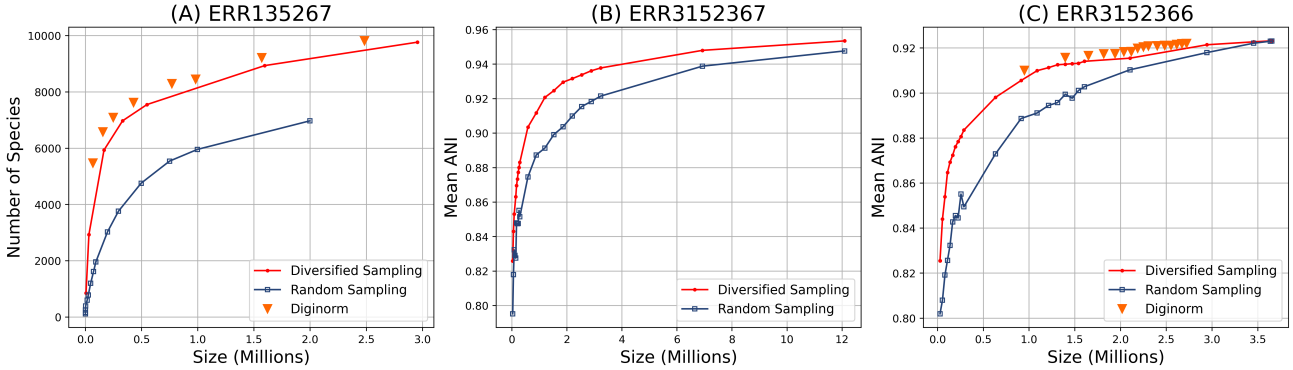

Figure 4. (A): Sample size vs. species count for long-read PromethION data. Higher is better. (B,C): ANI (roughly corresponds to similarity) between ground-truth contents of a mock metagenome and downsampled data on long-read (B) PromethION and (C) GridION datasets. Higher is better. Coresets required excessive resources ($>$ 1 day of computation), as did Diginorm on task (B).

et al., 2019) – a popular containment tool – on downsampled datasets. We use mock metagenome datasets for which the true containment status is known (Nicholls et al., 2019).

Protein Cluster Representatives: Gene ontology (GO) terms are labels that describe the function of proteins. To avoid processing the full reference archive of 300 million proteins (UniRef), practitioners often use a downsampled version that contains *representative* proteins (Suzek et al., 2015). We compare our sample against the official cluster-based samples released by the UniProt consortium (Suzek et al., 2007).

Algorithm Implementations: We implemented our downsampling algorithm in C++, and we compare against our own baseline C++ implementations of composable coresets and reservoir sampling. We used the Jaccard distance with and a window length of 100 sequences for the coreset implementation. We also compare against the official Diginorm release with default settings (Crusoe et al., 2015).

Hyperparameters: Our algorithm requires five hyperparameters: τ, R, B, k, and n. We use $R = 10$ RACE repetitions for all tasks. We use $B = 1$M except for on short-read datasets, where we use $B = 100$K to allow the sketch to fit into the L1 CPU cache. For the simple threshold-based rejection procedure, τ is roughly proportional to the number of samples in $\mathcal{S}$ and the choice of τ directly controls the sampling rate. k is the length of the k-mers used to vectorize the sequence. We use $k = 8$ for UniRef, $k = 18$ for short-read metagenomes and $k = 22$ for long-read metagenomes, following guidelines from (Ondov et al., 2016) and (Jain et al., 2018). n is the number of LSH concatenations (see Section 1); we simply use $n = 1$.

We sweep across the parameters that determine the sample rate for all methods. The official Diginorm package exposes a parameter C (coverage), which directly determines the downsampling rate, and the memory budget. Diginorm automatically detects when the memory is too small, so we set it to the lowest acceptable value. Coresets only set M,

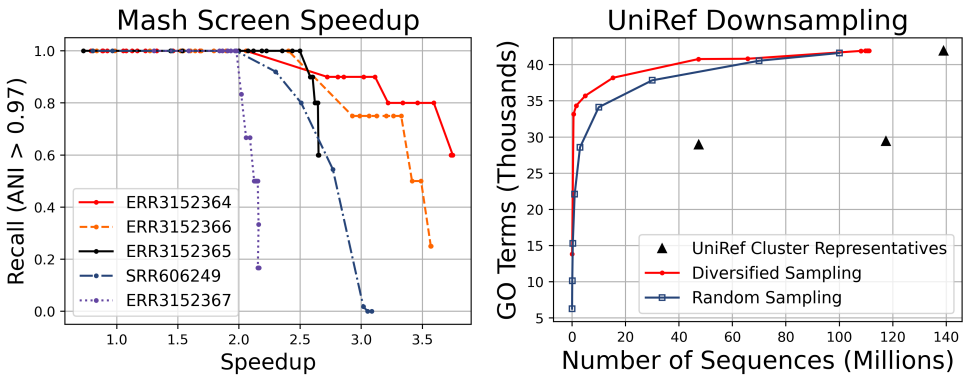

Figure 5. (A): End-to-end speedup vs the recall of all species identified by Mash Screen as having ANI $> 97\%$ (multiple datasets). Higher is better - 1.0 indicates that the subsample contains all relevant information. Timings include the downsampling time as well as the time to run Mash Screen. (B): Downsampling the UniRef archive while preserving GO terms. Higher is better.

Table 1. Comparison of computational resources on a large-scale PromethION dataset. We abbreviate random sampling (RS) and diversified sampling (DS).

Method	Throughput (Mbp/sec)	RAM	$99^{\text{th}}\%$ Latency
DS	12.11	4 MB	970 μs
RS	1012	1 GB	13.5 μs
Diginorm	4.65	3 GB	–
Coresets	0.012	1 GB	1.8 s

the number of samples. We sweep τ from 0.1 to 100, C from 1 to 20 and M from 1 to the number of reads.

Since τ is similar to a coverage parameter, it can be difficult to know what value will produce a given sample size M. This can be addressed in several ways. We can dynamically adjust τ if the fraction of recently accepted reads is too large / small. We can also allow τ to weakly depend on the number t of reads processed so far (i.e. $\tau = 0.01t$) or progressively increase τ with each rejection. In this work, we use fixed values and observe that the relationship between τ and M is roughly logarithmic ($\tau = 0.01$ implies aggressive 100x downsampling while $\tau = 100$ retains most samples).

5. Results

Species-Preserving Downsampling: Figure 3 and Figure 4 show the species diversity sampling performance of our method when applied to Illumina and PromethION datasets, respectively. We observe that diversified sampling selects sequences which represent a better-than-random fraction of the total number of species. While other algorithms such as coresets and Diginorm occasionally provide samples with similar or better diversity, they are substantially more expensive. Table 1 shows that our sketch needs 1000x less memory than coresets and is 2.6x faster than Diginorm. This corresponds to a processing time reduction from 8.8 hours (for Diginorm) to 3.3 hours on the PromethION dataset with 32 million reads. These trends are consistent for other datasets and tasks - our method is substantially cheaper than alternatives with similar diversity-preserving performance.

We also find that diversified sampling consistently preserves the metagenomic diversity for our two mock metagenomes. In this task, we downsample metagenomes with known contents and measure the sample quality by computing the mean ANI between the ground-truth contents and the downsampled sequences. Ideally, the ANI will be large (nearly 1) because the contents are known to be present in the dataset, but as the coverage may go down as the algorithm discards more sequences. Thus, the decrease of ANI is a measure of information loss. It should be noted that the metagenome datasets in Figures 4B and 4C have *logarithmically distributed* species counts. The only way to perform well in this setting is to ensure that all species – even the rare ones – are represented in the sample. Figure 4 clearly shows that the diversified sampling algorithm selectively retains sequences from rare organisms.

Containment Queries and Protein Clustering: Figure 5 shows our experiments with containment queries and the UniRef protein archive. Figure 5A shows the relationship between information loss and acceleration of Mash Screen. As the downsampling ratio becomes more aggressive, the size of the input seen by Mash Screen decreases, leading to a speedup. However, the sample also progressively loses information that is relevant to the containment query. We find that diversified sampling attains a 2x speedup without changing the output of Mash Screen (7 hours to 3 hours for large PromethION datasets). Our experiments with UniRef (Figure 5B) again show that diversified sampling preserves biologically-relevant information in a dataset. Using less than 8.5% of the protein sequences, we are able to represent

Table 2. Dataset information. We report the run accession, dataset size N, mean sequence length L, number of species / GO terms / classes in the dataset Z, and a description of the application. We use datasets with Illumina HiSeq 2000 short reads and with PromethION long reads.

Dataset	Type	N	L	Z	Description
SRR1565331	Illumina	269k	71.0	1,120	HMP2 sample of anterior nares
SRR1804823	Illumina	9.2M	98.6	6,387	HMP2 sample of supragingival plaque
SRR2175724	Illumina	10.2M	98.4	6,759	HMP2 sample of fecal matter
SRR1804468	Illumina	501k	99.4	3,630	HMP2 sample of buccal mucosa
ERR3152367	PromethION	32M	4,611	11,025	Mock metagenome data (Nicholls et al., 2019)
SRR606249	Illumina	54.8M	101	-	Spiked metagenome
UniRef	Protein	300M	293	41,910	Protein reference clusters

90% of the GO terms. It should be noted that UniRef clusters are intended to preserve sequence identity rather than gene ontology terms, but the fact remains that our samples outperform even the official subsets of UniRef.

6. Discussion

The most exciting feature of our algorithm is the practical speedup over existing algorithms to preserve diversity in a stream. In our experiments, diversified sampling processed 32 million PromethION reads at 12 Mbp/second using only 4 MB of RAM. This computational performance is 3x faster and 1000x more memory efficient than existing algorithms, and can be easily sustained on next-generation sequencing platforms such as the Oxford Nanopore MK1C, which has a 6-core ARM processor and 8 GB of RAM. To the best of our knowledge, no other adaptive downsampling methods can run in this computational environment. As sequencer data streams run for longer periods of time and generate more sequences, Diginorm and related methods will continue to require 10x memory upgrades. While 3 GB is completely reasonable, 300 GB pushes the boundaries of many commercial workstations. Our diversified sampling algorithm can operate on essentially unending streams with negligible memory increases. Since Algorithm 1 can operate with only a few megabytes of RAM, it can even be deployed on the onboard V100 GPUs with which many sequencers come equipped.

Our method also shows promise for database compression and data analysis. We were able to quickly compress the UniRef by more than 90%, while retaining the majority of gene ontology annotations. We also obtained a 2x end-to-end speedup for sequence containment queries without affecting the quality of the output. As a result, we expect our method to become useful to analysts who wish to perform a fast first-cut analysis.

Limitations: We believe that diversified sampling is useful in many contexts, but we only know that our method is effective for the applications we tested. Because inverse propensity sampling discards the original abundance information, there are some applications (e.g. differential expression analyses, relative abundance analyses, and metagenome assembly) where we explicitly do not recommend this preprocessing step. Even within supported applications, diversified sampling can inadvertently discard sequences that contain biologically relevant information. Therefore, we caution against the use of any sampling method in settings where false negatives could have disastrous consequences, such as pathogen detection. One practical issue with the algorithm is that it is difficult to determine how τ translates to the size of the sample. While this can be partially alleviated by placing a reservoir sampling buffer *after* our prefilter, this behavior has not yet been tested or implemented as part of our open source downsampling tool.

7. Conclusion

We have presented a fast, versatile and low-memory algorithm for diversified sampling. With advances in long-read sequencing hardware and the exponential growth of data archives, our method could play a crucial part in novel biological discoveries by facilitating real-time analyses on large datasets with clinical/public health relevance. Given its compatibility with some of the most ubiquitous tasks (alpha diversity analyses, microbial presence/absence) and databases (UniProt) in computational biology, our method is well positioned to be widely adopted by the large-scale comparative genomics research community.

References

Abbar, S., Amer-Yahia, S., Indyk, P., and Mahabadi, S. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 1–12. ACM, 2013a.

Abbar, S., Amer-Yahia, S., Indyk, P., Mahabadi, S., and Varadarajan, K. R. Diverse near neighbor problem. In *Proceedings of the 29th Annual Symposium on Computational Geometry*, pp. 207–214. ACM, 2013b.

Bolger, A. M., Lohse, M., and Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.

Broder, A. Z. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pp. 21–29, 1997a.

Broder, A. Z. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pp. 21–29. IEEE, 1997b.

Brown, C. T., Howe, A., Zhang, Q., Pyrkosz, A. B., and Brom, T. H. A reference-free algorithm for computational normalization of shotgun sequencing data. *arXiv preprint arXiv:1203.4802*, 2012.

Carter, J. L. and Wegman, M. N. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18: 143–154, 1979.

Coleman, B. and Shrivastava, A. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of the 2020 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2020.

Coleman, B., Baraniuk, R., and Shrivastava, A. Sub-linear memory sketches for near neighbor search on streaming data. In *International Conference on Machine Learning*, pp. 2089–2099. PMLR, 2020.

Cormode, G. and Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55:58–75, 2005.

Crusoe, M. R., Alameldin, H. F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., Charbonneau, A., Constantinides, B., Edvenson, G., Fay, S., et al. The khmer software package: enabling efficient nucleotide sequence analysis. *F1000Research*, 4, 2015.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pp. 253–262. ACM, 2004.

Edwards, H. S., Krishnakumar, R., Sinha, A., Bird, S. W., Patel, K. D., and Bartsch, M. S. Real-time selective sequencing with rubric: Read until with basecall and reference-informed criteria. *Scientific reports*, 9(1):1–11, 2019.

Gordienko, E. N., Kazanov, M. D., and Gelfand, M. S. Evolution of pan-genomes of escherichia coli, shigella spp., and salmonella enterica. *Journal of bacteriology*, 195(12):2786–2792, 2013.

Indyk, P. and Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

Indyk, P., Mahabadi, S., Mahdian, M., and Mirrokni, V. S. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 100–108. ACM, 2014.

Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T., and Aluru, S. High throughput ani analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nature Communications*, 9(1):5114, 2018.

Kavvas, E. S., Catoiu, E., Mih, N., Yurkovich, J. T., Seif, Y., Dillon, N., Heckmann, D., Anand, A., Yang, L., Nizet, V., et al. Machine learning and structural analysis of mycobacterium tuberculosis pan-genome identifies genetic signatures of antibiotic resistance. *Nature communications*, 9(1):1–9, 2018.

Köser, C. U., Ellington, M. J., and Peacock, S. J. Whole-genome sequencing to control antimicrobial resistance. *Trends in Genetics*, 30(9):401–407, 2014.

Koslicki, D. and Zabeti, H. Improving minhash via the containment index with applications to metagenomic analysis. *Applied Mathematics and Computation*, 354:206–215, 2019.

Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y., Cleland, I., Faruque, N., Goodgame, N., Gibson, R., et al. The european nucleotide archive. *Nucleic acids research*, 39(suppl_1):D28–D31, 2010a.

Leinonen, R., Sugawara, H., Shumway, M., and Collaboration, I. N. S. D. The sequence read archive. *Nucleic acids research*, 39(suppl_1):D19–D21, 2010b.

Loose, M., Malla, S., and Stout, M. Real-time selective sequencing using nanopore technology. *Nature methods*, 13(9):751, 2016.

Luo, C. and Shrivastava, A. Arrays of (locality-sensitive) count estimators (ACE): Anomaly detection on the edge. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1439–1448. International World Wide Web Conferences Steering Committee, 2018.

Manku, G. S. and Motwani, R. Approximate frequency counts over data streams. In *Proceedings of 28th International Conference on Very Large Data Bases*, pp. 346–357, 2002.

McCauley, S. Approximate similarity search under edit distance using locality-sensitive hashing. *arXiv preprint arXiv:1907.01600*, 2019.

McCorrison, J. M., Venepally, P., Singh, I., Fouts, D. E., Lasken, R. S., and Methé, B. A. Neatfreq: reference-free data reduction and coverage normalization for de novosequence assembly. *BMC bioinformatics*, 15(1):1–12, 2014.

Meyerson, M., Gabriel, S., and Getz, G. Advances in understanding cancer genomes through second-generation sequencing. *Nature Reviews Genetics*, 11(10):685, 2010.

Nasko, D. J., Koren, S., Phillippy, A. M., and Treangen, T. J. RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. *Genome Biology*, 19(1):165, 2018.

Nicholls, S. M., Quick, J. C., Tang, S., and Loman, N. J. Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *Gigascience*, 8(5):giz043, 2019.

Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., and Phillippy, A. M. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

Ondov, B. D., Starrett, G. J., Sappington, A., Kostic, A., Koren, S., Buck, C. B., and Phillippy, A. M. Mash screen: high-throughput sequence containment estimation for genome discovery. *Genome biology*, 20(1):1–13, 2019.

Peterson, J., Garges, S., Giovanni, M., McInnes, P., Wang, L., Schloss, J. A., Bonazzi, V., McEwen, J. E., Wetterstrand, K. A., Deal, C., et al. The NIH human microbiome project. *Genome Research*, 19(12):2317–2323, 2009.

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., et al. A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464 (7285):59–65, 2010.

Rosenbaum, P. R. and Rubin, D. B. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Rowe, W. P. When the levee breaks: a practical guide to sketching algorithms for processing the flood of genomic data. *Genome Biology*, 20(199):1–12, 2019.

Sankar, P. L. and Parker, L. S. The precision medicine initiative's all of us research program: an agenda for research on its ethical, legal, and social issues. *Genetics in Medicine*, 19(7):743–750, 2017.

Schatz, M. C. and Langmead, B. The dna data deluge: fast, efficient genome sequencing machines are spewing out more data than geneticists can analyze. *Ieee Spectrum*, 50(7):26, 2013.

Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, 9(8):811–814, 2012.

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., Iyer, R., Schatz, M. C., Sinha, S., and Robinson, G. E. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.

Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., and Wu, C. H. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H., and Consortium, U. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Timme, R. E., Leon, M. S., and Allard, M. W. Utilizing the public genometrakr database for foodborne pathogen traceback. In *Foodborne Bacterial Pathogens*, pp. 201–212. Springer, 2019.

Vitter, J. S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

Wedemeyer, A., Kliemann, L., Srivastav, A., Schielke, C., Reusch, T. B., and Rosenstiel, P. An improved filtering algorithm for big read datasets and its application to single-cell assembly. *BMC bioinformatics*, 18(1):1–11, 2017.

Wick, R. R., Judd, L. M., and Holt, K. E. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome biology*, 20(1):129, 2019.

Wood, D. E., Lu, J., and Langmead, B. Improved metagenomic analysis with kraken 2. *Genome biology*, 20(1): 1–13, 2019.

A. Running the Code

The diversified sampling algorithm is available as a web app that runs in the browser and as an open-source command line tool. To avoid violating double-blind review, we have included the repository as a zip file and deployed the web app to the (anonymous) URL: `Lc28kXQtqH.github.io`.

B. Experimental Procedures and Datasets

Computational Hardware

	Server 1	Server 2
CPU	56 cores Intel Xeon E5-2660 v4 @ 2.00GHz	64 cores Intel Xeon Gold 5218 @ 2.30GHz
GPU	ASPEED Graphics Family (rev 30)	ASPEED Graphics Family (rev 41)
Memory (GB)	528.27	394.86

Task 1: Species-Preserving Downsampling

We procured short-read sequence datasets from Illumina sequencers and long-read sequence datasets from PromethION sequencers. These datasets are summarized in table 2. We first annotate each dataset by running it through Kraken2 `https://github.com/DerrickWood/kraken2`, an open source software that maps sequences to their Taxon IDs. Concretely, the software outputs a copy of the dataset where each sequence's Taxon ID is appended to its description line; the sequences themselves remain unchanged. We then downsample these annotated files with RACE along with Diginorm, Coresets and random sampling as benchmarks. The output of these algorithms is a downsampled version of the annotated file; the file format is preserved and each retained entry is identical to the original entry. Each retained sequence has a corresponding description line with its Taxon ID. Finally, we then run a Python script to count the number of unique Taxon IDs in the subsamples.

Task 2: Accelerating Containment Queries

Given two sequences A and B which are represented as sets of k-mers, the task is to estimate the containment score $\frac{|A \cap B|}{|A|}$ (Koslicki & Zabeti, 2019). We call A the *reference* and B the *query*. Note that the containment score is close to 1 only if most of the elements in A are present in B. Containment queries are important as a first-cut analysis. To understand why, consider the situation where A is a genome and B is a sequencing run or a metagenome. If the species with genome A is present in the sample that produced B, then the containment score will be large. The converse statement is not true – sequences with a large containment score are not necessary present. Even still, containment is still a very useful way to pre-screen a large dataset to identify species candidates that *might* be present (Ondov et al., 2019).

The aptly-named *Mash Screen* tool attempts to provide just that - the ability to quickly identify a small set of genomes that may be present in a dataset. Mash screen is widely used, but we observe that the tool can take a long time to run on large datasets. Our goal for this task is to reduce the size of the input to Mash screen (i.e. downsample the query B) while retaining the ability to compute accurate containment scores.

We use mock metagenome datasets for which the true containment status is known (Nicholls et al., 2019). A mock metagenome is the result of running a sequencing experiment on a sample with a known, ground-truth metagenome community profile. Mash Screen returns an *identity* score, which estimates the number of bases that are shared between the query and the reference. The identity score is estimated from the containment score, and it is common practice to create a set of candidate species by cutting off the values at an appropriate score (i.e. identity > 0.97). Therefore, we consider two measures of quality - the average identity score against the reference genomes of ground-truth species, and the recall of ground-truth species when using a cutoff of 0.97.

Task 3: Protein Cluster Representatives

UniProt provides a UniRef id-mapping file, which provides a mapping from uniref100 id's to the corresponding uniref90 and uniref50 cluster representative ids as well as their gene ontology terms. We ran a python script that takes in the uniref100 fasta file and the idmapping file, creates a hashmap of uniref100id to GO terms, then iterates through the uniref100 file,

appending the GO terms to the description line, and writes out the sequence with the new description line to the output file. Like in the species alpha diversity experiment, this newly annotated file is then downsampled with RACE along with random sampling as a benchmark. The downsampled files are then run through another Python script that reads description lines and counts the number of unique gene ontology terms.

We also benchmarked against the number of unique GO terms found in uniref50 and uniref90, which contain representatives of protein sequences clustered at 90% and 50% sequence identity respectively. To count this, we again used the UniRef id-mapping. Recall that it maps uniref100 id's to the corresponding cluster representative id's in uniref50 and uniref90, in addition to the gene ontology terms. To count the number of unique GO terms in uniref50, we wrote a python script that iterates through every entry in id-mapping and only keeps gene ontology terms of an entry only if the uniref100 id matches the uniref50 cluster representative id, effectively filtering out the entries to just the uniref50 cluster representatives. We then simply count the number of unique go terms among the kept GO terms. The same goes for counting number of unique uniref90 GO terms, except now we keep entries where uniref100id matches the uniref90 id.

C. Theoretical Results

In this section, we provide proofs for the theorems.

Lemma 3.3. *Suppose that two classes C_1 and C_2 are Δ-separable and that there exists an $(R, \Delta, \alpha, \beta)$-sensitive hash family $\mathcal{H}$. Then given $\delta \in [0, 1]$, there exists an LSH function $h(x)$ such that $h(x) \neq h(y)$ for all $x \in C_1$ and $y \in C_2$ with probability $1 - \delta$.*

Proof. Construct a hash function $h(x)$ by concatenating n hash functions sampled from $\mathcal{H}$ and note that $h(x)$ is $(R, \Delta, \alpha^n, \beta^n)$-sensitive. We want a lower bound on the probability that there are no collisions between $x \in C_1$ and $y \in C_2$.

$$\Pr_{\substack{\forall x \in C_1, \\ y \in C_2}}[h(x) \neq h(y)] = 1 - \Pr\Big[\bigcup_{\substack{x \in C_1 \\ y \in C_2}} A_{x,y}\Big] \geq 1 - \delta$$

$$\Rightarrow \Pr\Big[\bigcup_{\substack{x \in C_1 \\ y \in C_2}} A_{x,y}\Big] \leq \delta$$

where $A_{x,y}$ is the event that $h(x) = h(y)$. Using the union bound and the fact that C_1 and C_2 are Δ-separable, we have

$$\Pr\Big[\bigcup_{\substack{x \in C_1 \\ y \in C_2}} A_{x,y}\Big] \leq \sum_{x \in C_1} \sum_{y \in C_2} \Pr[A_{x,y}]$$

$$= \sum_{x \in C_1} \sum_{y \in C_2} \rho(x, y)^n \leq |C_1||C_2|\beta^n$$

because $d(x, y) \geq \Delta$ for all $x \in C_1$ and $y \in C_2$. To get the result, set

$$n \geq \frac{\log \frac{\delta}{|C_1||C_2|}}{\log \beta}$$

□

Now we consider the discrete set of count values in A. Since the count values are non-decreasing, it is straightforward to see that Algorithm 1 causes them to converge to $\lceil \tau \rceil$.

Lemma 3.4. *Construct a single RACE array with B buckets using an LSH function $h(\cdot)$ on the elements of the sample set $\mathcal{S}$ obtained using Algorithm 1. Then Algorithm 1 causes each of the RACE counters in this array to converge to $\lceil \tau \rceil$.*

Proof. Let $f_{\mathcal{D}}$ be the underlying probability distribution that generates each element of the data stream $\mathcal{D}$. Restrict the domain of $h(x)$ to $x \in \text{supp}(f_{\mathcal{D}})$ and assume without loss of generality that the restricted range of $h(x)$ is the set of integers $\{1, \ldots, B\}$.

Consider one of the counters in the RACE array. As a sequence of elements $x_1, x_2, ...$ arrive from the stream, there will be a sequence of count values $n_1, n_2, ...$ within this counter. Note that there are B such sequences - one for each counter in the array that lies in the domain of $h(x)$ when $x \sim f_{\mathcal{D}}$.

Note that the sequence of counters $n_1, n_2, ...$ is non-decreasing and bounded above by $\lceil \tau \rceil$. Also note that at every time step, there is a nonzero probability that the counter will be incremented (if it is less than $\lceil \tau \rceil$). Therefore, the sequence of count values converges to $\lceil \tau \rceil$. This is true for all the count values, by the same argument. □

Diversity Indices: We measure sample diversity using the Shannon index (1), inverse Simpson index (2), and Berger–Parker index (3).

$$H' = -\sum_{i=1}^{Z} z_i \log\ z_i \qquad \lambda^{-1} = \left(\sum_{i=1}^{Z} z_i^2 \right)^{-1} \qquad D_B = \max_{1 \le i \le Z} z_i$$

Z is the number of sequence classes in the sample $\mathcal{S}$. The proportion z_i is the ratio n_i/M, where n_i is the number of times that class i appears in the sample and M is the sample size. Assume the classes are Δ-separable and use Lemma 3.3 to ensure that each class maps to a unique set of buckets in the RACE array. We are now ready to express the diversity index in terms of the RACE counts.

Theorem 3.5. *Assume that classes $C_1, \ldots, C_S$ are Δ-separable and use the hash function from Lemma 3.3 to construct a count sketch for the output of Algorithm 1. Then the ratio z_i converges to a constant:*

$$z_i = \frac{n_i}{m} \to \frac{\lceil \tau \rceil B_i}{\lceil \tau \rceil B} = \frac{B_i}{B}$$

B_i is the number of buckets in the hash range of class C_i.

Proof. This is a simple consequence of the previous lemmas. Let n_i be the number of species from class C_i in the sample. Because the buckets for the Z classes do not overlap, n_i is equal to the sum of count values in the B_i buckets that correspond to class C_i. By the previous lemma, the count value for each of the B_i different buckets in the domain of C_i converge to $\lceil \tau \rceil$, so we have that $n_i \to \lceil \tau \rceil B_i$.

Now consider the value of m, the total number of samples kept by the downsampling algorithm. Because the buckets do not overlap, the number of samples kept by the algorithm is equal to the sum of all the counters (since each sample that is retained causes an increment of 1 in the array). There are B buckets, each having (in the limit) $\lceil \tau \rceil$ counts. Therefore $m \to \lceil \tau \rceil B$, where $B = \sum_i B_i$. □

This theorem allows us to prove useful statements about the diversity index. Consider the simple case where all B_i are equal. The following statement is a straightforward consequence of the fact that $z_i \to 1/Z$ and the observation that the diversity indices attain their maximum when $z_i = 1/Z$.

Corollary 3.6. *Suppose each class maps to exactly k buckets (i.e. $B_i = k$). With $B = Zk$ buckets, the Shannon, Simpson, and Berger-Parker diversity measures converge to their maximum value.*

Proof. Under these conditions, $z_i \to \frac{1}{Z}$ according to the previous theorem. Observe that the optimal value of the diversity indices

The Shannon index is equal to the entropy of the empirical distribution across the Z classes, which is maximized when all classes have the same weight $\frac{1}{Z}$. The optimal value of the Simpson index can be found by maximizing the sum $\sum_i z_i^2$ subject to the simplex constraint $\sum_i z_i = 1$.

A large Berger-Parker index indicates poor diversity, where a single class dominates the profile (i.e. the index is maximized when a single species is present For this reason, we wish to maximize the *inverse* Berger-Parker index, or equivalently to *minimize* the Berger-Parker index. It is easy to see that $z_i = 1/Z$ provides the optimal index value. □

Acknowledgements

B.C, B.G, L.C and A.S were supported by National Science Foundation IIS-1652131, BIGDATA-1838177, RI-1718478, AFOSR- YIP FA9550-18-1-0152, Amazon Research Award, and the ONR BRC grant on Randomized Numerical Linear Algebra. T.T. was supported by by National Science Foundation grant EF-2126387 and National Institute of Allergy and Infectious Diseases grant P01AI152999. L.E. was supported by NLM Training Program (T15LM007093).

References

Abbar, S., Amer-Yahia, S., Indyk, P., and Mahabadi, S. Real-time recommendation of diverse related articles. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 1–12. ACM, 2013a.

Abbar, S., Amer-Yahia, S., Indyk, P., Mahabadi, S., and Varadarajan, K. R. Diverse near neighbor problem. In *Proceedings of the 29th Annual Symposium on Computational Geometry*, pp. 207–214. ACM, 2013b.

Bolger, A. M., Lohse, M., and Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, 30 (15):2114–2120, 2014.

Broder, A. Z. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pp. 21–29, 1997a.

Broder, A. Z. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pp. 21–29. IEEE, 1997b.

Brown, C. T., Howe, A., Zhang, Q., Pyrkosz, A. B., and Brom, T. H. A reference-free algorithm for computational normalization of shotgun sequencing data. *arXiv preprint arXiv:1203.4802*, 2012.

Carter, J. L. and Wegman, M. N. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18: 143–154, 1979.

Coleman, B. and Shrivastava, A. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of the 2020 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2020.

Coleman, B., Baraniuk, R., and Shrivastava, A. Sub-linear memory sketches for near neighbor search on streaming data. In *International Conference on Machine Learning*, pp. 2089–2099. PMLR, 2020.

Cormode, G. and Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55:58–75, 2005.

Crusoe, M. R., Alameldin, H. F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., Charbonneau, A., Constantinides, B., Edvenson, G., Fay, S., et al. The khmer software package: enabling efficient nucleotide sequence analysis. *F1000Research*, 4, 2015.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, pp. 253–262. ACM, 2004.

Edwards, H. S., Krishnakumar, R., Sinha, A., Bird, S. W., Patel, K. D., and Bartsch, M. S. Real-time selective sequencing with rubric: Read until with basecall and reference-informed criteria. *Scientific reports*, 9(1):1–11, 2019.

Gordienko, E. N., Kazanov, M. D., and Gelfand, M. S. Evolution of pan-genomes of escherichia coli, shigella spp., and salmonella enterica. *Journal of bacteriology*, 195(12):2786–2792, 2013.

Indyk, P. and Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

Indyk, P., Mahabadi, S., Mahdian, M., and Mirrokni, V. S. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 100–108. ACM, 2014.

Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T., and Aluru, S. High throughput ani analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nature Communications*, 9(1):5114, 2018.

Kavvas, E. S., Catoiu, E., Mih, N., Yurkovich, J. T., Seif, Y., Dillon, N., Heckmann, D., Anand, A., Yang, L., Nizet, V., et al. Machine learning and structural analysis of mycobacterium tuberculosis pan-genome identifies genetic signatures of antibiotic resistance. *Nature communications*, 9(1):1–9, 2018.

Köser, C. U., Ellington, M. J., and Peacock, S. J. Whole-genome sequencing to control antimicrobial resistance. *Trends in Genetics*, 30(9):401–407, 2014.

Koslicki, D. and Zabeti, H. Improving minhash via the containment index with applications to metagenomic analysis. *Applied Mathematics and Computation*, 354:206–215, 2019.

Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y., Cleland, I., Faruque, N., Goodgame, N., Gibson, R., et al. The european nucleotide archive. *Nucleic acids research*, 39(suppl_1):D28–D31, 2010a.

Leinonen, R., Sugawara, H., Shumway, M., and Collaboration, I. N. S. D. The sequence read archive. *Nucleic acids research*, 39(suppl_1):D19–D21, 2010b.

Loose, M., Malla, S., and Stout, M. Real-time selective sequencing using nanopore technology. *Nature methods*, 13(9):751, 2016.

Luo, C. and Shrivastava, A. Arrays of (locality-sensitive) count estimators (ACE): Anomaly detection on the edge. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1439–1448. International World Wide Web Conferences Steering Committee, 2018.

Manku, G. S. and Motwani, R. Approximate frequency counts over data streams. In *Proceedings of 28th International Conference on Very Large Data Bases*, pp. 346–357, 2002.

McCauley, S. Approximate similarity search under edit distance using locality-sensitive hashing. *arXiv preprint arXiv:1907.01600*, 2019.

McCorrison, J. M., Venepally, P., Singh, I., Fouts, D. E., Lasken, R. S., and Methé, B. A. Neatfreq: reference-free data reduction and coverage normalization for de novosequence assembly. *BMC bioinformatics*, 15(1):1–12, 2014.

Meyerson, M., Gabriel, S., and Getz, G. Advances in understanding cancer genomes through second-generation sequencing. *Nature Reviews Genetics*, 11(10):685, 2010.

Nasko, D. J., Koren, S., Phillippy, A. M., and Treangen, T. J. RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. *Genome Biology*, 19(1):165, 2018.

Nicholls, S. M., Quick, J. C., Tang, S., and Loman, N. J. Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *Gigascience*, 8(5):giz043, 2019.

Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., and Phillippy, A. M. Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132, 2016.

Ondov, B. D., Starrett, G. J., Sappington, A., Kostic, A., Koren, S., Buck, C. B., and Phillippy, A. M. Mash screen: high-throughput sequence containment estimation for genome discovery. *Genome biology*, 20(1):1–13, 2019.

Peterson, J., Garges, S., Giovanni, M., McInnes, P., Wang, L., Schloss, J. A., Bonazzi, V., McEwen, J. E., Wetterstrand, K. A., Deal, C., et al. The NIH human microbiome project. *Genome Research*, 19(12):2317–2323, 2009.

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., et al. A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464(7285):59–65, 2010.

Rosenbaum, P. R. and Rubin, D. B. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Rowe, W. P. When the levee breaks: a practical guide to sketching algorithms for processing the flood of genomic data. *Genome Biology*, 20(199):1–12, 2019.

Sankar, P. L. and Parker, L. S. The precision medicine initiative's all of us research program: an agenda for research on its ethical, legal, and social issues. *Genetics in Medicine*, 19(7):743–750, 2017.

Schatz, M. C. and Langmead, B. The dna data deluge: fast, efficient genome sequencing machines are spewing out more data than geneticists can analyze. *Ieee Spectrum*, 50(7):26, 2013.

Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, 9(8):811–814, 2012.

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., Iyer, R., Schatz, M. C., Sinha, S., and Robinson, G. E. Big data: astronomical or genomical? *PLoS biology*, 13(7):e1002195, 2015.

Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., and Wu, C. H. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H., and Consortium, U. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Timme, R. E., Leon, M. S., and Allard, M. W. Utilizing the public genometrakr database for foodborne pathogen traceback. In *Foodborne Bacterial Pathogens*, pp. 201–212. Springer, 2019.

Vitter, J. S. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, 1985.

Wedemeyer, A., Kliemann, L., Srivastav, A., Schielke, C., Reusch, T. B., and Rosenstiel, P. An improved filtering algorithm for big read datasets and its application to single-cell assembly. *BMC bioinformatics*, 18(1):1–11, 2017.

Wick, R. R., Judd, L. M., and Holt, K. E. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome biology*, 20(1):129, 2019.

Wood, D. E., Lu, J., and Langmead, B. Improved metagenomic analysis with kraken 2. *Genome biology*, 20(1):1–13, 2019.