Bioinformatics, 37, 2021, i334–i341
doi: 10.1093/bioinformatics/btab315
ISMB/ECCB 2021

Practical selection of representative sets of RNA-seq samples using a hierarchical approach

Laura H. Tung and Carl Kingsford*

Computational Biology Department, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

*To whom correspondence should be addressed.

Abstract

Motivation: Despite numerous RNA-seq samples available at large databases, most RNA-seq analysis tools are evaluated on a limited number of RNA-seq samples. This drives a need for methods to select a representative subset from all available RNA-seq samples to facilitate comprehensive, unbiased evaluation of bioinformatics tools. In sequence-based approaches for representative set selection (e.g. a k-mer counting approach that selects a subset based on k-mer similarities between RNA-seq samples), because of the large numbers of available RNA-seq samples and of k-mers/sequences in each sample, computing the full similarity matrix using k-mers/sequences for the entire set of RNA-seq samples in a large database (e.g. the SRA) has memory and runtime challenges; this makes direct representative set selection infeasible with limited computing resources.

Results: We developed a novel computational method called 'hierarchical representative set selection' to handle this challenge. Hierarchical representative set selection is a divide-and-conquer-like algorithm that breaks representative set selection into sub-selections and hierarchically selects representative samples through multiple levels. We demonstrate that hierarchical representative set selection can achieve summarization quality close to that of direct representative set selection, while largely reducing runtime and memory requirements of computing the full similarity matrix (up to 8.4× runtime reduction and 5.35× memory reduction for 10 000 and 12 000 samples respectively that could be practically run with direct subset selection). We show that hierarchical representative set selection substantially outperforms random sampling on the entire SRA set of RNA-seq samples, making it a practical solution to representative set selection on large databases like the SRA.

Availability and implementation: The code is available at https://github.com/Kingsford-Group/hierrepsetselection and https://github.com/Kingsford-Group/jellyfishsim.

Contact: carlk@cs.cmu.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A vast number of RNA-seq short-read samples are publicly available at large sequence databases [e.g. NIH's Sequence Read Archive (Leinonen *et al.*, 2011), known as SRA]. However, most bioinformatics tools for RNA-seq analyses are evaluated on a limited number of samples; this evaluation may be insufficient, as the tools may not be adequately evaluated by samples with a variety of cell/tissue types and disease conditions. To ensure general applicability, an RNA-seq analysis tool should be validated on varying cell/tissue types and experiments. On the other hand, using all available RNA-seq samples to evaluate RNA-seq analysis tools is infeasible, and many samples in databases are similar to each other. This leads to a need to select a representative subset from available RNA-seq samples that effectively summarizes a large collection of RNA-seq samples to capture various essential transcriptional phenomena. Moreover, bioinformatics tools have algorithm parameters to be optimized, and automatic learning of optimal parameters can be made more robust using representative samples. Thus, our objective is to develop a computational method of selecting a representative subset from a large collection of RNA-seq short-read samples for a given organism (e.g. human), such that RNA-seq analysis tools can be effectively evaluated on this subset. Bioinformatics tools such as transcript assemblers, read mappers and expression abundance estimators would benefit from a good selection of RNA-seq samples in their evaluation and parameter optimization.

Various representative set selection methods that solve the problem of finding a subset of data points (representatives) to efficiently describe the original collection of data have been developed in the fields of computer vision, signal/image processing, information retrieval and machine learning. Most common applications of representative set selection include image, video and text summarizations. Machine learning tasks such as classification and regression can also improve in terms of fast training and reduced memory usage by using a representative subset as the training set (Elhamifar *et al.*, 2012a; Garcia et al., 2012). One category of representative set selection methods is clustering-based algorithms (Daszykowski et al., 2002; Frey and Dueck, 2007), using k-means, k-medoids, spectral clustering or DBSCAN. Another category is sparse modeling-based algorithms (Elhamifar et al., 2012b; Wang et al., 2014), which formulate the representative set selection as a dictionary learning problem, based on the assumption that the entire set can be reconstructed by linear combinations of dictionary items. There are also Rank Revealing QR algorithms (Boutsidis *et al.*, 2009) that use matrix factorization to find a subset of columns of the data matrix corresponding to the best conditioned submatrix, and algorithms using mutual information and relative entropy to search for a representative subset (Pan et al., 2005).

In the field of RNA-seq analysis, Hie et al. developed a geometric sketching algorithm (Hie et al., 2019) for single-cell RNA-seq, which summarizes the transcriptomic heterogeneity within a dataset using a representative subset of cells to accelerate single-cell analysis. Using a covering algorithm that approximates the original data space as a union of equal-sized boxes, geometric sketching focuses on even coverage of the transcriptional space spanned by the original set, so that rare cell types can be sufficiently sampled and represented. Maintaining a similar density distribution to that of the original set is useful for video/photo summarizations. However, for RNA-seq analyses, even coverage of the transcriptional space is more important in order to represent rare cell types.

A Python package, apricot (Schreiber et al., 2020), has been developed for selecting representative subsets using submodular optimization. Based on the 'diminishing returns' property, apricot maximizes a monotone submodular function's value to find a representative subset. Using facility location functions, apricot maximizes the sum of similarities between each sample and its closest representative sample; as a result, the representative set selected by apricot approximately evenly spans the space of the original data, like geometric sketching. While geometric sketching requires knowing samples' gene expression vectors, apricot can work with the similarity matrix between the samples directly.

A main challenge in representative set selection for RNA-seq samples is that the number of RNA-seq samples in large databases is huge and each RNA-seq sample takes up substantial disk space; therefore, it is impractical to download all RNA-seq sequences of all the samples available at a large database like the SRA due to limited disk space. To perform representative set selection directly, we need to obtain gene expression vectors of, or distances between, all available SRA samples; however, SRA streaming is also not feasible due to issues with paired-end reads.

Given this challenge, one might attempt to select a representative set without looking at the sequences of each RNA-seq sample and relying instead on each sample's metadata, for example, using NCBI's BioSample attributes (Barrett *et al.*, 2012) that affect gene expression levels to predict gene expression distances between RNA-seq samples. However, for most large RNA-seq collections, including the SRA, the metadata is highly incomplete and most samples do not have the needed metadata values for predicting their gene expression distances.

Thus, we use a sequence-based approach for representative set selection of RNA-seq samples. We randomly sample a small subset of reads from each RNA-seq sample to download, such that the subsets of reads from all available RNA-seq samples at the SRA take a reasonable disk space. We count k-mers in the subset of reads of each sample and compute the similarity between k-mer distributions of samples. This approach selects a representative set based on k-mer similarities and thus sequence similarities among RNA-seq samples. Since the number of publicly available RNA-seq samples in the SRA is large ($N = 196\ 523$ for human) and the number of k-mers in each sample is large (~2 000 000 k-mers), computing the 196 523 × 196 523 similarity matrix with k-mers has memory and runtime challenges even using a chunking method for matrix computation (Li et al., 2010).

To tackle this challenge, we developed a novel method called 'hierarchical representative set selection.' The hierarchical representative set selection is a divide-and-conquer-like algorithm that hierarchically selects representative samples through multiple levels. At each level, samples are divided into smaller chunks, and representative set selection is performed on each chunk with a weighting scheme. The representative samples selected from every chunk are merged into the next level, and the process repeats until the size of the similarity matrix of the merged samples is feasible for the computing resources.

Our results show that hierarchical representative set selection achieves summarization quality close to that of direct representative set selection using apricot (5.37% average difference in the measure of how well a selected subset represents the full set), while substantially reducing runtime and memory requirements of computing the full similarity matrix (up to 8.4× runtime reduction and 5.35× memory reduction for 10 000 and 12 000 samples respectively that could be practically run with direct subset selection), thus making selecting representative samples from the entire SRA RNA-seq samples feasible (the estimated runtime reduction is 90× and memory reduction is 41.4× for the SRA full set of 196 523 human samples). We demonstrate that the representative subset selected by our hierarchical representative set selection method from all human RNA-seq samples in the SRA better represents the transcriptomic heterogeneity among those samples than that by random sampling, and thus can be used for more comprehensive and complete evaluation of bioinformatics tools.

2 Materials and methods

2.1 Problem formulation

Let set R be a large set of RNA-seq samples (such as all the RNA-seq samples in the NIH SRA database for a given organism), let $d(i,j)$ be a distance or dissimilarity measure between samples i and j in R, and let $d(x,\ S)$ be the distance between a data point x and its closest data point in a set S. A reasonable formulation is to find a representative subset $\tilde{R} \subseteq R$, such that

$$\max_{r \in R} d(r, \tilde{R})$$

is as small as possible.

This is equivalent to minimizing the classical Hausdorff distance which is defined as: $d_H(X,S) = \max_{x \in X}\{\min_{s \in S} d(x,s)\}$ where X is the full set and S is its representative subset (Hie *et al.*, 2019). The Hausdorff distance can be used to evaluate how well a selected subset represents the original full set (a smaller value is better) (Hie *et al.*, 2019). However, the classical Hausdorff distance is highly sensitive to extreme outliers (Huttenlocher et al., 1993; Sim et al., 1999). Thus, in practice, a more robust measure, the partial Hausdorff distance, is used to evaluate the representative subset (Hie *et al.*, 2019; Huttenlocher *et al.*, 1993). The partial Hausdorff distance is defined as: $d_{HK}(X,S) = K^{th}_{x \in X}\{\min_{s \in S} d(x,s)\}$ where $K^{th}_{x \in X}$ is the Kth largest value (counting from the minimum), and a parameter $q = 1 - K/|X|$ is used to determine K (when $q = 0$, $d_{HK} = d_H$; when q is small enough, d_{HK} is very close to d_H but is robust to extreme outliers) (Hie *et al.*, 2019; Huttenlocher *et al.*, 1993). d_{HK} is at the $((1-q) \times 100)$-th percentile of the distances from every sample x in the full set to its closest representative sample.

2.2 K-mer similarity-based approach

The similarity between k-mer distributions of RNA-seq samples reflects the similarity between their sequences and is a reasonable approach for computing $d(i,\ j)$. Thus, by counting the k-mers of RNA-seq reads, we can select a representative set based on the k-mer similarities and therefore sequence similarities among samples. Downloading all reads of all SRA samples is infeasible, so we download a small subset of reads from each RNA-seq sample. To represent a full RNA-seq sample, the sampled small subset of reads are random with respect to the genome coordinates. For approximately 88% of SRA RNA-seq samples, the reads are stored from the sequencer without alignment. Reads coming from Illumina sequencers without alignment are random with respect to the genome coordinates. Thus, a range of reads downloaded from unaligned samples by *fastq-dump* are random. We download 10 000 reads to represent each unaligned RNA-seq sample (we skip the first 5000 reads in the sample when downloading, since the beginning of sequencing may contain some technical variation of signal introduced in the sequencing process).

Choosing a proper k-mer size is important, as smaller k-mers give less information about sequence similarities, while larger k-mers may result in fewer matches due to sequencing errors. To select an optimal k-mer size, we plot the number of distinct k-mers with the varying k-mer size for a range of typical read lengths of Illumina (Supplementary Figs S1–S5). In the linearly increasing part of the curve, short k-mers match randomly; from the beginning of the horizontal part of the curve, k-mers start to reveal the genome structure. Thus, we want the smallest k-mer size in the horizontal part of the curve, so that the k-mer matching moves from being random to being representative of the read content and is still resilient to sequencing errors. Among the optimal k-mer sizes we obtained for long, medium and short read-lengths, we choose a compromise 17 as the optimal k-mer size.

Jellyfish (Marçais and Kingsford, 2011) was used to count k-mers in the subset of reads. We use canonical k-mers (i.e. the lexicographically smaller of a k-mer and its reverse complement), so all samples are compared based on the common k-mer sequences regardless of the sequenced strand. We use the cosine similarity as the similarity of k-mer distributions between samples. Cosine similarity is the cosine of the angle between two vectors and is commonly used in document clustering and information retrieval; our k-mer vectors have some similar aspects to Term Frequency–Inverse Document Frequency vectors (e.g. large vocabulary, word counts, high dimension and high sparsity). Cosine similarity is also a good measure for k-mer-based metagenome comparisons (Choi *et al.*, 2019).

2.3 Hierarchical representative set selection algorithm

To handle the memory and runtime challenges, hierarchical representative set selection (as shown in Algorithm 1) breaks the whole representative set selection into multiple levels of progressive subselections, like divide-and-conquer. At each level, the ‘full set’ of samples are divided into smaller equal-size chunks using a seeded-chunking method (see Algorithm 2), such that chunks are well separated with closer samples going into the same chunk if possible. The similarity matrix for each chunk is computed, and weights that determine the size of the representative set for every chunk are computed based on the density/sparseness of each chunk (samples in a denser chunk are more similar to each other). Representative set selection is then performed on each chunk (for example, using apricot’s facility location approach). The representative samples selected from every chunk are merged into a new set, which becomes the ‘full set’ of the next level. This process iterates until the size of the similarity matrix of the merged set is feasible for the computing resources. Lastly, the similarity matrix of the final merged set is computed, and representative set selection is performed on it to get the final representative set of a desired size.

```
Algorithm 1: Hierarchical Representative Set Selection
Input: Full set $R$ of $N$ samples (k-mer counts). $m$: chunk size.
    $Q$: average representative-set size for each chunk. $L$:
    max size of similarity matrix ($L \times L$) feasible for computing resources. $n$: user-desired size of final representative set.
Output: $n$ representative samples.
1. Divide: divide $R$ into $l = \lceil N/m \rceil$ chunks using a seeded-chunking method (see Algorithm 2), each has $m$ samples.
2. Compute similarity matrix ($m \times m$) for each chunk $i$.
3. Compute weight $w_i$ for each chunk $i$ (see ‘Weighting scheme’).
 Set the representative-set size $RSS_i = w_i Q$
4. For each chunk $i$, perform representative set selection on its $m \times m$ similarity matrix to get $RSS_i$ representative samples.
5. Merge: sequentially merge $RSS_i$ representative samples from every chunk into a set $R'$ of $N' = \sum_i RSS_i$ samples.
6. $R \leftarrow R'$, repeat steps 1–5 until $N' \leq L$
7. Compute similarity matrix ($N' \times N'$); store it for any user-desired smaller-size representative set selection.
8. Perform representative set selection on the $N' \times N'$ similarity matrix to get $n$ representative samples.
* Once step 7 is completed, step 8 can be independently performed repeatedly for different $n$.
```

```
Algorithm 2: Seeded-Chunking Method
Input: Full set $R$ of $N$ samples. $m$: chunk size. $l$: number of
    chunks. $J$: size of a randomly selected subset used for
    selecting seeds.
Output: $l$ chunks.
1. Randomly select a subset $X$ of $J$ samples from $R$.
2. Perform the ‘farthest point sampling’ on $X$ to find a set $S$ of $l$ seeds:
    Initialization: Seed $s_1$ = a randomly selected sample from $X$.
        $S = \{s_1\}$. Distance from $x \in X$ to $S$:
        $d_S(x) = distance(x, s_1)$
    For $i = 2, \ldots, l$, repeat steps (a)–(c):
    a. Find the farthest sample away from the already selected
       seeds’ set $S$: $s_i = \mathrm{argmax}\ d_S(x),\ x \in X$
    b. Add $s_i$ as a new seed into $S$.
    c. Update the distance from $x \in X$ to $S$:
       $d_S(x) \leftarrow \min\{d_S(x),\ distance(x, s_i)\}$
3. Compute distances between each sample in $R$ and $l$ seeds.
4. Assign each sample in $R$ to its closest seed:
    i. Find all currently non-full chunks (i.e. size $< m$).
    ii. Assign the sample to its closest seed among all non-full
        chunks.
```

Computing the similarity matrix with k-mers is the runtime and memory bottleneck, while apricot’s runtime is relatively negligible in comparison. For direct representative set selection using apricot, the computational cost for the similarity matrix of the full set is $O(N^2)$. Using the hierarchical selection (considering one iteration of divide-and-merge with l chunks, chunk size m and the final merged set size N'), the computational cost is reduced to $O(lm^2) + O(N'^2) = O(N^2/l) + O(N'^2)$. The seeded-chunking has an added computational cost $O(Nl)$. So the total computational cost of the hierarchical selection is $O(N^2/l) + O(N'^2) + O(Nl)$, where $l \ll N$ and $N' \ll N$. With multiple iterations, the computational cost is further reduced. Since $m \ll N$, the memory requirement for computing the similarity matrix is greatly reduced.

Figure 1 illustrates applying the hierarchical representative set selection to 196 523 human RNA-seq samples in the SRA, using two levels (two iterations) of divide-and-merge. The first level has $l_1 =$ 197 chunks; the second level has $l_2 = 40$ chunks. ~200 representative samples are selected from each chunk of 1000 samples.

2.3.1 Seeded-chunking method

The ideal situation for the hierarchical representative set selection is that chunks have no overlaps. Here, the ‘overlap’ of chunks means data points and their close neighbors are in different chunks (e.g.

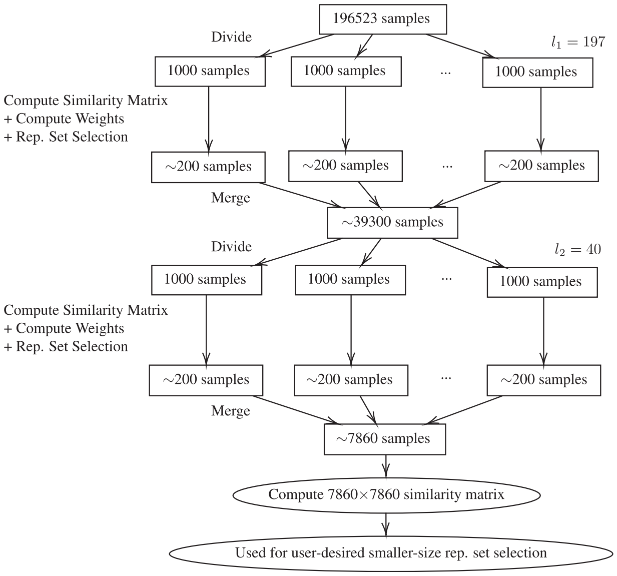

Fig. 1. Illustration of the hierarchical representative set selection for 196 523 human RNA-seq samples in the SRA

different chunks occupy the same dense cluster). When chunks have no overlaps, the union of the representative sets selected from every chunk would be similar to the representative set selected directly from the original set. When chunks have overlapping regions with similar densities, unnecessarily more representative data points may be selected in the overlapping regions from different chunks. The subsequent representative set selection on the merged set can alleviate this effect, but the over-use of the quota (i.e. a region's proportion of the desired number of representative data points) in the overlapping regions may cause other regions to have less quota. Thus, more separated chunks lead to more accurate hierarchical selection.

A sequential-chunking method that divides chunks sequentially along the SRA accession list can cause many overlaps, even complete overlaps between chunks. To overcome the introduction of overlaps with sequential chunking, we developed a novel seeded-chunking method (as shown in Algorithm 2). The seeded-chunking first uses the 'farthest point sampling' algorithm (Bronstein *et al.*, 2009; Pai et al., 2019) to find l seeds for l chunks, such that the seeds are farthest away from each other. That is, starting from a randomly selected seed, the seeds are chosen one at a time, such that each new seed has the largest distance to the set of already selected seeds (i.e. the largest minimum distance to the already selected seeds). The entire set of RNA-seq samples available at a large database cannot fit into memory at once, but if loading each sample one at a time when computing its distance to a seed, each sample would be repeatedly loaded for $l-1$ times. Thus, we randomly select a subset X from the full set and perform the 'farthest point sampling' on X to find l seeds. Each sample in the full set is assigned to its closest seed. Since chunks have equal sizes, the sample is assigned to its closest seed among all currently non-full chunks (i.e. their current size $< m$).

The seeded-chunking can generate well-separated chunks, with an added computational cost: $O(Nl)$ of computing similarities. Since $l \ll N$, this cost is fairly small compared to computing the full similarity matrix ($O(N^2)$). We benchmarked the seeded-chunking versus sequential chunking on various sizes of full sets (Supplementary Table S5). In all cases, the seeded-chunking outperforms the sequential chunking; as the full-set size increases, the seeded-chunking shows more advantages over the sequential chunking in the partial Hausdorff distance.

2.3.2 Weighting scheme

Denser chunks (in which samples are more similar to each other) should have fewer representative samples than sparser chunks, since we want representative data points to approximately evenly span the space of the original data, so that rare cell/tissue types can be sufficiently represented. Since chunks have equal sizes, denser chunks occupy smaller spaces than sparser chunks. Thus, we propose a novel weighting scheme ('mean^2-weighting scheme') to assign the representative-set size to each chunk based on their average density/sparseness.

The mean^2-weighting scheme is as follows. Let μ_i be the mean of distances between samples in chunk i; z_i is the size of chunk i (note that when N/m is not an integer, not all chunks are full); Q, l and m are as defined in Algorithm 1. Suppose the lth chunk is non-full: let $\alpha_l = z_l/m$. The weight w_i and the representative-set size RSS_i for chunk i are defined to be:

$$w_i = \mu_i^2/\text{weighted}_{\text{mean}}\{\text{all}\mu_j^2\} \qquad \text{where} i, j = 1, \ldots, l \tag{1}$$

$$RSS_i = w_i Q \qquad \text{for} i = 1, \ldots, l-1 \tag{2}$$

$$RSS_l = w_l \alpha_l Q \tag{3}$$

where the weighted$_{\text{mean}}\{\text{all}\mu_j^2\}$ uses weight $= 1$ for $j = 1, \ldots, l-1$ and weight $= \alpha_l$ for $j = l$. The following relationship holds:

$$\sum_i RSS_i = (l-1)Q + \alpha_l Q.$$

With the seeded-chunking, there could be multiple non-full chunks, but the same weighting method applies. The mean^2-weighting scheme is a heuristic to adjust the representative-set size for each chunk according to their average density/sparseness.

The code for the hierarchical representative set selection is available at https://github.com/Kingsford-Group/hierrepsetselection; the code for computing k-mer similarities is available at https://github.com/Kingsford-Group/jellyfishsim. We have two GitHub repositories since the second repository contains the code that can also be used for other k-mer similarity applications; other applications that need to compute k-mer similarities between samples can use the second repository only.

3 Results

3.1 Hierarchical selection achieves summarization quality close to that of direct representative set selection

Excluding SRA samples with no public access permission, aligned samples and samples with no valid 17-mers (read-lengths $<$ the k-mer size 17, or reads that contain many N's in the middle), we obtained 196 523 human bulk RNA-seq (Illumina) samples as the SRA entire set. Each sample corresponds to an SRA Experiment.

In this particular implementation of the hierarchical representative set selection, we use apricot's facility location approach as the base level to perform representative set selection on each chunk and on the merged set. We use cosine distance as the distance measure. In this context, we define the term 'summarization quality' as a measure of how well a selected subset represents the full set (i.e. 'representativeness'), evaluated by d_{HK}.

We refer to the direct representative set selection using apricot as 'direct apricot', which includes two parts: (i) computing the similarity matrix of the full set; (ii) applying apricot's facility location approach to the full similarity matrix. The main computational cost of direct apricot comes from part (i).

The motivation of the hierarchical representative set selection is to reduce the runtime and memory requirement of direct representative set selection, while not sacrificing too much summarization quality. Thus, we compared the performance between direct apricot, the hierarchical selection and random sampling, using the most recent 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets (Fig. 2; Supplementary Table S1). We also compared the three methods using the early-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets (in the earliest quarter of

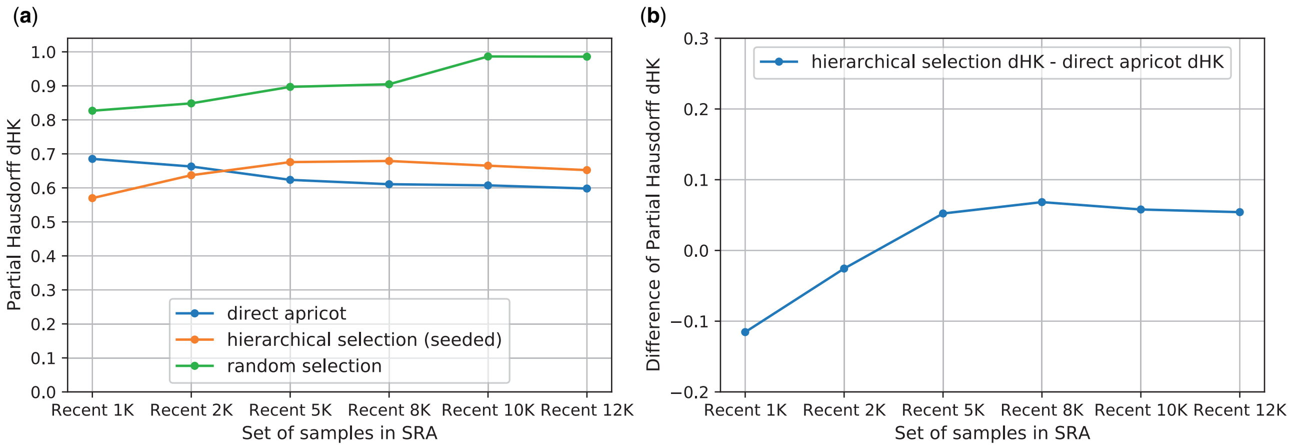

Fig. 2. Using the most recent 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets. $rep_{set}size/N$=0.1. (**a**) Partial Hausdorff distances d_{HK} of direct apricot, hierarchical selection and random selection. (**b**) Partial Hausdorff distances' difference: hierarchical selection d_{HK} – direct apricot d_{HK}

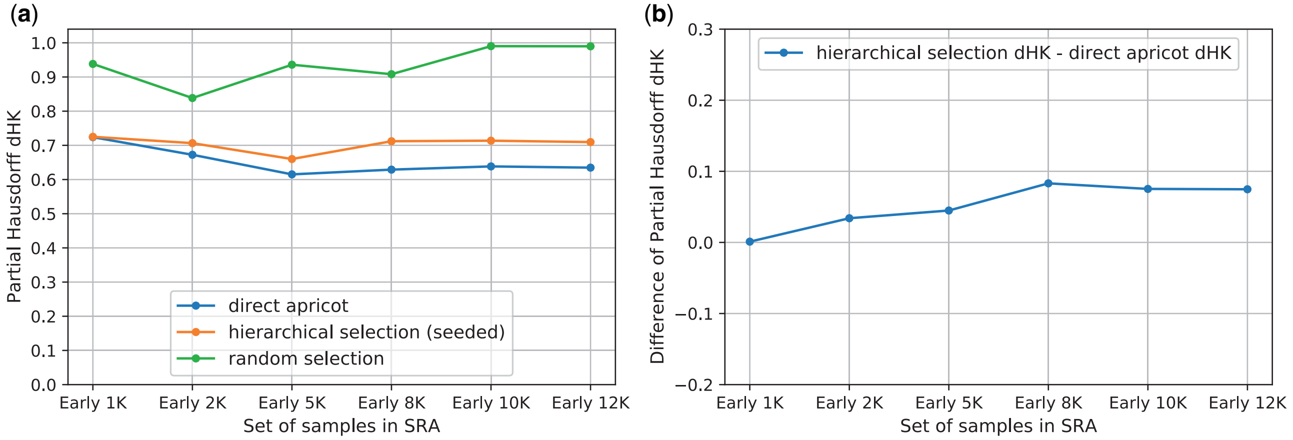

Fig. 3. Using the early-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets. $rep_{set}size/N$=0.1. (**a**) Partial Hausdorff distances d_{HK} of direct apricot, hierarchical selection and random selection. (**b**) Partial Hausdorff distances' difference: hierarchical selection d_{HK} – direct apricot d_{HK}

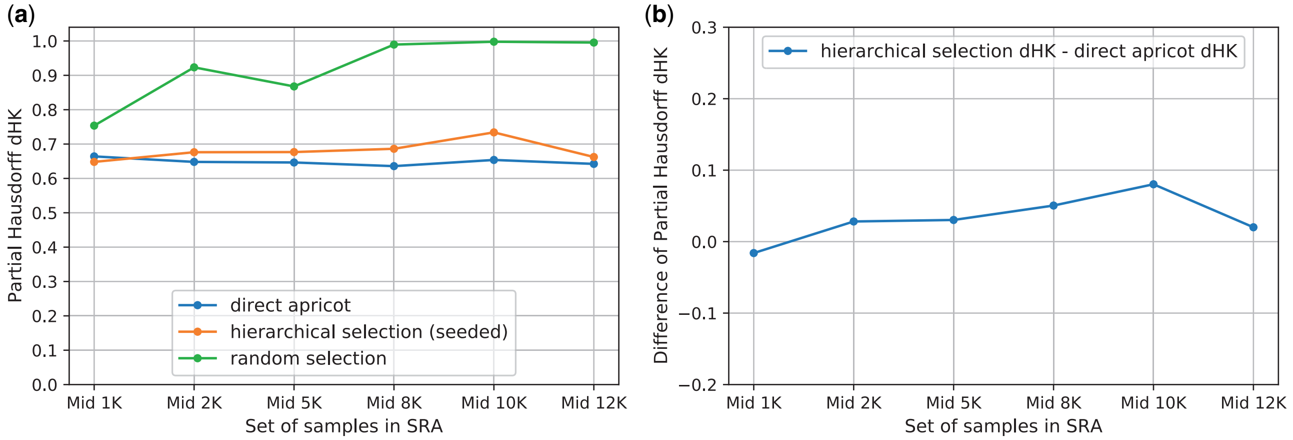

Fig. 4. Using the mid-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets. $rep_{set}size/N$=0.1. (**a**) Partial Hausdorff distances d_{HK} of direct apricot, hierarchical selection and random selection. (**b**) Partial Hausdorff distances' difference: hierarchical selection d_{HK} – direct apricot d_{HK}

the SRA time span, i.e. the 4th quarter of the SRA accession list) (Fig. 3; Supplementary Table S2), and using the mid-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets (around the middle of the SRA time span) (Fig. 4; Supplementary Table S3). For all these cases: we use 1 iteration, $rep_{set}size/N = 0.1$, $m = N/l$, $Q = m/5$. We set $l = 10$, except for $N = 1000$, $l = 5$.

The hierarchical selection achieves summarization quality close to that of direct apricot. For the recent 1000 and 2000 samples, the hierarchical selection performs better than direct apricot; for the recent 5000, 8000, 10 000 and 12 000 samples, the hierarchical selection is modestly less accurate than direct apricot but has representativeness close to that of direct apricot (Fig. 2). As N increases while keeping the same $rep_{set}size/N$, the d_{HK} difference of

Table 1. Runtime and memory reduction with the hierarchical selection over direct apricot using the most recent 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets

Reduction	Recent 1000 (select 100, $l = 5$)	Recent 2000 (select 200, $l = 10$)	Recent 5000 (select 500, $l = 10$)	Recent 8000 (select 800, $l = 10$)	Recent 10 000 (select 1000, $l = 10$)	Recent 12 000 (select 1200, $l = 10$)
Real-time reduction	1.47×	2.72×	6.04×	6.53×	7.02×	7.17×
User + Sys time reduction	2.99×	5.76×	8.33×	8.17×	8.4×	8.19×
Memory reduction	2.13×	2.9×	4.47×	4.59×	4.54×	5.22×

Table 2. Runtime and memory reduction with the hierarchical selection over direct apricot using the early-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets

Reduction	Early 1000 (select 100, $l = 5$)	Early 2000 (select 200, $l = 10$)	Early 5000 (select 500, $l = 10$)	Early 8000 (select 800, $l = 10$)	Early 10 000 (select 1000, $l = 10$)	Early 12 000 (select 1200, $l = 10$)
Real-time reduction	1.53×	2.14×	4.62×	4.99×	7.47×	6.35×
User + Sys time reduction	3.05×	4.89×	5.88×	6.2×	8.31×	7.06×
Memory reduction	2.2×	2.83×	4.33×	4.37×	4.68×	5.0×

Table 3. Runtime and memory reduction with the hierarchical selection over direct apricot using the mid-time 1000, 2000, 5000, 8000, 10 000, 12 000 samples in the SRA as the full sets

Reduction	Mid 1000 (select 100, $l = 5$)	Mid 2000 (select 200, $l = 10$)	Mid 5000 (select 500, $l = 10$)	Mid 8000 (select 800, $l = 10$)	Mid 10 000 (select 1000, $l = 10$)	Mid 12 000 (select 1200, $l = 10$)
Real-time reduction	0.94×	2.24×	4.92×	5.66×	6.92×	6.45×
User + Sys time reduction	3.13×	5.5×	7.12×	7.1×	7.79×	7.19×
Memory reduction	2.33×	2.78×	4.58×	4.5×	4.55×	5.35×

the hierarchical selection minus direct apricot initially increases from negative values, then levels out and then slightly decreases when N becomes very large, indicating that the d_{HK} difference between the hierarchical selection and direct apricot does not get larger when N further increases (Fig. 2). For the early and middle sets of samples, the trend of d_{HK} of the hierarchical selection is more similar to that of direct apricot, so their d_{HK} difference curves are flatter (Figs 3 and 4). For the early sets, the d_{HK} difference increases initially, and then slightly decreases and levels out when N becomes very large (Fig. 3). For the middle sets, the d_{HK} difference also eventually decreases when N becomes very large (Fig. 4). Overall, the average sacrifice in representativeness (% increase in d_{HK}) of the hierarchical selection versus direct apricot is 5.37%. These demonstrate that the hierarchical selection achieves summarization quality close to that of direct apricot, which is partially contributed by the seeded-chunking and mean^2-weighting.

The hierarchical selection substantially outperforms random sampling. Random sampling has substantially larger d_{HK} values than the hierarchical selection for all the recent, early and middle sets of samples, and this trend is consistent (Figs 2–4). When the full set becomes very large, random sampling's d_{HK} approaches 1.0 that is the maximum cosine distance. Random sampling follows the density distribution of the original set, so the rare cell/tissue types are not sufficiently represented, which yields large d_{HK}.

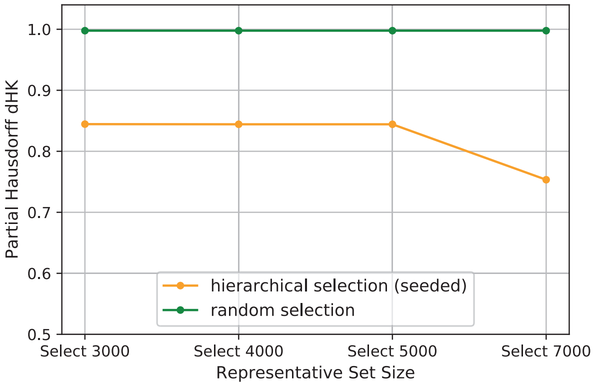

Fig. 5. Selecting different sizes of representative sets from the SRA entire set ($N = 196\ 523$ human RNA-seq samples): partial Hausdorff distances d_{HK} of hierarchical selection and random selection. For d_{HK}, $q = 0.0001$, so d_{HK} is the 21st-largest distance

3.2 Hierarchical selection substantially reduces runtime and memory of direct representative set selection

The hierarchical selection substantially reduces the runtime and memory usage of direct apricot. For the recent, early and middle sets of samples, the hierarchical selection and direct apricot were all run using 85 cores for parallelism; their runtime and memory are reported in Supplementary Tables S1–S3. As the full-set size increases, the real time and user + system time reductions generally increase (Tables 1–3). The user + system time reduction can reach 8.4× and the real-time reduction can reach 7.47× when $N = 10\ 000$. Real-time reductions are less than user + system time reductions, since in direct apricot the full similarity matrix computation is fully parallel, while in the hierarchical selection although the similarity matrix computation for each chunk is fully parallel, chunks are processed sequentially to reduce the memory usage. Runtime reductions for most recent samples are generally greater than those for early- and mid-time samples, as longer reads generate more k-mers. The memory reduction also generally increases as the full-set size increases (Tables 1–3). The memory reduction can reach 5.35× when $N = 12\ 000$.

For the entire set of SRA RNA-seq samples ($N = 196\ 523$), we use two levels of divide-and-merge (Fig. 1), with $m = 1000$, $Q = m/5 = 200$. At the 1st-level, $l_1 = 197$, the subset-size $J_1 = 2000$ (used for the seeded-chunking). At the 2nd-level, $l_2 = 40$, $J_2 = 1000$.

The hierarchical representative set selection makes selecting representative samples from the entire set of SRA RNA-seq samples feasible. When $N = 10\ 000$, direct apricot by computing the full $N \times N$ similarity matrix takes 4.1 h (using 85 cores) and 118.906 GB memory (Supplementary Table S1). With the $O(N^2)$ time complexity and $O(N)$ space complexity, for the entire SRA set ($N = 196\ 523$), the estimated runtime of direct apricot is 66 days (using 85 cores) and the estimated memory usage is 2336.776 GB, which is infeasible. The hierarchical selection on the entire SRA set ($N = 196\ 523$) takes 17.6 h (using 85 cores) and 56.430 GB memory, which makes this task fully feasible.

3.3 Hierarchical selection outperforms random sampling for the entire set of SRA RNA-seq samples

The hierarchical representative set selection outperforms random sampling for the entire set of SRA RNA-seq samples. We compare the hierarchical selection with random sampling by selecting different sizes of representative sets from the entire SRA set (Fig. 5; Supplementary Table S4). The hierarchical selection outperforms random selection in all these cases (Fig. 5); when selecting 7000 representative samples, the hierarchical selection outperforms random selection substantially, with a similar level of difference to those of smaller full sets. The d_{HK} values of the hierarchical selection are larger than that of selecting 1000 from the recent 10 000 samples, mainly because the ratios $rep_{set}size/full_{set}size$ are much smaller here, one magnitude smaller than the ratio 0.1 in selecting 1000 from 10 000. As the size of the representative set increases, the representativeness of the hierarchical selection initially barely changes but increases when going from selecting 5000 to selecting 7000 representative samples. Random sampling's d_{HK} are almost 1.0 (the maximum cosine distance) and do not change as the size of the representative set increases, indicating its poor performance on the entire SRA set. The final representative sets of different sizes are available at the GitHub repository: https://github.com/Kingsford-Group/hierrepsetselection.

4 Discussion

Our results show that the hierarchical representative set selection is a close estimate to the direct representative set selection, while substantially reducing the runtime and memory usage of the direct selection, which makes subset selections feasible on big data such as the entire available RNA-seq samples in the SRA.

The chunk-size m needs to be large enough to avoid chunk overlaps in the seeded-chunking (Supplementary Table S7). If the chunk-size is too small, a large dense blob containing many more samples than the chunk-size could have multiple chunks overlapping there. Chunks need to have equal sizes (except for 1 or 2 chunks when N/m is not an integer) to fit to the resource capacity, so when the chunk of the closest seed is full, the data point has to be assigned to its closest non-full seed. So although the seeds are spread out, if the chunk-size is not large enough, there could still be chunk overlaps in the seeded-chunking.

The choices of the parameters m (which yields l), Q and the number of iterations (levels) depend on the full-set size N and the memory of available computing resources. These parameters can affect the selection accuracy, runtime and memory usage of the hierarchical selection. In all the results here, we use $Q = m/5$, which means $rep_{set}size/chunk_{size} = \sim 0.2$ for each chunk. Increasing this ratio could increase the selection accuracy at each chunk, however, the subsequent merged set would be larger, causing more chunks at the next level and thus increasing the runtime. Decreasing m (and thus increasing l) could reduce the runtime, however, smaller chunks have more overlaps which decrease the overall selection accuracy (Supplementary Table S7). Thus, the overall design of the hierarchy with parameters' choices involves trade-offs between selection accuracy, runtime and the resource capacity.

The mean^2-weighting uses the average distance between samples of a chunk to indicate the chunk overall density. This is a heuristic. In a case that a chunk has several dense clusters that are far apart, causing a bigger average distance than the distances within clusters, the chunk may be assigned an unnecessarily larger weight. This may be partially addressed by performing clustering on each chunk and using the weighted-mean of average distances of all clusters as the chunk density. However, an accurate clustering incurs more computational cost. In our observation, most chunks do not have a distinctly clustered structure, and rather have a mixture of a few clusters and many roughly uniformly distributed data points. Thus, the mean^2-weighting is a viable trade-off between runtime and selection accuracy.

In addition to the partial Hausdorff distance, a useful evaluation for a representative set could be using the representative set as the training set to train classifiers to compare the classification accuracy. The classifiers could take the gene expression vectors or transcript abundance vectors as input features. We could compare the classification accuracy of the models trained by using different representative sets. This is a direction for future work.

Additional discussions can be found in the 'Additional discussions' section in Supplementary Materials.

5 Conclusion

We demonstrate that our novel hierarchical representative set selection method can greatly reduce the runtime and memory usage of the direct representative set selection with the full similarity matrix computation, while still achieving summarization quality close to that of the direct representative subset selection and substantially outperforming random sampling. This is the first approach to this problem that can scale to collections of the size of the full set of public human RNA-seq samples in the SRA.

Acknowledgements

The authors thank Dr. Guillaume Marçais for helping improve the speed of per-pair computation of k-mer similarity significantly. They also thank Dr. Cong Ma and Dr. Natalie Sauerwald for useful discussions.

Funding

This work was supported by the National Science Foundation Graduate Research Fellowship Program [DGE1745016 to L.H.T.] (any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation); the US National Science Foundation [1937540 to C.K.]; the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative [GBMF4554 to C.K.] and the US National Institutes of Health [R01GM122935].

Conflict of Interest: C.K. is a co-founder of Ocean Genomics, Inc.

References

Barrett,T. *et al.* (2012) BioProject and BioSample databases at NCBI: facilitating capture and organization of metadata. *Nucleic Acids Res.*, **40**(D1), D57–D63.

Boutsidis,C. *et al.* (2009) An improved approximation algorithm for the column subset selection problem. In *Proceedings of SODA*, pp. 968–977. New York, New York.

Bronstein,A. *et al.* (2009) *Numerical Geometry of Non-Rigid Shapes*. Springer-Verlag, New York, New York.

Choi,I. *et al.* (2019) Libra: scalable k-mer-based tool for massive all-vs-all metagenome comparisons. *GigaScience*, 8(2), giy165.

Daszykowski,M. *et al.* (2002) Representative subset selection. *Anal. Chim. Acta*, **468**(1), 91–103.

Elhamifar,E. *et al.* (2012a) Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pp. 19–27. Lake Tahoe, California.

Elhamifar,E. *et al.* (2012b) See all by looking at a few: sparse modeling for finding representative objects. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1600–1607. Providence, Rhode Island.

Frey,B. and Dueck,D. (2007) Clustering by passing messages between data points. *Science*, **315**, 972–976.

Garcia,S. *et al.* (2012) Prototype selection for nearest neighbor classification: taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, **34**(30), 417–435.

Hie,B. *et al.* (2019) Geometric sketching compactly summarizes the single-cell transcriptomic landscape. *Cell Syst.*, **8**(6), 483–493.E7.

Huttenlocher,D. *et al.* (1993) Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, **15**(9), 850–863.

Leinonen,R. *et al.*; International Nucleotide Sequence Database Collaboration. (2011) The sequence read archive. *Nucleic Acids Res.*, **39**(suppl L1), D19–21.

Li,Q. *et al.* (2010) A chunking method for Euclidean distance matrix calculation on large dataset using multi-GPU. In: *The Ninth International Conference on Machine Learning and Applications*, pp. 208–213. Washington D.C.

Marçais,G. and Kingsford,C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, **27**(6), 764–770.

Pai,G. *et al.* (2019) DIMAL: deep isometric manifold learning using sparse geodesic sampling. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 819–828. Waikoloa, Hawaii.

Pan,F. *et al.* (2005) Finding representative set from massive data. In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*, p. 8. Houston, Texas.

Schreiber,J. *et al.* (2020) apricot: submodular selection for data summarization in Python. *J. Mach. Learn. Res.*, **21**(161), 1–6.

Sim,D. *et al.* (1999) Object matching algorithms using robust Hausdorff distance measures. *IEEE Trans. Image Process.*, **8**(3), 425–429.

Wang,Y. *et al.* (2014) Representative selection based on sparse modeling. *Neurocomput.*, **139**, 423–431.