Analyses of Nuclear Reads Obtained using Genome skimming

Siavash Mirarab and Vineet Bafna

Abstract In this protocol paper, we review a set of methods developed in recent years for analyzing nuclear reads obtained from genome skimming. As the cost of sequencing drops, genome skimming (low-coverage shotgun sequencing of a sample) becomes increasingly a cost-effective method of measuring biodiversity at high resolution. While most practitioners only use assembled over-represented organelle reads from a genome skim, the vast majority of the reads are nuclear. Using assembly-free and alignment-free methods described in this protocol, we can compare samples to each other and reference genomes to compute distances, characterize underlying genomes, and infer evolutionary relationships.

1 Introduction

Ecology and conservation biology studies would greatly benefit from the ability to quickly and inexpensively analyze the biodiversity of an ecological niche to identify "hot spots" that could be targeted for preservation [1]. Genomic sequencing provides an attractive alternative to visual cataloging, as falling costs have made it possible to shotgun sequence a specimen sample with low cost [2, 3] as long as high coverage is not needed. However, traditional analyses, based on mapping reads to reference genomes or aligning pairs of genomes, do require assembling and finishing a reference genome, which can be prohibitively costly. It could be many decades before we have acquired high-quality genomes and identified population variants for each species to enable rapid monitoring. Due to this limitation, molecular biodiversity measurement has mostly relied on (meta)barcoding [4, 5, 6], which involves DNA

Siavash Mirarab
Electrical and Computer Engineering, 9500 Gilman Drive, La Jolla, CA, e-mail: smirarab@ucsd.edu

Name of Second Author
Computer Science and Engineering, 9500 Gilman Drive, La Jolla, CA, e-mail: vbanfa@ucsd.edu

sequencing of taxonomically informative marker genes (e.g., mtDNA COI [7, 4], 12S/16S [8], plastid [9], and ITS [10]). Biodiversity measurements have been conducted using computational methods [6, 11, 12, 13] applied to reference barcode databases (e.g., BOLD [14]). However, existing barcoding has several drawbacks. PCR for marker gene amplification requires high-quality DNA and thus cannot be applied to samples with heavily fragmented DNA. Moreover, barcodes are short and hence, their phylogenetic signal is limited [15]. For example, 896 of the 4,174 species of wasps could not be distinguished from other species using COI barcodes [16]; similarly, five phylogenetic clades of a frog species, important for conservation efforts, could not be distinguished using mtDNA markers [17].

One alternative that has emerged in the past ten years is genome skimming. A *genome skim* is a low-coverage acquisition of short reads from a sample, typically around 1-5 Gbp [18], providing low coverage (e.g., 0.2-4×) and insufficient for assembling contigs. Falling sequencing costs have made genome skimming cost-effective, but the inability to assemble data makes them hard to analyze. Initial approaches relied on assembling organelle genome or marker genes [19]. However, *accurate* assembly of organelle and marker regions from genome skims remains non-trivial [20]. Besides, this approach discards all reads from the nuclear genome (the vast majority of the data) and is thus wasteful.

More recently, a set of tools have been developed for *assembly-free* analysis of all genomic information from genome skims including the nuclear reads [21]. These methods are mostly *assembly- and alignment-free* and can be applied in the absence of a reference genome. The insight underlying these methods is that many of the downstream applications of genome skims can be performed successfully if we have an accurate way to measure the distance between two genomes from which two genome skims are generated. By distance, in the rest of this manuscript, we refer to the normalized hamming distance, also known as genomic nucleotide distance: the percentage of positions that do not match in a correct alignment of the two genomes. Of course, this quantity cannot be simply measured if we do not have the genome assemblies. But a hallmark of the methods described here is that without having the genome, we can accurately estimate the distance anyway. Once the distance is calculated, it can be used to 1) identify an unknown sample by comparing it to reference genome skims, 2) reconstruct the phylogenetic relationships and perhaps even delimitate species, and 3) measure population genetic measures of diversity. Below, we describe our preferred way, denoted the Skmer protocol, of addressing these questions using methods developed in recent years.

2 Materials

The Skmer protocol will use the software tools listed in Table 1. Each of the tools has ample documentation and user support through Github pages listed. In addition, the GitHub page labeled “Pipeline” in the table includes additional instructions for installation and use of other methods. It also includes links to several step-by-

Table 1 Software tools used in this tutorial

Tool	Use	URL
Skmer [22, 23]	Distance between two genome skims	github.com/shahab-sarmashghi/Skmer
RESPECT [24]	Accurate repeat/coverage estimates	github.com//shahab-sarmashghi/respect
APPLES [25, 26]	Phylogenetic placement using distances	github.com/balabanmetin/apples
CONSULT [27]	Read membership test (contamination)	github.com/noraracht/CONSULT
Kraken2 [28]	Read classification	github.com/DerrickWood/kraken2
Bowtie2 [29]	Mapping reads to reference genomes	sourceforge.net/projects/bowtie-bio/files/bowtie2/
BBTools [30]	Read cleanup	sourceforge.net/projects/bbmap/files
FastME [31]	Phylogenetic inference using distances	www.atgc-montpellier.fr/fastme
Pipeline[a]	Miscellaneous scripts	github.com/smirarab/skimming_scripts/

[a] A set of scripts mostly combining the methods above and converting the formats, when needed. Includes more detailed instructions for installing other tools.

Table 2 A glossary of terms and symbols.

L	: Length of the genome
c	: parameter asserting that cL bp were sequenced in the genome skim.
ϵ	: Sequencing error; Probability that a single bp is erroneously sequenced.
λ	: The average number of k-mers sampling a specific nucleotide.
ξ	: Expected number of k-mers sampling a specific nucleotide without sequencing error; $\xi = \lambda(1-\epsilon)^k$.
η	: Probability that a k-mer location is sampled without error. $\eta = 1 - e^{-\xi}$
r_i	: $r_i \equiv$ number of k-mers that appear exactly i times in the genome
	: Repeat spectrum = $[r_1, r_2, \ldots]$
d_{ij}	: Edit distance between genomes i and j
D	: $D = (d_{ij})$ denotes a matrix of genomic distances.
J	: Jaccard index of two sets A, B. $J = {}^{\lvert A \cap B \rvert}/_{\lvert A \cup B \rvert}$

step tutorials covering many of the topics mentioned here and video tutorials (see github.com/smirarab/tutorials/blob/master/skimming-tutorials.md).

In addition, it is helpful to have a dataset used as a test case throughout the manuscript. If the reader does not have one ready, they can download the THE LINK TO BE ADDED.

3 Methods

The inputs to the protocol described below are as follows.

- A set $\mathcal{S} = \{\mathcal{S}_1 \ldots \mathcal{S}_n\}$ of reference genome skims and reference assembled genomes $\mathcal{G} = \{\mathcal{G}_1 \ldots \mathcal{G}_N\}$, ideally all with known taxonomic labels.
- A set of $\mathcal{Q} = \{\mathcal{Q}_1 \ldots \mathcal{Q}_m\}$ of query genome skims with unknown identity.
- A reference phylogenetic tree with labels matching reference species ($\mathcal{S} \cup \mathcal{G}$).

Of these, most are optional; we only need to have at least some reference species (i.e., $n + N > 0$), but we do not need both assemblies and genome skims in the reference.

Queries and the reference phylogeny are optional. For phylogenetic analyses, we need at least four reference species (i.e., $n + N \geq 4$).

The general pipeline will include the following step. Note that we perform steps 1–3 on each reference skim in $\mathcal{S}$ and each query skim in $\mathcal{Q}$.

Step 1. Filter genome skim to remove adapters, remove duplicates, and (potentially) merge reads. This step requires `bbtools` [30]. See Section 3.1.

Step 2. Remove human and microbial contamination. Requires using `CONSULT` [27] and `bowtie` [29]. See Section 3.2.

Step 3. Characterize genome properties from skims, including length and the repeat spectrum. Requires `RESPECT`[24]. See Section 3.3.

Step 4. Build a reference library: compute the pairwise distances among references using `Skmer` [22], and build/update a reference phylogeny using `FastME` [31]. See Section 3.4.

Step 5. Characterize each query: compute its distances to references using `Skmer` and place it on the reference phylogeny using `APPLES` [26]. See Section 3.5.

The outputs include the following (See Table 2):

- Cleaned-up genome skims after step 2.
- The estimated sequencing error ϵ, coverage parameters (c, λ, ξ) for each genome skim as well as genome length (L) and repeat spectra $(\mathcal{R})$ of their underlying genomes.
- A distance matrix D, giving the distance between all pairs of reference species.
- A distance matrix D', giving the distance between every query and reference.
- A reference phylogeny T with appropriate branch lengths (with the same topology as the input tree, when one is provided).
- The placement of each query Q_i on the reference tree T in jplace format [32].
- Some figures and tables summarizing the data.

These outputs provide all the information necessary for several other tasks. Taxon identification is tightly related to estimating distances. Trivially, if the distance between a query and a known organism in the library is (almost) zero, species-level identification is complete. Otherwise, one can use the placement results to assign a taxonomic label to a query at the level appropriate. Essentially, we can assign the most fine-grained taxonomic label that would keep the group monophyletic (assuming all taxonomic groups are monophyletic in the reference tree). More generally, when a query is relatively novel, the phylogenetic placement results give a more information-rich identification of a query than taxonomic identification.

The outputs also immediately enable population genetic measurements. Consider a small number of samples from a population. If the population size is fixed (N_e) and evolving neutrally with expected μ mutations per base pair per generation (the Wright-Fisher model), the expected genomic distance (heterogeneity) d between the two genomes is $\mathbb{E}[d] = 2N_e\mu = \theta$ [33]. Thus, for a fixed μ, changes in the observed distance directly relate to changes in population size, a major point of interest in ecological research. Moreover, we can use θ to determine the sub-population structure. For example, we can compute the popular F_{ST} measure using $1 - \frac{\theta_w}{\theta_a}$, where

θ_w (respectively, θ_a) denotes the average distance between pairs within (respectively, across) sub-populations S and T. Finally, we can use distances to compute low-dimensional embeddings using multi-dimensional scaling and overlay them on geographical maps to understand species distribution and migration events.

Below, we dive into each step. We describe steps 1–3 for a given skim, which can be either a reference ($\mathcal{S}_i$) or query ($\mathcal{Q}_i$); the handling of these is identical in these steps. Steps 1–3 are skipped for reference genomes in $\mathcal{G}$. Step 4 uses the entire sets of references and Step 5 uses all the input.

3.1 Read cleanup

We use BBTools [30] to 1) remove adapters, 2) deduplicate reads, and 3) merge paired-end reads. In particular, the presence of high number of duplicate reads can be very detrimental. The merging of paired-end reads is optional: it is not important for downstream steps and if your reads are not paired-end with small insert sizes, it is not possible to merge most reads. Even if we do not merge reads files, however, it would be good to combine the two read files. The following commands can be used for read cleanup.

Read cleanup commands (bbtools)

Input: `S1_R1.fq` and `S1_R2.fq` for a paired-end genome skim.

```
# remova adaptors
bbmap/bbduk.sh \
   -Xmx24g -Xms24g in1=S1_R1.fq in2=S1_R2.fq  \
   out1=S1_R1DUK.fq out2=S1_R2DUK.fq ref=adapters,phix \
   ktrim=r k=23 mink=11 hdist=1 tpe tbo overwrite=true

$ deduplicate
bbmap/dedupe.sh \
   -Xmx24g -Xms24g in1=S1_R1DUK.fq in2=S1_R2DUK.fq \
   out=S1_OUT.fq overwrite=true
rm S1_R1DUK.fq S1_R2DUK.fq

# reformat to get separate out the reads
bbmap/reformat.sh \
  in=S1_OUT.fq out1=S1_OUT1.fq out2=S1_OUT2.fq overwrite=true
rm S1_OUT.fq

# merge the two reads
bbmap/bbmerge.sh \
   in1=S1_OUT1.fq in2=S1_OUT2.fq out1=S1_MERGED.fq \
   overwrite=true mix=t
```

```
rm S1_OUT1.fq in2=S1_OUT2.fq
```

This code assumes 24Gb RAM is available, which you can easily change to fit your machine. The final output, used in the next steps is `S1_MERGED.fq`. The `bbmerge.sh` part can be skipped if the user does not wish to merge the two reads; in that case, `S1_OUT1.fq` and `S1_OUT2.fq` can be simply concatenated.

Note that if your genome skim is particularly large, some of the steps above can be memory intensive. One way to solve the issue is to instead divide the inputs to small chunks, perform the steps on each, and then merge them back. We have implemented such an approach in a shell script called `bbmap_pipeline.sh` available on the Pipeline GitHub referenced in Table 1.

3.2 Contamination removal

Mathematical modeling and empirical analyses have shown that Skmer has substantial tolerance for contamination [34]. Nevertheless, distances can be over- or under-estimate the true distance if contamination levels are very high, especially when contaminants are very similar across samples. Removing contamination ~~can be performed using many approaches [?] and we cannot claim the approach described below is either optimal or always sufficient.~~ is an rea of active research[**?**]. The steps provide a reasonable starting point, and may be supplanted by better methods in the near future. We approach contamination using an exclusion strategy: we examine each read and remove it if we find evidence that belongs to a group of organisms other than what is intended. ~~We believe t~~ The dominant contamintion in genome skims is likely to come from human and microbial organisms. ~~types of contamination are common in genome skimming: human and microbial)~~ Below, we describe how to address these contaminants; other contaminants can be removed with the same methodology, but adding appropriate reference libraries.

For contaminants whose genome is well characterized (e.g. human), we use mapping tools such as Minimap2 [35] or bowtie [29] to map genome skims to the contaminant reference:

Human decontamination commands

Input: a reference human genome (e.g., GCF_000001405.39_GRCh38.p13) and an input genome skim (e.g., S1_MERGED.fq)

```
minimap2 -ax sr -t 24 GCF_000001405.39_GRCh38.p13.fna \
    S1_MERGED.fq > S1_MERGED_NoHM.fq

bowtie2 -x GCF_000001405.39_GRCh38.p13.fna -U S1_MERGED.fq \
    -S S1_MERGED_NoHM.fq --very-sensitive-local
```

For filtering microbial reads, where the contaminant reference is likely to be incomplete, we recommend using a k-mer matpping tool. Examples that we have tested are Kraken-II[28], or our own tool, CONSULT [27]. Comparing the two, CONSULT is more sensitive and can find more contaminants without sacrificing the false positive detection; on the negative side, it requires access to a machine with 120GB of memory and is somewhat slower than Kraken-II.

Building Kraken-II standard library

Output: the Kraken-library `krakenlib`

```
kraken2-build --standard --no-masking --use-ftp --db krakenlib
```

Note that Kraken-II can be used to search for human contamination and microbial contamination at the same time. To do so, simply run:

Adding human to Kraken-II library

Note: this updates the same library `krakenlib`.

```
kraken2-build --download-library human --db krakenlib
kraken2-build --build --db krakenlib
```

A more complete list of Kraken libraries can be found at Kraken-II webiste. Once the library is built, we can use the following command to search against the library

Search Kraken-II library

Input: a Skim file `S1_MERGED_NoHM.fq` and kraken library `krakenlib`

```
kraken2 --use-names --threads 24 --report S1_Kraken_Report \
  --db krakenlib --confidence 0 --classified-out S1_Cont.fq \
  --unclassified-out S1_MERGED_NoHM.fq S1_MERGED_NoHM.fq
```

TO fix: update consult commands and libraries. https://tera-trees.com/data/consult/v1.0.0/all_nbrhood_kmers_k32_p3l2clmn7_K15-map2-171_gtdb.tar.gz

Command

Input: a Skim file `S1_MERGED_NoHM.fq` and kraken library `krakenlib`

```
consult/main_search -i all_nbrhood_kmers_k32_p3l2clmn7_K15-map2-171_gtdb \
   -c 1 -t 60 -q processed_skims/
```

3.3 Characterizing the genome: repeat spectra, length, and coverage

Without access to a reference genome, answering questions that may sound simple becomes tricky and requires advanced algorithms: What is the coverage of the genome skim? What is the level of sequencing error? What is the length of the genome from which this skim is generated? How repetitive is the underlying genome? As it turns out, these questions are all interrelated. Our recent method, RESPECT [24], answers all these questions jointly for a given genome skim. RESPECT is based on a basic observation: The number of times a k-mer is observed in a genome skim can be predicted by multiplying two matrices: one that captures properties of sampling reads across the genome and is a function of coverage and sequencing error, and another that captures the genome length and how often a k-mer repeats across the genome. RESPECT uses several algorithmic tricks to decouple these two matrices. Specifically, RESPECT focuses on estimating the Repeat Spectrum, $\mathcal{R}$ (Table 2). It is defined by $\mathcal{R} = \{r_1, r_2, \ldots\}$, where r_i denotes the number of k-mers repeated exactly i times across a genome. Note that genome length $L = \sum_{i \geq 1} i r_i$ Another useful parameter is the *uniqueness ratio*, r_i/L, which estimates the extent of repetitive content in the genome. Knowledge of L and $\mathcal{R}$ gives us fundamental insights into the structure of a genome.

To run RESPECT, use the following command.

RESPCT

Input: A directory `refs` with a set of cleaned-up skims `S1_MERGED_NoHM.fq`, `S2_MERGED_NoHM.fq`, ... and genomes `G1.fasta`, `G2.fasta`, ...

```
respect -d refs/ -N 100 --debug -o respectout
```

Here, the `-N` option is used to reduce the running time (default: 1000). You can provide RESPECT with one or more input skims, and the output does not change; the only difference is in how efficiently it utilizes multiple cores. To run RESPECT on a single input instead of a directory, replace `-d` with a `-i` option.

The outputs of RESPECT are written in two tab-separated tables files called `estimated-parameters.txt` and `estimated-spectra.txt` (by default, saved in the current directory; can chage with `-o`), as shown in tables below.

Note that the uniqueness ratio is r_1/L, and indicates that these two sample genome skims are relatively repetitive. For comparison to other species, refer to the RESPECT [24]. Most other columns are self-explanatory, except for HCRM, which

Table 3 Example of an `estimated-parameters.txt` file

sample	input type	sequence type	coverage	genome length	uniqueness ratio	HCRM	sequencing error rate	average read length
`S1_MERGED_NoHM.fq`	sequence	genome-skim	7.68	991548841	0.83	321.4	0.006	109.0
`S2_MERGED_NoHM.fq`	sequence	genome-skim	1.69	731408229	0.85	268.8	0.006	110.3

Table 4 Example of an `estimated-spectra.txt` file

sample	r1	r2	r3	r4	r5
`S1_MERGED_NoHM.fq`	821,766,922	50,526,211	5,231,529	2,132,489	4,171,142
`S2_MERGED_NoHM.fq`	624,848,998	18,854,063	4,342,127	1,517,065	2,606,964

stands for high copy repeats per million. HCRM is the average frequency (per million base-pairs) of the 10 most highly repetitive k-mers. High HCRM values can be attributed to the presence of transposable elements.

3.4 Skmer reference library construction

Skmer[22] computes the distance between two genome skims, two assemblies, or a genome skim and an assembly. Skmer is used in two steps: building a reference library, and comparing a query skim (or genome) against a reference library. We cover the reference library building here and leave the query to the next step.

For each skim, Skmer uses Jellyfish [36] to compute k-mer frequency histograms; let M_l be the number of k-mers observed l times. Skmer estimates coverage parameters (Table 2) as follows:

$$h = \underset{l \geq 2}{\operatorname{argmax}}\, M_l \tag{1}$$

$$\text{Effective k-mer coverage, } \xi = \frac{M_{h+1} h + 1}{M_h} \tag{2}$$

$$\text{k-mer coverage, } \lambda = \frac{M_1}{M_h} \frac{\xi^h}{h!} e^{-\xi} + \xi(1 - e^{-\xi}) \tag{3}$$

$$\text{Sequencing error, } \epsilon = 1 - (\xi/\lambda)^{1/k} \tag{4}$$

$$\tag{5}$$

For an assembly, we set $\lambda = \infty$ and $\epsilon = 0$. Note that RESPECT *re-estimates* these parameters using the above calculations as a starting point, and also estimates $\mathcal{R}$, L.

Once these are calculated, Skmer can compute distances using the *Jaccard index* J–computed as the size of the the intersection of k-mer sets of the two genomes divided by the size of their union–while carefully accounting for dependence on coverage, sequencing error, and genome length. Thus, its approach is similar in principle to Mash [37] with the major difference being that it accounts for the low coverage.

Let $\eta = 1-e^{-\xi}, \zeta = \eta+\lambda-\xi$. For genomes with parameters, L_i, η_i, ζ_i ($i \in \{1,2\}$), Skmer estimates the distance using Jaccard index J as

$$D = 1 - \left(\frac{2(\zeta_1 L_1 + \zeta_2 L_2)J}{\eta_1\eta_2(L_1+L_2)(1+J)}\right)^{1/k}. \tag{6}$$

3.4.1 Reference library construction

To construct a Skmer library, we need to build a directory with all the reference genomes and genome skims (i.e., $\mathcal{S} \cup \mathcal{G}$) included. We then run:

Skmer: building the library

Input: a directory called `refs` including all reference skims `S1_MERGED_NoHM.fq`, `S2_MERGED_NoHM.fq`, ... and all the genomes `G1.fasta`, `G2.fasta`, ...

```
skmer reference refs -p 24
```

This will run Skmer on all the genomes and genome skims in the reference library. It will use 24 cores maximum. If you don't specify the number of cores, it uses all that is available on a machine. Note that Skmer cannot use more cores than there are skims, so you may want to reduce if you have fewer references. Also, each additional core requires the usage of more memory. If your machine does not have a lot of memory available per core, you can get around that limitation by reducing the number of cores. In general, when facing errors, rerunning with `--debug` can help diagnose the error.

The command will create a folder named `library` (can be adjusted using the `-l` option). A file called `CONFIG` inside this folder gives all the configurations used to build the library. In this folder, for each input file, we have three important files:

1. The `.hist` file gives k-mer frequency histogram (M_l)
2. The `.dat` file gives genome length (L), coverage (λ), sequencing error (ϵ), and average read length. For genomes, all but L are left as `NA`.
3. The `.msh` file includes the minimum-hashed version of k-mer sets produced using Mash. The default size is 100000 (can be changed with `-s`).

Among these, the `.dat` file is most useful. We can combine all the information in all these files into a single tab deliminated text file:

Collect Skmer statistics

Input: The skmer library `library`

```
grep "" library/*/*.dat|sed -e "s/:/\t/g" -e "s/^library.//" \
   -e "s:/[^/]*.dat\t:\t:"|sort -k2 > stats.csv
```

Occasionally, for a genome skim, estimates of coverage are left as `NA`. These are cases where the k-mer frequency profile did not make sense to Skmer. Oftentimes (though not always), something is wrong with these samples and they should be treated with care.

Once all references are processed, Skmer also computes all pairwise distances between reference phylogenies, producing the matrix D, which we will use in Section 3.4.3. The file containing the distance matrix is called `ref-dist-mat.tex` by default (change using `-o` option). This file can be read as a text file and visualized using any tool. An example script for (unsophisticated) visualization called `draw-distances.R` is available on the Pipeline GitHub referenced in Table 1.

3.4.2 Augmenting reference library

After the reference library is used, it can be augmented to add new references. The only point to keep in mind is that when adding to a library, you need to use the same k-mer length (`-k`; default 31), sketch size (`-s`; default 100000), and random seed number (`-S`; default 42). You can find the values used for your existing library inside the `CONFIG` file in that library. This should not be a problem as long as you are using default parameters. To augment a library to add a new skim or genome, use the following command.

Skmer: augmenting the library

Input: a new reference `NEW_REF.fq` and an existing library `library`

```
skmer query -a NEW_REF.fq library
```

If you want to add several new genomes or genome skims to the library, run the above command for each separately.

3.4.3 Inferring a reference phylogeny

Whether or not you already have a reference phylogeny available for your reference samples, we need to perform some phylogenetic analyses. Note that even if you do have a reference phylogeny, you need to recompute its branch lengths using the

(transformed) Skmer distances so that the branch lengths on your reference tree are consistent with Skmer distances. In our analyses, we use a leading distance-based phylogenetic inference method called FastME [31], though other tools could be used as well.

Before performing phylogenetic inference, you need to apply the Jukes-Cantor (JC) [38] phylogenetic correction (i.e., $-\frac{3}{4}\ln(1-\frac{4}{3}d)$) to computed hamming distances. You can do this manually using the `jc-correction.sh` script, available as part of the Pipeline repository provided in Table 1.

Skmer: Jukes-Cantor correction

Input: Skmer distance matrix `ref-dist-mat.txt`

```
jc-correction.sht ref-dist-mat.txt > ref-dist-jc.txt
```

You can alternatively ask Skmer to recompute distances with the JC correction, though this will take a bit more time than the simple transformation:

Skmer: Jukes-Cantor correction.

Input: existing library `library`

```
skmer distance -t -o ref-dist-jc library
```

Before actual phylogenetic inference, we need to reformat the distance matrix to the phylip format understood by FastME.

Reformatting the distance matrix

Note: `tsv_to_phymat.sh` is available on the Pipeline GitHub on Table 1.

```
tsv_to_phymat.sh ref-dist-jc.txt  ref-dist-jc.phy
```

Now, we are ready to infer the phylogeny. When you do not have any trees available, you can simply run:

FastME: *de novo* backbone inference

Input: The distance matrix `ref-dist-jc.phy`

```
fastme -i ref-dist-jc.phy -o reference.tre
```

If you already have a tree available, run:

FastME: re-estimate reference tree branch lengths

Input: an existing phylogeny `input.tre` and distance matrix `ref-dist-jc.phy`

```
fastme -i ref-dist-jc.phy -o reference.tre -u input.tre
```

Add commands for Bootstrapping and uncertainty.

3.5 Step 5: Characterizing the query

Now we analyze query genome skims with the goal of identifying them with respect to the reference set.

3.5.1 Query distance calculation

Computing distances between a query and a reference is in theory identical to computing the distances between references. Skmer uses the same algorithm in both cases. The only difference is that for queries, Skmer does not save the three files (`.hist`, `.dat`, and `.msh`) and only outputs the final distances. To compute distances from a query skim to all reference species, use:

Skmer: query distance calculation

Input: a query genome or genome skim `Query.fq` and an existing library `library`

```
skmer query -p 2 -t Query.fq library
```

This will output the distance from the query to each reference taxon in a file called `dist-Query.txt`. This command can be run for each query separately.

As before, `-p` controls the number of cores used and the `-t` option ensures the distances already have the JC correction applied. If you would instead like hamming distances, drop the `-t` option. Note the great similarity between this command and the one used under *augmenting the library* using Skmer; the only difference is that here, we do not use the `-a` option. This option tells Skmer to save the `.hist`, `.dat`, and `.msh` files of the query inside the `library`, making a query identical to a reference. Without `-a`, the query is not added to the reference library so that other queries are *not* compared to it. To find the relationship between query samples, you can simply treat them as references and add them all to the same library.

Importantly, note that the `library` folder is just a collection of sub-folders. Any reference can be added/removed from a library by moving corresponding sub-folders

in or out of `library`. After each addition/removal to a library, rerunning the `skmer distance` command will update the reference distance matrix.

3.5.2 Phylogenetic placement on an existing tree

The distances from a query to reference species immediately give you some information about their identity. When a query is not very close to one and only one reference species, there may still be doubt about its identity. Phylogenetic placement using distances [25] can help in such situations. Placement adds the query to the reference phylogeny, and hence, gives you a more clear picture in terms of how the query relates to other organisms. For example, if it falls somewhere within a genus, we can classify it as belonging to that genus.

To phylogenetically place the quer(ies) on the reference tree, you need first to reformat its distance vector.

Reformatting the distance vector

Input: the query JC distance vector file `dist-Query.txt` from Skmer.

```
convert_to_tsv.sh dist-Query.txt > dist-Query.tsv
```

Note: `convert_to_tsv.sh` is available on the Pipeline GitHub on Table 1.

If you have multiple queries, you can give all of them to `convert_to_tsv.sh` at the same time as multiple arguments (e.g., `dist-query*.txt`). We are not ready to use APPLES-II to perform the placement:

APPLES phylogenetic placement

Input: the query JC distance vector file `dist-Query.txt` from Skmer.

```
run_apples.py -t reference.tre -d dist-Query.tsv -o queries.jplace
```

The output is in the jplace format [32]. For each query, it gives you the position of the query on the reference tree. Users who prefer a more conventional format such as newick can use a program called gappa (`https://github.com/lczech/gappa`) to convert the format (in particular, use the `gappa examine graft` command). Note that the results show the relationship between queries and the reference species, but not the relationship between the queries. If you are interested to find the relationship between queries, you can treat the queries as additional reference species and repeat steps in Section 3.4.3.

4 Notes

Several points need to be kept in mind throughout the pipeline.

- While we provided some approaches for dealing with contamination, these methods are not always enough. Here, we specifically have looked for contamination by microbes, ignoring things like fungal contamination. Alternatively, if close-enough reference genomes are available for the group of interest, one can simply map reads to the reference genomes and only keep reads that map.
- Bot RESPECT and Skmer internally run Jellyfish to compute the histogram of k-mer frequencies. If one has already run Skmer, there is no need to repeat this time-consuming step for RESPECT. One can instruct RESPECT to use the output from Skmer; an example script to do so is provided under the Pipeline repository provided in Table 1.
- For delimitating species, we can first infer a phylogenetic tree and use branch lengths to decide what organisms belong to the same species. The so-called species (or barcode) gap [39] provides a path forward. The idea is that branch lengths within species and across species are drawn from two distinct distributions (low and high mean, respectively). This notion can be further formulated using forward speciation models like Yule [40, 41]. As mentioned earlier, Skmer distances closely relate to coalescence parameter θ, which also enables the use of F_{ST} as mentioned earlier.
- We focused on the identification of single-species samples in this tutorial. Methodological steps for analyzing mixed non-microbial samples are less studied. One exception is the MISA method [42] designed to study a mixture of exactly two species in a phylogenetic framework. More generally, metagenomic methods can be adopted but their effectiveness requires further study.

As we all know, even the simplest techniques go wrong from time to time. Would you therefore indicate any major problems or faults that can occur with your technique? Try to indicate the major sources of problems and how they can be identified and overcome. With reference to related techniques, any variations of the technique that you have described should also be made in this section, as well as–where relevant–an indication of the sensitivity of the method, timescale for the singled technique, etc. This ”Notes” section is a hallmark of this series and has been singled out for praise by a number of reviewers. Please try and make this section as extensive as possible by putting on paper all of your various experiences with the technique. Each ‘Note’ should be cross-referenced with the ‘Materials’ and ‘Methods’ sections, e.g. (see Note 1).

References

1. Megan A. Supple and Beth Shapiro. Conservation of biodiversity in the genomics era. *Genome Biology*, 19(1):1–12, 2018.

2. DNA Sequencing Costs-NHGRI. https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data.
3. Nebula Genomics, Partnering with BGI, Sets Industry Standard by Offering 30x Whole-Genome Sequencing for $299. https://www.biospace.com/article/releases/nebula-genomics-partnering-with-bgi-sets-industry-standard-by-offering-30x-whole-genome-sequencing-for-299/ , 2020.
4. P. D. N. Hebert, A. Cywinska, S. L. Ball, and J. R. deWaard. Biological identifications through DNA barcodes. *Proceedings of the Royal Society B: Biological Sciences*, 270(1512):313–321, 2003.
5. V. Savolainen, R. S. Cowan, A. P. Vogler, G. K. Roderick, and R. Lane. Towards writing the encyclopaedia of life: an introduction to DNA barcoding. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1462):1805–1811, 2005.
6. Pierre Taberlet, Eric Coissac, Francois Pompanon, Chrisian Brochmann, and Eske Willlerslev. Towards next-generation biodiversity assessment using DNA metabarcoding. *Molecular Ecology*, 21(8):2045–2050, 4 2012.
7. K. A. Seifert, R. A. Samson, J. R. DeWaard, J. Houbraken, C. A. Levesque, J.-M. Moncalvo, G. Louis-Seize, and P. D. N. Hebert. Prospects for fungus identification using CO1 DNA barcodes, with Penicillium as a test case. *Proceedings of the National Academy of Sciences*, 104(10):3901–3906, March 2007.
8. Miguel Vences, Meike Thomas, Arie van der Meijden, Ylenia Chiari, and David R Vieites. Comparative performance of the 16S rRNA gene in DNA barcoding of amphibians. *Frontiers in zoology*, 2:5, 2005. ISBN: 1742999425.
9. P. M. Hollingsworth, Laura L. Forrest, John L. Spouge, Mehrdad Hajibabaei, Sujeevan Ratnasingham, Michelle van der Bank, Mark W. Chase, Robyn S. Cowan, David L. Erickson, Aron J. Fazekas, Sean W. Graham, Karen E. James, K.-J. Kim, W. John Kress, Harald Schneider, Jonathan van AlphenStahl, Spencer C.H. Barrett, Cassio van den Berg, Diego Bogarin, Kevin S. Burgess, Kenneth M. Cameron, Mark Carine, Juliana Chacon, Alexandra Clark, James J. Clarkson, Ferozah Conrad, Dion S. Devey, Caroline S. Ford, Terry A.J. Hedderson, Michelle L. Hollingsworth, Brian C. Husband, Laura J. Kelly, Prasad R. Kesanakurti, J. S. Kim, Y.-D. Kim, Renaud Lahaye, H.-L. Lee, David G. Long, Santiago Madrinan, Olivier Maurin, Isabelle Meusnier, Steven G. Newmaster, C.-W. Park, Diana M. Percy, Gitte Petersen, James E. Richardson, Gerardo A. Salazar, Vincent Savolainen, Ole Seberg, Michael J. Wilkinson, D.-K. Yi, and Damon P. Little. A DNA barcode for land plants. *Proceedings of the National Academy of Sciences*, 106(31):12794–12797, 8 2009.
10. C. L. Schoch, K. A. Seifert, S. Huhndorf, V. Robert, J. L. Spouge, C. A. Levesque, W. Chen, E. Bolchacova, K. Voigt, P. W. Crous, A. N. Miller, M. J. Wingfield, M. C. Aime, K.-D. An, F.-Y. Bai, R. W. Barreto, D. Begerow, M.-J. Bergeron, M. Blackwell, T. Boekhout, M. Bogale, N. Boonyuen, A. R. Burgaz, B. Buyck, L. Cai, Q. Cai, G. Cardinali, P. Chaverri, B. J. Coppins, A. Crespo, P. Cubas, C. Cummings, U. Damm, Z. W. de Beer, G. S. de Hoog, R. Del-Prado, B. Dentinger, J. Dieguez-Uribeondo, P. K. Divakar, B. Douglas, M. Duenas, T. A. Duong, U. Eberhardt, J. E. Edwards, M. S. Elshahed, K. Fliegerova, M. Furtado, M. A. Garcia, Z.-W. Ge, G. W. Griffith, K. Griffiths, J. Z. Groenewald, M. Groenewald, M. Grube, M. Gryzenhout, L.-D. Guo, F. Hagen, S. Hambleton, R. C. Hamelin, K. Hansen, P. Harrold, G. Heller, C. Herrera, K. Hirayama, Y. Hirooka, H.-M. Ho, K. Hoffmann, V. Hofstetter, F. Hognabba, P. M. Hollingsworth, S.-B. Hong, K. Hosaka, J. Houbraken, K. Hughes, S. Huhtinen, K. D. Hyde, T. James, E. M. Johnson, J. E. Johnson, P. R. Johnston, E. B. G. Jones, L. J. Kelly, P. M. Kirk, D. G. Knapp, U. Koljalg, G. M. Kovacs, C. P. Kurtzman, S. Landvik, S. D. Leavitt, A. S. Liggenstoffer, K. Liimatainen, L. Lombard, J. J. Luangsa-ard, H. T. Lumbsch, H. Maganti, S. S. N. Maharachchikumbura, M. P. Martin, T. W. May, A. R. McTaggart, A. S. Methven, W. Meyer, J.-M. Moncalvo, S. Mongkolsamrit, L. G. Nagy, R. H. Nilsson, T. Niskanen, I. Nyilasi, G. Okada, I. Okane, I. Olariaga, J. Otte, T. Papp, D. Park, T. Petkovits, R. Pino-Bodas, W. Quaedvlieg, H. A. Raja, D. Redecker, T. L. Rintoul, C. Ruibal, J. M. Sarmiento-Ramirez, I. Schmitt, A. Schussler, C. Shearer, K. Sotome, F. O. P. Stefani, S. Stenroos, B. Stielow, H. Stockinger, S. Suetrong, S.-O. Suh, G.-H. Sung, M. Suzuki, K. Tanaka, L. Tedersoo, M. T.

Telleria, E. Tretter, W. A. Untereiner, H. Urbina, C. Vagvolgyi, A. Vialle, T. D. Vu, G. Walther, Q.-M. Wang, Y. Wang, B. S. Weir, M. Weiss, M. M. White, J. Xu, R. Yahr, Z. L. Yang, A. Yurkov, J.-C. Zamora, N. Zhang, W.-Y. Zhuang, and D. Schindel. Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi. *Proceedings of the National Academy of Sciences*, 109(16):6241–6246, 4 2012.

11. D. Steinke, M. Vences, W. Salzburger, and A. Meyer. TaxI: a software tool for DNA barcoding using distance methods. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1462):1975–1980, 2005.
12. Frederick A Matsen, Robin B Kodner, and E Virginia Armbrust. pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC bioinformatics*, 11(1):538, 1 2010.
13. Simon A. Berger, Denis Krompass, and Alexandros Stamatakis. Performance, accuracy, and Web server for evolutionary placement of short sequence reads under maximum likelihood. *Systematic biology*, 60(3):291–302, May 2011.
14. Sujeevan Ratnasingham and Paul D N Hebert. BOLD : The Barcode of Life Data System (www.barcodinglife.org). *Molecular Ecology Notes*, 7(April 2016):355–364, 2007. arXiv: gr-qc/9809069v1 ISBN: 1471-8286.
15. Michael J. Hickerson, Christopher P. Meyer, Craig Moritz, and Marshal Hedin. DNA Barcoding Will Often Fail to Discover New Animal Species over Broad Parameter Space. *Systematic Biology*, 55(5):729–739, October 2006. ISBN: 1063-5157.
16. DONALD L. J. QUICKE, M. ALEX SMITH, DANIEL H. JANZEN, WINNIE HALLWACHS, JOSE FERNANDEZ-TRIANA, NINA M. LAURENNE, ALEJANDRO ZALDÍVAR-RIVERÓN, MARK R. SHAW, GAVIN R. BROAD, SERAINA KLOPFSTEIN, SCOTT R. SHAW, JAN HRCEK, PAUL D. N. HEBERT, SCOTT E. MILLER, JOSEPHINE J. RODRIGUEZ, JAMES B. WHITFIELD, MICHAEL J. SHARKEY, BARBARA J. SHARANOWSKI, REIJO JUSSILA, IAN D. GAULD[DECEASED], DOUGLAS CHESTERS, and ALFRIED P. VOGLER. Utility of the DNA barcoding gene fragment for parasitic wasp phylogeny (Hymenoptera: Ichneumonoidea): data release and new measure of taxonomic congruence. *Molecular Ecology Resources*, 12(4):676–685, July 2012.
17. Evan McCartney-Melstad, Müge Gidiş, and H. Bradley Shaffer. Population genomic data reveal extreme geographic subdivision and novel conservation actions for the declining foothill yellow-legged frog. *Heredity*, 121(2):112–125, aug 2018.
18. Eric Coissac, Peter M. Hollingsworth, Sébastien Lavergne, and Pierre Taberlet. From barcodes to genomes: Extending the concept of DNA barcoding. *Molecular Ecology*, 25(7):1423–1428, 2016.
19. Shanlin Liu, Yiyuan Li, Jianliang Lu, Xu Su, Min Tang, Rui Zhang, Lili Zhou, Chengran Zhou, Qing Yang, Yinqiu Ji, Douglas W. Yu, and Xin Zhou. SOAP Barcode: revealing arthropod biodiversity through assembly of Illumina shotgun sequences of PCR amplicons. *Methods in Ecology and Evolution*, 4(12):1142–1150, 12 2013.
20. Ashot Margaryan, Christina Lehmkuhl Noer, Stine Raith Richter, Marlene Elise Restrup, Julie Lee Bülow-Hansen, Frederik Leerhøi, Emilia Marie Rolander Langkjær, Shyam Gopalakrishnan, Christian Carøe, M. Thomas P. Gilbert, and Kristine Bohmann. Mitochondrial genomes of Danish vertebrate species generated for the national DNA reference database, DNAmark. *Environmental DNA*, 3(2):472–480, March 2021.
21. Kristine Bohmann, Siavash Mirarab, Vineet Bafna, and M. Thomas P. Gilbert. Beyond DNA barcoding: The unrealized potential of genome skim data in sample identification. *Molecular Ecology*, 29(14):2521–2534, July 2020.
22. Shahab Sarmashghi, Kristine Bohmann, M. Thomas P. Gilbert, Vineet Bafna, and Siavash Mirarab. Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biology*, 20(1):34, December 2019.
23. Eleonora Rachtman, Shahab Sarmashghi, Vineet Bafna, and Siavash Mirarab. Quantifying the uncertainty of assembly-free genome-wide distance estimates and phylogenetic relationships using subsampling. *Cell Systems*, 13(10):817–829.e3, October 2022.

24. Shahab Sarmashghi, Metin Balaban, Eleonora Rachtman, Behrouz Touri, Siavash Mirarab, and Vineet Bafna. Estimating repeat spectra and genome length from low-coverage genome skims with RESPECT. *PLOS Computational Biology*, 17(11):e1009449, November 2021.
25. Metin Balaban, Shahab Sarmashghi, and Siavash Mirarab. APPLES: Scalable Distance-Based Phylogenetic Placement with or without Alignments. *Systematic Biology*, 69(3):566–578, May 2020.
26. Metin Balaban, Yueyu Jiang, Daniel Roush, Qiyun Zhu, and Siavash Mirarab. Fast and accurate distance-based phylogenetic placement using divide and conquer. *Molecular Ecology Resources*, 22(3):1213–1227, April 2022.
27. Eleonora Rachtman, Vineet Bafna, and Siavash Mirarab. CONSULT: accurate contamination removal using locality-sensitive hashing. *NAR Genomics and Bioinformatics*, 3(3):10.1101/2021.03.18.436035, June 2021.
28. Derrick E. Wood, Jennifer Lu, and Ben Langmead. Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20(1):257, December 2019.
29. Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9:357–359, 2012. arXiv: #14603 ISBN: 1548-7105 (Electronic)\r1548-7091 (Linking).
30. Brian Bushnell, Jonathan Rood, and Esther Singer. BBMerge – Accurate paired shotgun read merging via overlap. *PLOS ONE*, 12(10):1–15, 2017. Publisher: Public Library of Science.
31. Vincent Lefort, Richard Desper, and Olivier Gascuel. FastME 2.0: A comprehensive, accurate, and fast distance-based phylogeny inference program. *Molecular Biology and Evolution*, 32(10):2798–2800, 2015. ISBN: 1537-1719 (Electronic)\r0737-4038 (Linking).
32. Frederick A. Matsen, Noah G. Hoffman, Aaron Gallagher, and Alexandros Stamatakis. A Format for Phylogenetic Placements. *PLoS ONE*, 7(2):e31009, February 2012.
33. N. A. Rosenberg and M. Nordborg. Genealogical trees, coalescent theory and the analysis of genetic polymorphisms. *Nat Rev Genet*, 3(5):380–390, May 2002.
34. Eleonora Rachtman, Metin Balaban, Vineet Bafna, and Siavash Mirarab. The impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters. *Molecular Ecology Resources*, 20(3):1755–0998.13135, May 2020.
35. Heng Li. Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.
36. Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 3 2011.
37. Brian D Ondov, Todd J Treangen, Páll Melsted, Adam B Mallonee, Nicholas H Bergman, Sergey Koren, and Adam M Phillippy. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*, 17(1):132, December 2016.
38. T H Jukes and C R Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 3:21–132, 1969.
39. N. Puillandre, A. Lambert, S. Brouillet, and G. Achaz. ABGD, Automatic Barcode Gap Discovery for primary species delimitation. *Molecular Ecology*, 21(8):1864–1877, April 2012.
40. Jacob A. Esselstyn, Ben J. Evans, Jodi L. Sedlock, Faisal Ali Anwarali Khan, and Lawrence R. Heaney. Single-locus species delimitation: A test of the mixed yule-coalescent model, with an empirical application to Philippine round-leaf bats. *Proceedings of the Royal Society B: Biological Sciences*, 279(1743):3678–3686, 2012.
41. Tomochika Fujisawa and Timothy G. Barraclough. Delimiting Species Using Single-Locus Data and the Generalized Mixed Yule Coalescent Approach: A Revised Method and Evaluation on Simulated Data Sets. *Systematic Biology*, 62(5):707–724, September 2013.
42. Metin Balaban and Siavash Mirarab. Phylogenetic double placement of mixed samples. *Bioinformatics*, 36(Supplement_1):i335–i343, July 2020.