5240–5249 Nucleic Acids Research, 2012, Vol. 40, No. 12 Published online 9 March 2012
doi:10.1093/nar/gks227

Bayesian estimation of bacterial community composition from 454 sequencing data

Lu Cheng[1,*], Alan W. Walker[2] and Jukka Corander[1]

[1]Department of Mathematics and Statistics, P.O.Box 68 (Gustaf Hällströmin katu 2b), University of Helsinki, 00014 Helsinki, Finland and [2]Pathogen Genomics Group, Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridgeshire CB10 1SA, UK

Received September 2, 2011; Revised January 27, 2012; Accepted February 23, 2012

ABSTRACT

Estimating bacterial community composition from a mixed sample in different applied contexts is an important task for many microbiologists. The bacterial community composition is commonly estimated by clustering polymerase chain reaction amplified 16S rRNA gene sequences. Current taxonomy-independent clustering methods for analyzing these sequences, such as UCLUST, ESPRIT-Tree and CROP, have two limitations: (i) expert knowledge is needed, i.e. a difference cutoff between species needs to be specified; (ii) closely related species cannot be separated. The first limitation imposes a burden on the user, since considerable effort is needed to select appropriate parameters, whereas the second limitation leads to an inaccurate description of the underlying bacterial community composition. We propose a probabilistic model-based method to estimate bacterial community composition which tackles these limitations. Our method requires very little expert knowledge, where only the possible maximum number of clusters needs to be specified. Also our method demonstrates its ability to separate closely related species in two experiments, in spite of sequencing errors and individual variations.

INTRODUCTION

For simplicity, we use the term ‘bacterial communities’ to refer to ‘bacterial community composition’ throughout this article.

It is very challenging to estimate bacterial communities using traditional sequencing methods due to the high cost, and therefore, attention has rapidly shifted toward next-generation sequencing technologies.

Most bacteria share some conserved regions in the 16S rRNA gene (1), which are flanked by more variable segments. Thus, the variable parts can be used as a fingerprint for a specific bacterial species. Usually the variable regions are amplified by using the conserved regions as primers in (PCR). Then the amplicons are sequenced by next-generation sequencing technologies, such as 454 pyrosequencing, which results in large amounts of sequence reads.

Sequencing errors (mismatches, insertions and deletions) are inevitably generated during the sequencing phase. Chimeric sequences, which are hybrids of two or more parent sequences, are often formed in PCR. These errors contribute to the diversity of the reads, which easily leads to an inflated diversity estimate of the true number of species in the community.

There exists two standard types of algorithms for estimating bacterial communities: taxonomy-dependent algorithms and taxonomy-independent algorithms. The ribosomal database project (RDP) classifier (2) belongs to the first category. The user needs to import a reference sequence database which contains all relevant species, then a community label is assigned to each read by the naive Bayesian algorithm in the software. Taxonomy-dependent algorithms largely depends on the quality of the reference sequence database, which have a poor performance when there are many novel species in the sequencing data.

Current taxonomy-independent algorithms can be further categorized into hierarchical clustering algorithms (HC), greedy heuristic clustering algorithms (GHC) and Bayesian clustering algorithms (BC). As recommended by Sun *et al.* (3), we choose a representative software of each category for comparative purposes; HC: ESPRIT-Tree (4), GHC: UCLUST (5), BC: CROP (6). There are also other algorithms available, such as Mothur (7) and CD-HIT (8). However, earlier comparisons (3) have already shown that they are outperformed by ESPRIT-TREE and UCLUST, respectively. Taxonomy-independent algorithms try to cluster the sequencing data to several operational taxonomic units (OTUs), such that all sequences in an OTU are within a certain distance of its consensus sequence.

*To whom correspondence should be addressed. Tel: +358504155294; Fax: +358919151400; Email: lu.cheng@helsinki.fi

HC algorithms generally require a distance matrix between all sequences in the dataset, from which a hierarchical tree can be constructed. Then, by applying a user-specified cutoff to the hierarchical tree, leaf nodes within the cutoff are assigned into an OTU. Two most important issues of HC are how to define the distance and how to generate the distance matrix. The distance should have the following properties (4): (i) $d(A, B) = d(B, A)$, (ii) $d(A, A) = 0$, (iii) $d(A, B) \leq d(A, C) + d(B, C)$, where A, B and C are sequences. The distance given by pairwise sequence alignment does not necessarily fulfill the property (iii). Although the distance given by a multiple sequence alignment (MSA) fulfills all three properties, the quality of MSA is usually very poor due to the high divergence in the sequencing data in this context. Also, MSA suffers from the high computational cost. ESPRIT-Tree defines a probabilistic distance which fulfills the distance properties in most cases. It uses a set of heuristics to avoid calculating the whole distance matrix, while still managing to generate the hierarchical tree even for large data sets. ESPRIT-Tree calculates all results for cutoffs from 1% to 15% in a single run. A lower cutoff results in more OTUs than that of a higher cutoff.

For each considered sequence, GHC algorithms either assign it to an existing cluster if the distance to it is within a certain distance cutoff, or create a new singleton cluster. GHC algorithms are therefore usually very fast. UCLUST actually utilizes a reference database, thus it is not a purely taxonomy-independent algorithm in this sense. UCLUST usually generates more OTUs than other methods, and it can label some sequences as noise, which might discard useful information.

As far as we know, CROP is the only existing BC algorithm for clustering 16S rRNA gene sequencing data. CROP sets up a Bayesian model to cluster a set of sequences, which utilizes a Gaussian mixture model and a birth–death process. First CROP splits the dataset into small blocks and perform BC on the blocks. After that, the generated clusters are replaced by their consensus sequences. CROP further performs BC on the derived consensus sequences to get the final clustering result. CROP uses a lower bound and an upper bound as inputs for the BC, which can be transformed to a cutoff. It usually produces less OTUs than HC and GHC algorithms.

For all the methods mentioned above, the user needs to set several different cutoffs (3 and 5% are commonly used) to cluster the sequencing data. Then, the user needs to check the results to determine the best cutoff, which requires a lot of expert knowledge in practice. Also, the same cutoff does not have the same meaning in different software, since the distances are defined in different ways. Thus, it is required that the user has a clear understanding of properties of the distances before using the software.

Another flaw is that the current methods cannot separate closely related species into different OTUs, even when the underlying sequence information is robust enough to separate them. Instead, closely related species may be grouped together into one large OTU. It is known that different bacterial species/genera have different levels of inherent variation in 16S rRNA gene sequence. Some harbor more variation, while others might have several subgroups with only little variation. This means that, in practice, it is impossible to create a single OTU cutoff value that will result in accurate species-level designation in all cases. When we set the cutoff to a low value (e.g. 1%) to detect differences between closely-related species lots of spurious OTUs are generated due to polymerase errors, sequencing errors, chimera sequences and alignment errors, as well as individual variations in other, more highly variable, species. When setting the cutoff to a higher value (e.g. 3 or 5%), then closely-related, but distinct, species can be erroneously placed into one large OTU.

Considering the above limitations, we developed a novel taxonomy-independent method to estimate bacterial communities. The user only needs to specify the possible maximum number of clusters in the data. Our method can separate closely related sequences into different OTUs and tends not to generate spurious OTUs. Also it generates highly accurate consensus sequences from the identified OTUs.

We implemented our method in a software called BEBaC (Bayesian Estimation of Bacterial Communities), which can be freely downloaded at: http://www.helsinki.fi/bsg/software/BEBaC.

MATERIALS AND METHODS

Our method is based on a stochastic search clustering algorithm originally developed for applications in population genetics (9–12). The basic idea of the clustering algorithm is to calculate an unnormalized posterior probability for an arbitrary partition of the reads, such that any two partitions can be compared. Thus, the best partition can be inferred through a stochastic search process over the partition space.

Due to the biological and computational limitations of meaningfully aligning huge amounts of diverse sequences, we adopt a divide and conquer strategy to cluster the reads. First, an alignment free method is used to separate the reads to several large crude clusters. Then, aligning the reads within a crude cluster is more sensible since the reads are more similar and the number of reads is smaller. Thereafter, fine clusters are derived from the alignment. Figure 1 shows the workflow of our method, which consists of three modules: pregroup, crude clustering and fine clustering.

Pregroup and crude clustering

All the reads are assumed to be preprocessed such that all the reads are of high quality and similar lengths, e.g. 450∼550 bp. In the pregroup phase (Supplementary Pregroup algorithm), some heuristics are used such that highly similar reads are assigned to a pregroup, which significantly reduces the computational burden in the crude clustering phase.

Here we assume there are n read sequences in total, which are indexed by a set $N = \{1, 2, \ldots, n\}$. The subset of sequences for $s \subseteq N$ is represented as $\mathbf{x}^{(s)} = \{\mathbf{x}_i : i \in s\}$, thus $\mathbf{x}^{(N)}$ represents all the reads. We then transform each sequence $\mathbf{x}_i$ to a 3-mer count vector $\mathbf{y}_i$, where a 3-mer

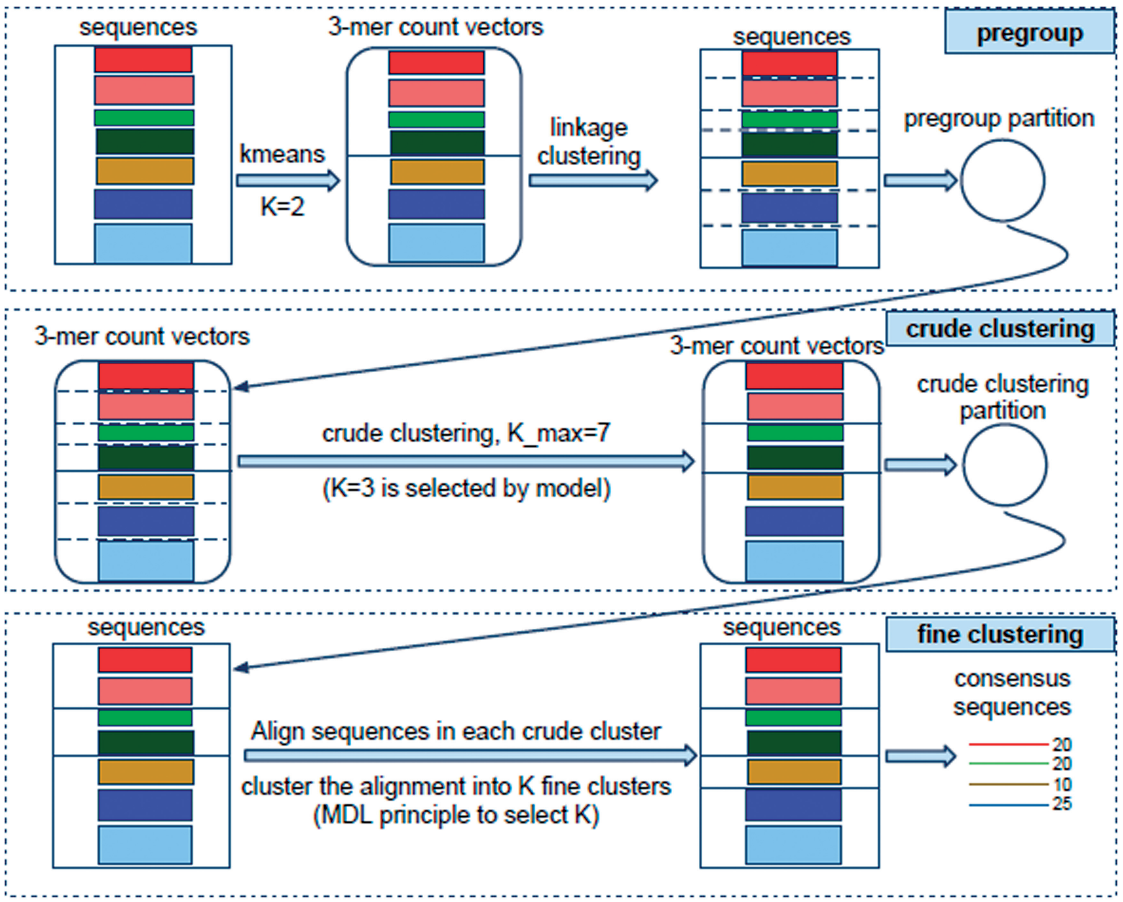

Figure 1. BEBaC workflow. BEBaC contains three modules: pregroup, crude clustering and fine clustering. During pregroup phase, highly similar reads are assigned to pregroups. Here we have seven different pregroups indicated with different colors. In the crude clustering phase, similar 3-mer count vectors are assigned into crude clusters, which resulted in three crude clusters. In the fine clustering phase, sequences in a crude cluster are aligned and further assigned into fine clusters. Here one of the crude clusters is further split into two fine clusters. Consensus sequences are then generated from the fine clusters. The number after each consensus sequence shows how many sequence reads it represents.

means 3 consecutive DNA bases ranging from 'AAA' to 'TTT'. Each element of the 1×64 vector $\mathbf{y}_i = (y_{i1}, y_{i2}, \ldots, y_{ij}, \ldots y_{i64})$ represents the count of its corresponding 3-mer in the given sequence $\mathbf{x}_i$. Hence the sequence set $\mathbf{x}^{(N)}$ is transformed to a 3-mer count set $\mathbf{y}^{(N)} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_n\}$.

Since there is information loss in the transformation from sequences to 3-mer count vectors, we name the clustering process of the 3-mer count vectors as crude clustering. Let $S = (s_1, s_2, \ldots, s_k)$ denote a partition of k crude clusters for $\mathbf{y}^{(N)}$, such that $\bigcup_{c=1}^{k} s_c = N$ and $s_c \cap s_{c'} = \varnothing$ for all pairs $\{c, c'\}$ ranging between 1 and k. Given the maximum number of clusters $K_{\max}$ defined by the user, we use $\mathcal{S}$ to denote the space of all possible partitions S, such that $k \leq K_{\max}$.

Now we want to find a posterior probability $p(S|\mathbf{y}^{(N)})$ for a given partition S, which allows us to search the partition space $\mathcal{S}$ for the best partition. According to Bayes's theorem, the posterior probability of S is

$$p(S|\mathbf{y}^{(N)}) = \frac{p(\mathbf{y}^{(N)}|S)p(S)}{\sum_{S \in \mathcal{S}} p(\mathbf{y}^{(N)}|S)p(S)} \qquad (1)$$

If we assume a uniform prior probability for any partition S, i.e. $p(S)$ equals a constant, then the right-hand side of (1) simplifies to $p(\mathbf{y}^{(N)}|S)$ divided by the constant in the denominator. Thus

$$p(\mathbf{y}^{(N)}|S) \propto p(S|\mathbf{y}^{(N)}) \qquad (2)$$

For all 3-mer count vectors in a crude cluster c, we assume the probability to observe any 3-mer is the same. Here we denote the probabilities to observe the 3-mers in cluster c as $(p_{c1}, p_{c2}, \ldots, p_{c64})$. Then, the conditional likelihood of the data is defined as

$$p(\mathbf{y}^{(N)}|\theta, S) = \prod_{c=1}^{k} p(\mathbf{y}^{(s_c)}|\theta, S) = \prod_{c=1}^{k} \prod_{j=1}^{64} p_{cj}^{n_{cj}} \qquad (3)$$

where $n_{cj} = \sum_{\mathbf{y}_i \in \mathbf{y}^{(s_c)}} y_{ij}$ is the total count of the j-th 3-mer in cluster c, and $\theta = \{p_{cj} | 1 \leq c \leq k, 1 \leq j \leq 64\}$ is the whole set of k multinomial probability vectors.

We assume a Dirichlet prior probability for the 3-mer probabilities in the same cluster, i.e. $(p_{c1}, p_{c2}, \ldots, p_{c64}) \sim \mathrm{Dir}(\lambda_{c1}, \lambda_{c2}, \ldots, \lambda_{c64})$. We use a reference prior, such that $\lambda_{cj} = 1/64$, $j = 1 \ldots 64$. More discussion about choosing priors for a multinomial likelihood can be found in (13). Dirichlet prior is a conjugate prior for multinomial likelihood, so that we can get an analytical form for

Equation (2). The joint distribution of the counts and θ, conditional on S equals

$$p(\mathbf{y}^{(N)}, \theta | S) = p(\mathbf{y}^{(N)} | \theta, S) p(\theta) \propto \prod_{c=1}^{k} \prod_{j=1}^{64} p_{cj}^{\lambda_{cj}+n_{cj}} \quad (4)$$

Since θ are nuisance parameters, we integrate them out to obtain (2). The resulting marginal likelihood is

$$\begin{aligned} p(\mathbf{y}^{(N)}|S) &= \int_\theta p(\mathbf{y}^{(N)}|\theta, S) d\theta \\ &= \prod_{c=1}^{k} \int \cdots \int \prod_{j=1}^{64} p_{cj}^{\lambda_{cj}+n_{cj}} dp_{c1} \cdots dp_{c64} \\ &= \prod_{c=1}^{k} \frac{\Gamma\left(\sum_{j=1}^{64} \lambda_{cj}\right)}{\Gamma\left(\sum_{j=1}^{64} \lambda_{cj} + n_{cj}\right)} \prod_{j=1}^{64} \frac{\Gamma(\lambda_{cj} + n_{cj})}{\Gamma(\lambda_{cj})} \end{aligned} \quad (5)$$

Equation (5) provides an analytical form of the marginal likelihood of $\mathbf{y}^{(N)}$ given the partition S, which is proportional to the posterior probability as suggested by Equation (2). Consequently, we define the best partition as

$$\hat{S} = \arg\max_{S \in \mathcal{S}} p(S|\mathbf{y}^{(N)}) = \arg\max_{S \in \mathcal{S}} p(\mathbf{y}^{(N)}|S) \quad (6)$$

We propose a greedy stochastic search algorithm to search for the best partition Ŝ in the partition space $\mathcal{S}$, where the results from the pregroup phase are used. The search algorithm is fast and efficient, but it can only be guaranteed to converge to a local optimum. By our practical experience, the results yielded by the algorithm are in general highly accurate and they outperform a Markov chain Monte Carlo-based approach (14) in terms of computational cost and time required to identify high density areas of $p(S|\mathbf{y}^{(N)})$. The greedy search algorithm is designed as follows:

Input

3-mer count vectors $\mathbf{y}^{(N)}$, pregroup partition, maximum number of clusters $K_{\max}$ defined by the user

Initialization

Cluster $\mathbf{y}^{(N)}$ into $K_{\max}$ clusters using complete linkage algorithm, set the current partition S as the initial partition. Note that special programming techniques are used here such that vectors in a pregroup will always be assigned to the same cluster.

Stochastic search

Apply each of the four search operators described below to the the current partition S in a random order. Then, if the resulting partition leads to a higher marginal likelihood (5), update the current partition S, otherwise keep the current partition. If all operators fail to update the current partition, then stop and set the best partition Ŝ as the current partition S.

(i) In a random order relocate all vectors in a pregroup to another cluster that leads to the maximal increase in the marginal likelihood (5). The option of moving vectors into an empty cluster is also considered, unless the total number of clusters exceeds $K_{\max}$.
(ii) In a random order, merge the two clusters which leads to the maximum increase in the marginal likelihood (5). This operator considers also merging of singleton clusters (only one pregroup in the cluster) that might be generated by the other operators.
(iii) In a random order, split each cluster into two subclusters using complete linkage clustering algorithm, where the distance between two pregroups are calculated as the average linear correlation coefficient between vectors in the two pregroups. Then try reassigning each subcluster to another cluster including empty clusters. Choose the split and reassignment that leads to the maximal increase in the marginal likelihood (5).
(iv) In a random order, split each cluster into m subclusters using complete linkage clustering algorithm as described in operator (iii), where $m = \min(20,\ n\text{Pregroup}/5)$ and nPregroup is the total number of pregroups in the cluster. Then try to reassign each subcluster to another cluster; choose the split and reassignment that leads to the maximal increase in the marginal likelihood (5).

Output

The best partition Ŝ, the highest marginal likelihood $p(\mathbf{y}^{(N)}\ |\hat{S})$.

Fine clustering

The fine clustering phase further assigns the sequences in a crude cluster into several fine clusters. Corander and Tang (CT) (10) proposed a second-order Markov model for unsupervised clustering of DNA sequence data, in which the partition is automatically determined given the maximum number of fine clusters. However, the method is too sensitive to sequencing errors to be directly applied to our case, because it detects some patterns from the sequencing errors such that there are usually some anomalous small clusters in the resulting partition. Barbara *et al.* (15) proposed a minimum description length (MDL) based criterion to determine the number of clusters in a similar problem, in which the description length is calculated for several candidate partitions and the one with the MDL is selected. We adopted CT's method together with Barbara's MDL-based criterion to determine the number of fine clusters, aiming to eliminate small clusters caused by sequencing errors.

We focus on sequences in a crude cluster which are indexed by a set $M = (1, 2, \ldots, m)$. Next the sequences are aligned using a MSA algorithm. We denote the set of aligned sequences by $\mathbf{z}^{(M)} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$, where all sequences are of the same length d. A specific sequence $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{id})$ is actually a sequence of length d consisting of bases in $\{A, C, G, T, -\}$. Let $S = (s_1, s_2, \ldots, s_k)$ denote a partition of k fine clusters for the set $\mathbf{z}^{(M)}$ of the aligned sequences. Thus sequences belonging to the fine cluster s_c are denoted as $\mathbf{z}^{(s_c)} = \{\mathbf{z}_i : i \in s_c\}$, which is a $|s_c| \times d$ matrix.

Table 1. Description length for transmitting $\mathbf{z}^{(M)}$ partitioned by S

Items	Amount	Unit length
Number of sequences m	1	Constant
Number of clusters k	1	Constant
Cluster labels of sequences	m	$\log(k)$, element ranges [1, k]
Coding table for cluster c	$5 \times d$	$\log(m+1)$, element ranges [0, m]
Coded sequences for cluster c	$\lvert s_c \rvert$	$\sum_{j=1}^{d}(\sum_{i=1}^{5} -p_{cij} log(p_{cij}))$

p_{cij} denotes the frequency of the i-th base for the j-th column in $\mathbf{z}^{(s_c)}$.

Now we consider the MDL criterion proposed by Barbara *et al.* Let us think about transmitting $\mathbf{z}^{(M)}$ partitioned by S through an optimal noiseless channel (16). First, we transmit the number of sequences m and the number of clusters k. Then we transmit the cluster labels of each sequence. After that we transmit the counts of each base in each column for a specific cluster $\mathbf{z}^{(s_c)}$, i.e. the coding table ($5 \times d$) for cluster $\mathbf{z}^{(s_c)}$. With the coding table, we compute the base distribution in each column and then encode each column of $\mathbf{z}^{(s_c)}$. In the end we transmit the encoded sequences. In this way, sequences in each cluster are transmitted. We want to find the partition S of $\mathbf{z}^{(M)}$ such that the smallest amount of bits need to be transmitted.

Table 1 shows the description lengths for all items in the coding process. The number of sequences m and the number of clusters k are could be described by a 32-bit integer, which is fixed length. The cluster labels are m values ranging [1, k]. For k different values, the average coding length is $\log(k)$ bits. Similarly, each element in the coding table, i.e. the number of a base in a column for a specific cluster, takes $\log(m+1)$ bits on average since it has $m+1$ different values ranging from 0 to m. With the coding table, we compute the base distribution for the j-th column in cluster c. We denote the distribution as $(p_{c1j}, p_{c2j}, \ldots, p_{c5j})$, which are the frequencies of $\{A, C, G, T, -\}$, respectively. Thus the average coding length for the j-th column is $\sum_{i=1}^{5} -p_{cij}\log(p_{cij})$, and the sum over all columns represents the average coding length for a sequence in cluster c.

Discarding the constant items in Table 1, we define the description length for partition S of $\mathbf{z}^{(M)}$ as

$$\mathrm{DL}(S, \mathbf{z}^{(M)}) = m \cdot \log(k) + \sum_{c=1}^{k} \Big\{ 5 \cdot d \cdot \log(m+1) + |s_c| \cdot (\sum_{j=1}^{d}\sum_{i=1}^{5} -p_{cij}\log(p_{cij})) \Big\} \quad (7)$$

Also we define the entropy for partition S of $\mathbf{z}^{(M)}$ as

$$\mathrm{entropy}(S, \mathbf{z}^{(M)}) = \sum_{c=1}^{k} |s_c| \cdot (\sum_{j=1}^{d}\sum_{i=1}^{5} -p_{cij}\log(p_{cij})) \quad (8)$$

which is the coded length of $\mathbf{z}^{(M)}$ given partition S, i.e. the sum of the last item in Table 1.

Next we use CT's method (10) to propose several candidate partitions, for which we calculate the description length and choose the one with MDL. CT's method works in a similar fashion as the crude clustering method, where the user only provides the alignment of the sequences and the maximum number of clusters $K_{\max}$. The method then searches for the best partition from all possible partitions with $\leq K_{\max}$ clusters.

Assume that the optimal partition has k^* clusters and $k^* < K_{\max}$. If we set $K_{\max}$ to $k^* - 1$, then our method is expected to return an estimate of the optimal partition having $k^* - 1$ clusters. A simple explanation for why there are $k^* - 1$ clusters in the estimated partition is as follows. Since the method aims at detecting k^* clusters in the sequences, when it is forced to detect at most $k^* - 1$ clusters, it merges the most heterogeneous cluster with the cluster that is the most similar to it, to achieve the maximum posterior probability. Thus by setting $K_{\max}$ from 1 to k^*, we obtain partition estimates with $k (1 \leq k \leq k^*)$ clusters, corresponding to the candidate partitions.

Then the description length and the entropy are calculated for the candidate partitions using (7) and (8), respectively. A good clustering algorithm will make the clusters more and more homogeneous as the number of clusters k increases, i.e. sequences in clusters become more and more similar. Hence it is expected that the entropy always decreases as k increases from 1 to k^*, while the description length usually achieves its minimum in the middle. The candidate partition with the MDL is selected as the final partition $\hat{S}$.

For each fine cluster s_c in $\hat{S}$, we construct a consensus sequence to represent all the sequences in s_c. To gain a better alignment, we first realign the sequences in the fine cluster s_c and denote the resulted alignment as $\mathbf{w}^{(s_c)}$. Then we choose the majority base for each column of $\mathbf{w}^{(s_c)}$. Next the majority base of each column is deleted if it is '–', or it is supported by < 50% sequences in $\mathbf{w}^{(s_c)}$. In the end, the remaining majority bases are concatenated to generate the consensus sequence.

RESULTS

Our method has been implemented in the software BEBaC, which is designed for parallel usage on a computer cluster. To illustrate the efficiency and accuracy of our method, we performed two experiments on different datasets using BEBaC, including a simulated dataset and a dataset from previous publications.

We compared BEBaC with previous OTU generation methods, which include UCLUST, ESPRIT-Tree and CROP. To evaluate the clustering results, we used the normalized mutual information (NMI) criterion (4). It measures the degree of information provided by the clustering results with respect to the ground truth, which ranges from 0 to 1. NMI = 1 means the clustering result agrees with the ground truth, while NMI = 0 means that the clustering result is random.

By default, ESPRIT-Tree provides all results by setting the cutoff from 1% to 15%. CROP has two cutoff options: *-s* and *-g*, corresponding to 3 and 5%, respectively. It is possible to adjust the cutoff to other values by setting the

-l and *-u* option, but it is not easy to interpret the two values to a cutoff. We use *-l* = 0.5 and *-u* = 1 to represent a lower cutoff than 3%, which is specially denoted by 1.5% here for convenience. For UCLUST, we set the cutoff to 3 and 5% for simplicity. Since a lower cutoff than 3% generates lots of spurious OTUs, we do not use such values. Since UCLUST typically marks some sequences as noise and does not assign them to any OTU, it is not possible to calculate the NMI value for its results. Thus we have created a singleton OTU for each of these noisy sequences to enable the calculation.

Experiment 1: simulated dataset

We simulated a 454 pyrosequencing dataset which contains 11 taxa, as shown by Figure 2. The phylogenetic tree in Figure 2 is designed such that there are multiple different distance levels in the tree. As can be seen from Figure 2, the minimum distance is 1% and the maximum distance is 25%. Particularly, we want to keep Taxon 9, Taxon 10 and Taxon 11 as a closely related group.

According to the phylogenetic tree in Figure 2, 11 consensus sequences (500 bp) for the 11 taxa were generated by Seq-Gen (17) software. Different substitution rates were applied to different regions in the 500 bp consensus sequences (Details in Supplementary Experiment 1).

Then, for each taxon *i*, we used a Gaussian distribution to model individual variations. That is, the number of mismatches between an individual sequence and the consensus sequence was obtained using the following three steps: (i) a random number is drawn from a Gaussian distribution $N(0, \sigma_i)$; (ii) the drawn value is multiplied by the length of the consensus sequence; (iii) the absolute value is truncated into the nearest integer. Hence, the average deviation of a random sequence from the consensus sequence was σ_i. For the 11 taxa, we set the σ_i ($i = 1, 2, \ldots 11$) to: 2, 4, 5, 4, 2, 1, 4, 3, 0.2, 0.6 and 0.4%. Using the specified deviation, we generated 2000 individual sequences (500 bp) for each taxon. The deviations of Taxa 1–8 were randomly selected from 1% to 5%, while the deviations for Taxa 9–11 were randomly selected from 0% to 1%.

Next, we generated sequencing errors for Taxa 1–8, while no sequencing errors were generated to Taxa 9–11. It is known (18) that the error rate is higher in homopolymer cases, which means a stretch of sequence containing the same base. We used the error rates shown in Table 2, which are transformed from those in (18). Here we counted ambiguous bases as mismatches for simplicity.

As a result, 22 000 sequences (~500 bp) of 11 taxa are generated. We expect that Taxa 9–11 will be detected as 3 separate OTUs instead of one, since 2000 sequences with little variations support each taxon to be an independent OTU. Thus the ground truth is set as 11 clusters, each of which contains 2000 sequences.

Table 3 shows the results given by different software. ESPRIT-Tree results from 7% to 15% are omitted here, since they have relatively low NMI values. As can be seen from Table 3, BEBaC successfully detected all 11 taxa, and has the highest NMI value. UCLUST, ESPRIT-Tree and CROP generate some spurious clusters when

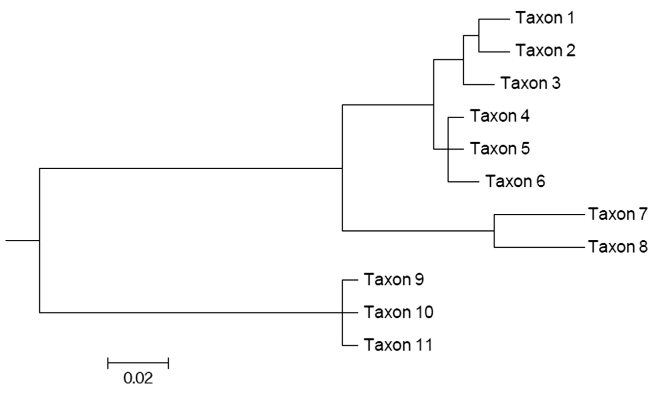

Figure 2. Phylogenetic tree of 11 taxa. Taxon 9–11 are 1% different from each other. Taxon 1 and Taxon 7 differ by ~ 25%. The distance matrix between the 11 taxa is shown in Supplementary Experiment 1.

Table 2. Sequencing error rates for simulated data

	Match	Mismatch	Insertion	Deletion
Non-homopolymer	0.9922	0.0018	0.004	0.002
Homopolymer	0.9862	0.0018	0.008	0.004

The error rates are transformed from Table 1 in (18), where ambiguous bases are counted as mismatches. Homopolymer is a stretch of sequence containing the same base.

Table 3. Simulated dataset results

Software(parameters)	Number of OTUs	NMI	Computational time
BEBaC ($K_{\max}$ = 20)	11	0.9981	19.11 CPU hours
UCLUST(3%)	1404	0.7570	30 s
UCLUST(5%)	27	0.9027	30 s
ESPRIT-Tree(1%)	10 609	0.5361	9.72 CPU hours
ESPRIT-Tree(2%)	558	0.8906	–
ESPRIT-Tree(3%)	9	0.9334	–
ESPRIT-Tree(4%)	9	0.9334	–
ESPRIT-Tree(5%)	8	0.9026	–
ESPRIT-Tree(6%)	7	0.8572	–
CROP(1.5%)	24	0.9321	1.88 CPU hours
CROP(3%)	9	0.9327	1.72 CPU hours
CROP(5%)	10	0.9265	1.16 CPU hours

ESPRIT-Tree provides all the results for cutoffs from 1% to 15% at one time. CROP(1.5%) is specially denoted for option (*-l* = 0.5 *-u* = 1). The simulated dataset contains 11 taxa, each of which contains 2000 sequences.

the cutoff is set to a low value. ESPRIT-Tree(3%) and CROP(3%) have the best NMI values, but they assign Taxa 9–11 into one OTU, thus only detect 9 OTUs from the dataset. We also checked that ESPRIT-Tree(2%) and CROP(1.5%) fail to separate Taxa 9–11. An interesting observation is that CROP increases the number of OTUs (9 to 10), when the cutoff is increased from 3% to 5%. It splits an OTU into two subclusters, which is to some extent counter-intuitive.

The results clearly show the limitations of current methods. When the cutoff is low, lot of spurious OTUs are generated, e.g. 1404 OTUs for UCLUST(3%) and 10 609 OTUs for ESPRIT-Tree(1%). When the cutoff is higher, the closely related species are assigned into one OTU.

Regarding the computational time, UCLUST costs the least time and BEBaC costs the most time. However, BEBaC can be run in parallel on a computer cluster, which requires 4.82 h to analyse the simulated dataset. Moreover, BEBaC actually detected all 11 clusters after the crude clustering phase, which took 1.02 h (3.12 CPU hours) on a computer cluster.

Experiment 2: Haas *et al.*'s eMC dataset

Haas *et al.* (19) generated an even composition Mock Communities (eMC) dataset for chimeric sequence detection, where the bacterial species in eMC were known in advance. Three regions (V1–V3, V3–V5, V6–V9) of the 16S rRNA gene of eMC were sequenced using 454 sequencing machines in four different sequencing centers. We used the V3–V5 region datasets since the PCR worked for all bacterial species. To utilize the sequencing data, we combined the datasets from four different sequencing centers into one dataset.

There are 114 reference sequences (including all rRNA operons), encompassing 22 bacterial species in total in the eMC dataset. Note that many species encode more than one 16S rRNA gene, and there are usually multiple distinct 16S rRNA genes for one species. They are usually quite similar, but this is not necessarily always the case. Each of the 114 reference sequences represents a distinct 16S rRNA gene, thus there are multiple reference sequences for each species. The species *Methanobrevibacter smithii* is an archaeon and *Candida albicans* is a eukaryote. Thus, *C. albicans* is a negative control since the primers are targeted only toward prokaryotes. Figure 3 shows the similarities between the 114 reference sequences. As can be seen from Figure 3, *C. albicans* and *M. smithii* are more distant to the other species, while *Staphylococcus aureus* and *S. epidermidis* are extremely similar to each other, which might result in a problem in discriminating the two species from each other.

After removing the chimeric sequences with Chimera Slayer, as proposed in (19), the eMC dataset contains around 110 000 sequence reads of varying lengths. However, BEBaC requires that the input sequences should be of similar lengths. Thus we removed sequences shorter than 450 bp and trim sequences longer than 550 bp to 550 bp from the end. After that, a sequence in the dataset was removed if $< 80\%$ of its sites have a quality score ≥ 20. In the end, 91 240 sequences ranging 450–550 bp are left in the eMC dataset.

To obtain the ground truth, we aligned (local alignment) each of the 91 240 sequences to all the reference sequences and labeled it with the the highest scored reference sequence's species label. Thus we derived the 'true' species distribution in the eMC dataset.

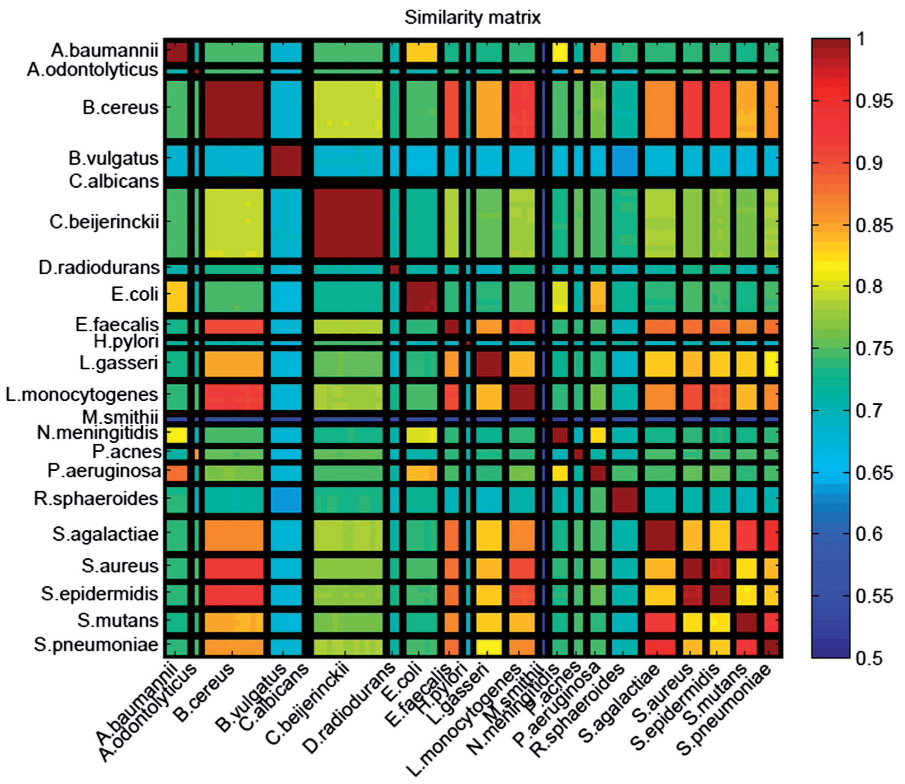

Figure 3. Similarity matrix between reference sequences of the eMC dataset. Higher value indicates higher similarity. *S. aureus* and *S. epidermidis* are highly similar. Rows and columns between black lines are reference sequences from the same species. Note that the column size for each species represents the number of reference sequences, which reflects the number of rRNA operons encoded by each species. The distance matrix was calculated as described in Supplementary Experiment 2. *C. albicans* has only one reference sequence, thus it is overlapped by the black lines.

However, we cannot directly compare the clustering results given by different software with the ground truth. Different software, or the same software with different parameters, try to cluster the sequences at different levels. Thus it is meaningless if we try to compare sub-species level clustering results with the species level ground truth.

A common hidden assumption of all the software is that a cluster can be approximately represented by its consensus sequence. Thus we can project a sub-species level OTU to species-level using the species-level label of its consensus sequence. To obtain the species-level label of a consensus sequence, we first align (local alignment) the consensus sequence with all the reference sequences, and then label the consensus sequence using the the highest scored reference sequence's species label. In this way, we reconstructed the species distribution in the eMC data.

Given the ground truth, we calculated the NMI scores of the reconstructed species distributions by different software options. Various experiences in the literature shows that 3 and 5% are good cutoffs for estimating the species distribution, thus we used these cutoffs for UCLUST, ESPRIT-Tree and CROP. In addition, we used a lower cutoff for ESPRIT-Tree(2%) and CROP(1.5%) to detect closely related species. Also, we are interested in how accurately the derived consensus sequences agree with their corresponding reference sequences. The accuracy is calculated as the number of matches in the local alignment divided by the sum of the alignment length and the flank bases of the consensus sequence (details in Supplementary Experiment 2).

Table 4 shows the results given by different software. Since ESPRIT-Tree does not provide consensus sequences for each OTU, we derived the consensus sequence for each OTU using the same method as described in section 'Fine clustering'. The ESPRIT-Tree(1%) result is not included here, because it generates 4627 OTUs, indicating that there are many spurious OTUs.

As can be seen from Table 4, BEBaC has a higher NMI score and accuracy than other software. UCLUST(3%) generated lots of spurious clusters. ESPRIT-Tree and CROP actually also generated some spurious clusters. Although the NMI scores for ESPRIT-Tree and CROP are relatively high, they fail to detect *S. epidermidis*, which is clustered together with *S. aureus*, as shown in Figure 4. This indicates that NMI score does not penalize too much if two original clusters are assigned into one predicted cluster. As can be seen from Figure 4, the BEBaC result is almost identical to the ground truth, while UCLUST(3%) differs a little from the ground truth.

For the eMC dataset, BEBaC requires 0.77 h for the crude clustering phase and 12.03h for the fine clustering phase on a computer cluster.

Table 4. eMC dataset results

Software(parameters)	Numbr of OTUs	NMI	Accuracy (std)	Time
BEBaC ($K_{\max} = 60$)	46	0.9980	0.9993 (0.0020)	34.5 h
UCLUST(3%)	3376	0.9527	0.9915 (0.0114)	2 min
UCLUST(5%)	229	0.9719	0.9734 (0.0328)	2 min
ESPRIT-Tree(2%)	333	0.9717	0.9711 (0.0250)	6.08 h
ESPRIT-Tree(3%)	115	0.9717	0.9620 (0.0361)	–
ESPRIT-Tree(5%)	45	0.9719	0.9583 (0.0484)	–
CROP(1.5%)	73	0.9722	0.9557 (0.0377)	6.70 h
CROP(3%)	45	0.9720	0.9562 (0.0467)	6.93 h
CROP(5%)	44	0.9719	0.9565 (0.0474)	5.02 h

ESPRIT-Tree provides all the results for cutoffs from 1% to 15% at one time. The NMI score is calculated using the reconstructed species distribution and the ground truth. The accuracy column shows the mean and standard deviation of all consensus sequences. 'h' in the time column means CPU hours and 'min' means minutes. CROP(1.5%) is specially denoted for option ($-l = 0.5$ $-u = 1$). The ESPRIT-Tree(1%) result is not shown here since it generateed 4627 OTUs.

DISCUSSION

Rapid environmental changes have the potential to generate a considerable amount of novel bacterial variation. Since bacteria affect human life in different aspects, estimating resident bacterial communities is often of interest. Due to the improvements in next-generation sequencing technology, huge amounts of the 16S rRNA gene sequencing data for estimating bacterial communities are currently generated. The problem of estimating bacterial communities thus transforms to clustering huge amounts of sequencing data.

Proper clustering of such sequencing data has thus become a common concern for many microbiologists. Comparing the sequences to a bacterial DNA database cannot detect novel species, but only yield high *P*-values for them. As shown by the experiments in the Results section, current taxonomy-independent algorithms can only cluster the sequencing data on a crude difference scale, such that closely related species are assigned into a large OTU. In many potential cases, separating these closely related species to independent OTUs is of considerable importance.

It should be noted that acquiring high quality sequencing data is the fundamental issue for accurately estimating bacterial communities. PCR bias, chimeric sequences, sequencing errors, etc., might lead to a misprediction of the underlying bacterial communities. Here only sequencing errors are modeled in the BEBaC framework. Effective chimera removing methods are already available (19), (20). PCR bias is generally unavoidable, although it can be reduced by using as few PCR cycles as possible during amplicon generation (21), and we leave it for future models.

MSA and alignment free methods could be utilized for quantizing the differences between sequences. MSA is very accurate in detecting differences between similar sequences. However, when there are huge amounts of highly distinct sequences, generating a MSA is computationally expensive and not desirable, since it is hard to generate an accurate MSA and large parts of the MSA will be filled with non-informative indels, i.e. '−'. Current fast MSA methods, such as MUSCLE (22) and MAFFT (23), actually also utilize alignment free techniques. Alignment free methods usually utilize *k*-mer information for detecting sequence differences, in which information

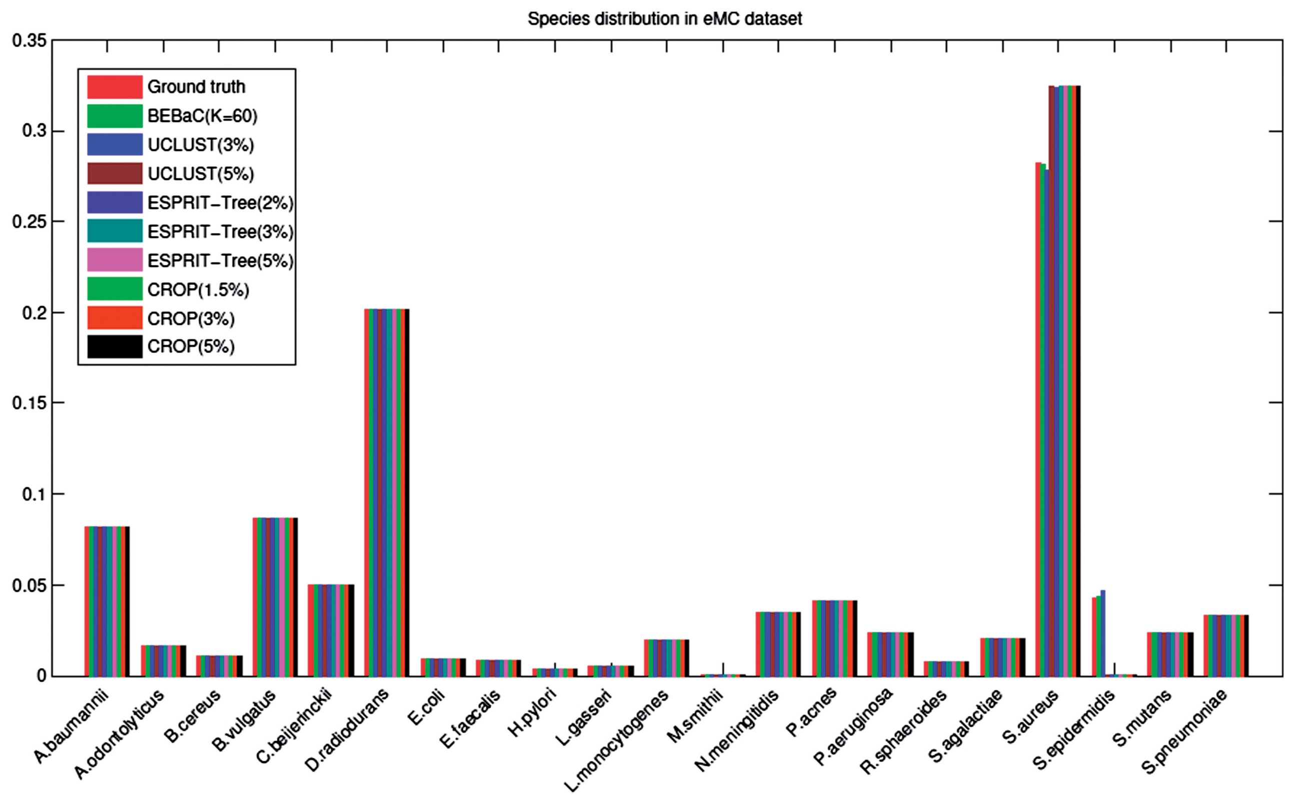

Figure 4. Reconstructed species distributions given by different software. BEBaC(K = 60) and UCLUST(3%) is almost the same as the ground truth, whereas the other software fails to detect the proportions of *S. aureus* and *S. epidermidis*. *C. albicans* is not detected by any software, thus we did not show it in this figure. *M. smithii* is correctly detected by all software, although this OTU only contains 14 sequences.

loss exists and thus sequences are anchored in a crude difference scale. Alignment free algorithms are fast, but not so accurate as 'true' MSA.

Two fundamental issues of developing an alignment free method are: which k-mer should be used and how the k-mer information should be used. As we know, a DNA sequence can be equivalently represented by its k-mers arranged in a unique order, e.g. the sequence '*AAAATT*' can be equivalently represented by its 3-mer in order of ('*AAA*', '*AAA*', '*AAT*', '*ATT*'). When we calculate the k-mer count vector, as described in crude clustering (see 'Materials and Methods'), the order information of the k-mers is lost. If we only calculate the occurrence of a k-mer, i.e. the count for a k-mer is either 1 or 0, then the count information of the k-mers is also lost. BEBaC preserves the k-mer count information, thus 3-mer is enough for the crude clustering purpose. The RDP classifier only utilizes the k-mer occurrence information, thus it needs to use long k-mer (8-mer) to conserve the difference information for clustering the sequences. The choice of 3-mer arises from the statistical perspective considering a balance between preserving order information of the original sequences and avoiding spurious clustering results. The latter would result from higher order k-mers, leading to extremely small word frequencies for a majority of the words.

BEBaC combines alignment free methods and alignment-based methods seamlessly. The dependencies within 3-mers have been considered both in crude and fine clustering phases. If two sequences are very similar, then their 3-mer count vectors should also be similar. Hence, if two 3-mer count vectors are not similar, then their corresponding sequences cannot be similar. This means similar sequences will not be assigned to different crude clusters in the crude clustering phase.

The greedy search algorithm in the crude clustering phase of BEBaC does not guarantee the global optimum, instead it converges to a local optimum. Thus one might get different results in different runs. However, the results are usually only slightly different and their posterior probabilities [Equation (5)] can be compared. Consequently, one can choose the clustering result with the highest posterior probability.

With the traditional methods, the user needs to compare the results given by different cutoffs, which is a tedious process. BEBaC relieves the users from the issue of selecting an appropriate cutoff. Instead, the user only needs to input the possible maximum number of clusters. If the detected number of clusters is smaller than the number $K_{\max}$ specified by the user, then no further input is required. Otherwise the user needs to increase the maximum number of clusters. BEBaC will automatically double $K_{\max}$ up to four cycles (2^4 of the original $K_{\max}$) until the detected number of clusters is less than $K_{\max}$. Thus normally the user does not need to increase the maximum number of clusters.

The computational time of BEBaC is acceptable on a computer cluster. Also the running time depends on the purpose of the user. If the interest is only in an approximate description of the underlying species distribution, then it is enough to perform only a crude clustering of the sequencing data, which is quite fast (see RESULTS).

CONCLUSION

We developed a novel method for detecting bacterial communities from 454 sequencing data. Compared with traditional methods, it determines the number of bacterial species automatically, while requiring no external reference databases. It is capable of separating closely related species into independent OTUs. The method could also be used for other problems that require clustering of large amounts of DNA sequences.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online: Supplementary Figure 1, Supplementary Pregroup Algorithm, Supplementary Experiment 1 and 2 and Supplementary Reference [24].

ACKNOWLEDGEMENTS

The authors would like to thank two anonymous referees for their invaluable comments, which helped to improve the paper significantly.

FUNDING

Finnish Population Genetics Graduate School (to L.C.); Sigrid Juselius Foundation (to L.C.); European Research Council (39784 to J.C.). Funding for opean access charge: European Research Council (39784 to J.C.).

Conflict of interest statement. None declared.

REFERENCES

1. Neefs,J.M., Van De Peer,Y., De Rijk,P., Chapelle,S. and De Wachter,R. (1993) Compilation of small ribosomal subunit RNA structures. *Nucleic Acids Res.*, **21**, 3025–3049.
2. Wang,Q., Garrity,G.M., Tiedje,J.M. and Cole,J.R. (2007) Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl. Environ. Microbiol.*, **73**, 5261–5267.
3. Sun,Y., Cai,Y., Huse,S., Knight,R., Farmerie,W., Wang,X. and Mai,V. (2012) A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis. *Brief. Bioinform.*, **13**, 107–121.
4. Cai,Y. and Sun,Y. (2011) ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. *Nucleic Acids Res.*, **39**, e95.
5. Edgar,R.C. (2010) Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, **26**, 2460–2461.
6. Hao,X., Jiang,R. and Chen,T. (2011) Clustering 16S rRNA for OTU prediction: a method of unsupervised Bayesian clustering. *Bioinformatics*, **27**, 611–618.
7. Schloss,P.D., Westcott,S.L., Ryabin,T., Hall,J.R., Hartmann,M., Hollister,E.B., Lesniewski,R.A., Oakley,B.B., Parks,D.H., Robinson,C.J. *et al.* (2009) Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl. Environ. Microbiol.*, **75**, 7537–7541.
8. Li,W., Jaroszewski,L. and Godzik,A. (2001) Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, **17**, 282–283.
9. Corander,J. and Marttinen,P. (2006) Bayesian identification of admixture events using multi-locus molecular markers. *Mol. Ecol.*, **15**, 2833–2843.
10. Corander,J. and Tang,J. (2007) Bayesian analysis of population structure based on linked molecular information. *Math. Biosci.*, **205**, 19–31.
11. Hanage,W.P., Fraser,C., Tang,J., Connor,T. and Corander,J. (2009) Hyper-recombination, diversity and antibiotic resistance in the pneumococcus. *Science*, **324**, 1454–1457.
12. Cheng,L., Connor,T.R., Aanensen,D.M., Spratt,B.G. and Corander,J. (2011) Bayesian semi-supervised classification of bacterial samples using MLST databases. *BMC Bioinformatics*, **12**, e302.
13. Bernardo,J.S. and Smith,A.F.M. (1994) *Bayesian Theory*. Wiley, Chichester, UK.
14. Corander,J., Gyllenberg,M. and Koski,T. (2009) Bayesian unsupervised classification framework based on stochastic partitions of data and a parallel search strategy. *Adv. Data Anal. Classif.*, **3**, 3–24.
15. Barbara,D., Couto,J. and Li,Y. (2002) COOLCAT: an entropy-based algorithm for categorical clustering. In: *Proceedings of CIKM '02:* pp. 582–589.
16. MacKay,D. (2003) *Information Theory, Inference, and Learning Algorithms.*. Cambridge University Press, Cambridge, UK.
17. Rambaut,A. and Grassly,N.C. (1997) Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.*, **13**, 235–238.
18. Gilles,A., Meglecz,E., Pech,N., Ferreira,S., Malausa,T. and Martin,J.F. (2011) Accuracy and quality assessment of 454 GS-FLX Titanium pyrosequencing. *BMC Genomics*, **12**, e245.
19. Haas,B.J., Gevers,D., Earl,A.M. *et al.* (2011) Chimeric 16S rRNA sequence formation and detection in Sanger and 454-pyrosequenced PCR amplicons. *Genome Res.*, **21**, 494–504.
20. Edgar,R.C., Haas,B.J., Clemente,J.C., Quince,C. and Knight,R. (2011) UCHIME improves sensitivity and speed of chimera detection. *Bioinformatics*, **27**, 2194–2200.
21. Bonnet,R., Suau,A., Dore,J., Gibson,G.R. and Collins,M.D. (2002) Differences in rDNA libraries of faecal bacteria derived from 10- and 25-cycle PCRs. *Int. J. Syst. Evol. Microbiol.*, **52**, 757–763.
22. Edgar,R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.
23. Katoh,K., Misawa,K., Kuma,K. and Miyata,T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.
24. Edgar,R.C. (2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, **5**, 113.