Bayesian estimation of bacterial community composition from 454 sequencing data

Lu Cheng, Alan Walker and Jukka Corander

25.01.2012

1 Pregroup algorithm

Here we assume there are \( n \) read sequences in total, which are indexed by a set \( N = \{1, 2, ..., n\} \). The subset of sequences for \( s \subseteq N \) is represented as \( x^{(s)} = \{x_i : i \in s\} \), thus \( x^{(N)} \) represents all the reads. We then transform each sequence \( x_i \) to a 3-mer count vector \( y_i \), where a 3-mer means 3 consecutive DNA bases ranging from 'AAA' to 'TTT'. Each element of the \( 1 \times 64 \) vector \( y_i = (y_{i1}, y_{i2}, ..., y_{ij}, ..., y_{i64}) \) represents the count of its corresponding 3-mer in the given sequence \( x_i \). Hence the sequence set \( x^{(N)} \) is transformed to a 3-mer count set \( y^{(N)} = \{y_1, y_2, ..., y_n\} \).

The goal of pregroup is to assign extremely similar sequences to a pregroup, which facilitates the greedy search algorithm in the crude clustering phase. As shown in Figure 1 in the original paper, the pregroup phase contains 3 steps: initial clustering, calculating the similarity matrix, and linkage clustering. Since the amount of sequences \( n \) might be very large, the similarity matrix \( (n \times n) \) will be too large to compute. Thus we adopted a divide and conquer strategy. As two highly similar sequences have similar 3-mer count vectors, we can use k-means algorithm to cluster the 3-mer count vectors, and then calculate the similarity matrix for a cluster.

1.1 Initial clustering

Here we use k-means algorithm to cluster the 3-mer count vectors \( y^{(N)} \). We set \( k = \lceil n/500 \rceil \), where we expect the average cluster size is 500. To calculate the distance between two 3-mer count vectors \( y_i \) and \( y_j \), we use the linear correlation coefficient between the \( y_i \) and \( y_j \).

After standard k-means algorithm has been applied, we get a partition \( S = (s_1, s_2, ..., s_k) \), where \( \bigcup_{c=1}^{k} s_c = N \) and \( s_c \cap s_{c'} = \emptyset \), for all pairs \( \{c, c'\} \) ranging between 1 and \( k \).

1.2 Calculating the similarity matrix

Next we calculate the similarity matrix for sequences in cluster \( t \) (\( 1 \leq t \leq k \)), which are denoted by \( x^{(s_t)} \). Let us denote \( m \) as the number of sequences in cluster \( c \) and the similarity matrix as \( M \). We adopted the distance measure defined by eq.(1) in [1].

We use the following algorithm to calculate \( M \).
for $i = 1 \rightarrow m$ do
  for $j = i \rightarrow m$ do
    $M_{ij} = \sum_{l=1}^{1024} \min(\tau_l(x_i), \tau_l(x_j))/[\min(L(x_i), L(x_j)) \times 5 + 1]$, where
    $\tau_l(x_i)$ is the number of $l$th 5-mer in $x_i$ and $L(x_i)$ is the length of sequence $x_i$.
    $M_{ji} = M_{ij}$
  end for
end for

All elements in $M$ range from 0 to 1, where higher value means higher similarity. Thus all elements in the diagonal of the similarity matrix $M$ are equal to 1.

1.3 Linkage clustering

First we transform the similarity matrix $M$ to a dissimilarity matrix $M^\sim$ by $M^\sim = 1 - M$, whose diagonal values are all 0. With the dissimilarity matrix $M^\sim$, we then use complete linkage algorithm to cluster the sequences into small pregroups. Here we set the cutoff by 0.1, which is obtained from various empirical datasets.

It should be noted that very similar 3-mer count vectors might be assigned to different initial clusters, thus then assigned to different pregroups. But similar 3-mer count vectors in different pregroups will be again assigned into the same crude cluster during the crude clustering phase (see METHODS in the original paper). The goal of the pregroup phase is to assign highly similar sequences into pregroups to reduce dimensionality of the problem, while not necessarily including all similar sequences in a single cluster.

2 Experiment 1: details

2.1 Consensus sequence generation

We generated 11 consensus sequences with the software Seq-Gen using the following command:

```
./seq-gen -mHKY -t3.0 -f0.3,0.2,0.2,0.3 -l500 -n1 -p3 ......
```

Here we specified different substitution rate weights (0.482364, 1.615264, 0.697283) for the consecutive 150bp, 200bp, 150bp segments, in order to mimic the evolutionary heterogeneity between different regions along the sequences. Figure 1 shows the similarities between the 11 generated consensus sequences. The similarity is calculated as the percentage of matches between two consensus sequences.

2.2 Analysis using different software

We use the following commands for running CROP

3%: ./CROPLinux -i INFILE -o OUTFILE -b 44 -e 2000 -s -m 15 -z 300
5%: ./CROPLinux -i INFILE -o OUTFILE -b 44 -e 2000 -g -m 15 -z 300
1.5%: ./CROPLinux -i INFILE -o OUTFILE -b 44 -e 2000 -l 0.99 -u 0.99 -m 15 -z 300
Figure 1: Similarity matrix between the 11 consensus sequences in experiment 1. The similarity is calculated as the percentage of matches between two consensus sequences. As can be seen from this figure, the differences between Taxa 9-11 are around 1%. Note that the lowest blue corresponds to 70% matches.

To configure UCLUST, we set the minimum size of a cluster to be 0, i.e. DEFAULT_MINSIZE=0. UCLUST still treats some sequences as noise and does not assign them to any OTU. Also we use the recommended file gold.fa for UCHIME_REFDB.

3 Experiment 2: details

3.1 Similarity matrix between the reference sequences

The calculation of similarity matrix is different from that in the pregroup phase. Since the reference sequences are of different lengths, we need to align them first to calculate the percentage of difference. However, the alignment is quite poor due to the huge diversity of the reference sequences. Thus we adopted an algorithm described below.

Here we denote $m$ as the number of reference sequences and the similarity matrix as $M$. We use the following algorithm to calculate $M$.

\[
\text{for } i = 1 \rightarrow m \text{ do} \\
\quad \text{for } j = i \rightarrow m \text{ do} \\
\quad \quad M_{ij} = \text{global alignment score between } i\text{th and } j\text{th reference sequences.} \\
\quad \quad M_{ji} = M_{ij} \\
\quad \text{end for}
\]
end for
for $i = 1 \rightarrow m$ do
  for $j = i \rightarrow m$ do
    $M_{ij} = M_{ij}/\sqrt{M_{ii}M_{jj}}$ \{Normalize the values in the matrix to 0 and 1.\}
    $M_{ji} = M_{ij}$
  end for
end for

All elements in $M$ range from 0 to 1, where higher value means higher similarity. Thus all elements in the diagonal of the similarity matrix $M$ are equal to 1.

### 3.2 Consensus sequence accuracy

First we perform a local alignment between a given consensus sequence and a reference sequence. The reference sequence is longer than the consensus sequence. We calculate the accuracy 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. Here we show an example alignment:

AAAACCTTTAAA (reference sequence)

| | | |

GCCCGTTG (consensus sequence)

As can be seen from the above local alignment: there are 5 matches; the length of the alignment is 6; there are 2 flanked bases beside the alignment in the consensus sequence. Thus the accuracy is calculated as $5/(6 + 2) = 0.625$.

### References

[1] Robert Edgar. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(1):113, 2004.