SUMATRA and SUMACLUST: fast and exact comparison and clustering of sequences

http://metabarcoding.org/sumatra/

celine.mercier@metabarcoding.org

Table of contents

Introduction

With the development of next-generation sequencing, efficient tools are needed to handle millions of sequences in reasonable amounts of time.

SUMACLUST and SUMATRA are a package of two programs developed by the **LECA**.

SUMACLUST and SUMATRA aim to compare sequences in a way that is fast and exact at the same time. These tools have been developed to be adapted to the type of data generated by DNA metabarcoding, i.e. entirely sequenced, short markers.

SUMATRA computes the pairwise alignment scores from one dataset or between two datasets, with the possibility to specify a similarity threshold under which pairs of sequences that have a lower similarity are not reported. The output can then go through a classification process with programs such as **MCL** or **MOTHUR**.

SUMACLUST clusters sequences using the same clustering algorithm as **UCLUST** and **CD-HIT**. This algorithm is mainly useful to detect the 'erroneous' sequences created during amplification and sequencing protocols, deriving from 'true' sequences.

Currently, SUMACLUST and SUMATRA are available as a package of two local programs that you can download and install on Unix-like machines.

Download and installation of SUMATRA and SUMACLUST

Download

SUMATRA and SUMACLUST can be downloaded from the public read-only **subversion** directory by using the following svn command:

```
svn co http://www.grenoble.prabi.fr/public-
svn/LECASofts/sumatra/tags/version1.0 suma_package
```

The versions downloaded this way are for Unix-like systems compatible with SIMD SSE2 instructions, openMP and POSIX threads. Pre-compiled versions of GCC can be found **here**, that might be helpful if you encounter problems compiling the programs. Send a mail at celine.mercier@metabarcoding.org for other versions, or if you have any inquiries.

Installation

Go into the newly created directories and compile:

For SUMATRA:

```
cd suma_package/sumatra

make
```

For SUMACLUST:

```
cd suma_package/sumaclust

make
```

Documentation

SUMATRA computes the pairwise alignment scores from one dataset or between two datasets, with the possibility to specify a similarity threshold under which pairs of sequences that have a lower similarity are not reported.

SUMACLUST clusters sequences using the same clustering algorithm as UCLUST and CD-HIT.

SUMATRA documentation

SUMATRA computes the pairwise alignment scores from one dataset or between two datasets, with the possibility to specify a similarity threshold under which pairs of sequences that have a lower similarity are not reported. The output can then go through a classification process with programs such as MCL or MOTHUR.

Input

Files must be in **FASTA format**.

Usage

```
sumatra [-l|L|a|n|r|d|g|x] [-t threshold value] [-p number of threads] dataset1 [dataset2]
```

First argument: the sequence dataset in fasta format to analyse.

Second argument: optionally the second sequence dataset in fasta format.

For help:

```
sumatra -h
```

Examples:

```
sumatra -t 0.97 my dataset.fasta >
pairs_of_seqs_with_similarity_>_97%.txt
```

```
sumatra -d -r -t 2 my dataset.fasta >
pairs_of_seqs_with_distance_<=_2_differences.txt
```

Options

```
-h : Print the help.

-l : Reference sequence length is the shortest.

-L : Reference sequence length is the largest.

-a : Reference sequence length is the alignment length
(default).

-n : Score is normalized by reference sequence length
(default).

-r : Raw score, not normalized.

-d : Score is expressed in distance (default: score is
expressed in similarity).

-t ##.## : Score threshold. If the score is normalized
and expressed in similarity (default), it is an identity,
e.g. 0.95 for an identity of 95%. If the score is
normalized and expressed in distance, it is (1.0 -
```

```
identity), e.g. 0.05 for an identity of 95%. If the score
is not normalized and expressed in similarity, it is the
length of the Longest Common Subsequence. If the score is
not normalized and expressed in distance, it is
(reference length - LCS length). Only sequence pairs with
a similarity above ##.## are printed. Default: 0.00 (no
threshold).

-p ## : Number of threads used for computation
(default=1).

-g : n's are replaced with a's (default: sequences with
n's are discarded).

-x : Adds four extra columns with the count and length of
both sequences.
```

Output

Results table description:

```
column 1 : Identifier sequence 1

column 2 : Identifier sequence 2

column 3 : Score

column 4 : Count of sequence 1  (only with option -x)

column 5 : Count of sequence 2  (only with option -x)

column 6 : Length of sequence 1 (only with option -x)

column 7 : Length of sequence 2 (only with option -x)
```

Example:

```
GQ245026_1      GQ245022_1      0.730769
GQ245026_1      GQ245021_1      0.702128
GQ245026_1      GQ245019_1      0.593220
GQ245026_1      GQ245017_1      0.631579
GQ245026_1      GQ245016_1      0.590164
GQ245026_1      GQ245014_1      0.653846
GQ245026_1      GQ245009_1      0.642857
GQ245026_1      GQ245008_1      0.660714
GQ245026_1      GQ245007_1      0.547170
GQ245022_1      GQ244830_1      0.651515
GQ245022_1      GQ244829_1      0.666667
GQ245022_1      GQ244828_1      0.980000
GQ245022_1      GQ244827_1      0.722222
GQ245022_1      GQ244826_1      0.901961
```

How does SUMATRA work

Each pair of sequences presenting a similarity above or equal to the chosen threshold is printed.

See **Similarity computation** for details on how the similarities between sequences are computed.

SUMACLUST documentation

SUMACLUST clusters sequences using the same clustering algorithm as UCLUST and CD-HIT. This algorithm is mainly useful to detect the "erroneous" sequences created during amplification and sequencing protocols, deriving from "true" sequences.

Input

Input file must be in **FASTA format**.

Usage

```
sumaclust [-l|L|a|n|r|d|e|o|g|f] [-t threshold value]
[-s sorting key] [-R maximum ratio] [-p number of threads]
[-B file name for BIOM-formatted output]
[-O file name for OTU table-formatted output]
[-F file_name_for_FASTA-formatted_output] dataset
```

Argument: the sequence dataset to cluster.

For help :

```
sumaclust -h
```

Examples:

```
sumaclust -t 0.97 my dataset.fasta >
clusters_of_seqs_with_similarity_>_97%.fasta
```

```
sumaclust -d -r -t 2 my dataset.fasta >
clusters_of_seqs_with_distance_<=_2_differences.fasta
```

Options

```
-h : [H]elp - print the help

-l : Reference sequence length is the shortest.

-L : Reference sequence length is the largest.

-a : Reference sequence length is the alignment length
(default).

-n : Score is normalized by reference sequence length
(default).

-r : Raw score, not normalized.

-d : Score is expressed in distance (default : score is
expressed in similarity).

-t ##.## : Score threshold for clustering. If the score
is normalized and expressed in similarity (default),
it is an identity, e.g. 0.95 for an identity of 95%. If
the score is normalized and expressed in distance, it is
(1.0 - identity), e.g. 0.05 for an identity of 95%. If
the score is not normalized and expressed in similarity,
it is the length of the Longest Common Subsequence. If
the score is not normalized and expressed in distance, it
is (reference length - LCS length). Only sequences with a
similarity above ##.## with the representative sequence
of a cluster are assigned to that cluster. Default: 0.97.

-e : Exact option : A sequence is assigned to the cluster
with the representative sequence presenting the highest
similarity score > threshold, as opposed to the default
```

```
'fast' option where a sequence is assigned to the first
cluster found with a representative sequence presenting a
score > threshold.

-R ##    : Maximum ratio between the counts of two
sequences so that the less abundant one can be considered
as a variant of the more abundant one. Default: 1.0.

-p ## : Multithreading with ## threads using openMP.

-s #### : Sorting by ####. Must be 'None' for no sorting,
or a key in the fasta header of each sequence, except for
the count that can be computed (default : sorting by
count).

-o : Sorting is in ascending order (default: descending).

-g : n's are replaced with a's (default: sequences with
n's are discarded).

-B ###   : Output of the OTU table in BIOM format is
activated, and written to file ###.

-O ###   : Output of the OTU map (observation map) is
activated, and written to file ###.

-F ###   : Output in FASTA format is written to file ###
instead of standard output.

-f : Output in FASTA format is deactivated.
```

Output

SUMACLUST's default output is in fasta format. There are four fields added in the headers of all sequences. Those fields are of the form [key=value;]. The four keys are "cluster", "cluster_score", "cluster_center" and "cluster_weight" and their values correspond respectively to the identifier of the center of the sequence's cluster, the similarity score of the sequence with this center, a boolean indicating whether the

sequence is the center of its cluster, and the total number of sequences in the cluster to which the sequence belongs.

Example where `seq_1` is a cluster center and `seq_2` is clustered with `seq_1`:

```
>seq 1 species=Heracleum maximum; count=3; cluster=seq 1;
cluster score=1.0; cluster center=True; cluster weight=5;
atcctattttccaaaaacaaacaaaggcccagaaggtgaaaaaag

>seq 2 species=Cnidium cnidiifolium; count=2;
cluster=seq 1; cluster score=0.955556;
cluster center=False; cluster weight=5;
atcctattttccaaaaacaacaaaggcccataaggtgaaaaaag
```

There is a possibility to print the clusters in **BIOM format** with the *–B* option, and/or in OTU map (observation map) format with the *–O* option. The FASTA output can then be deactivated with the *–f* option. The FASTA output is written to the standard output by default, but can be written to a file using the *–F* option.

In the following examples, the first one prints results in FASTA and BIOM formats, and the second one prints results in BIOM and OTU map formats:

```
sumaclust -B clusters of seqs with similarity > 97%.biom
my_dataset.fasta > clusters_of_seqs_with_similarity_>_97%.fasta
```

```
sumaclust -F -B clusters of seqs with similarity > 97%.biom -O
clusters_of_seqs_with_similarity_>_97%.txt my_dataset.fasta
```

How does SUMACLUST work

Clustering algorithm

SUMACLUST clusters sequences using the same clustering algorithm as UCLUST and CD-HIT. The problem is defined as follows:

With:

- $centre(ordered\ list)$ = the head of an $ordered\ list$
- $id(seq1, seq2)$ = the similarity score between the sequences $seq1$ and $seq2$

Data:

- $S = \{S_1, \ldots, S_m\}$ an ordered list of m sequences;
- $threshold \in \mathbb{R}_+^*$, the similarity threshold.

Problem:

Identifying an ordered partition $C = \{C_1, \ldots, C_n\}$ of S such that:

$$\begin{aligned} &\forall (C_i, S_j), \quad \text{if } S_j \in C_i \rightarrow id(centre(C_i), S_j) \geq threshold; \\ &\forall (C_i, C_j), \quad \rightarrow id(centre(C_i), centre(C_j)) < threshold. \end{aligned}$$

SUMACLUST browses through the dataset, in the order in which the sequences have been sorted with the *-s* option. By default, sequences are sorted by decreasing abundance, because this enables to identify 'true' and 'erroneous' sequences the best, as 'true' sequences tend to end up as cluster centers. The first sequence of the ordered list is considered the center of the first cluster. Each sequence, following the ordered list, is compared with the centers of the existing clusters, respecting the initial list's order. If the similarity of the query sequence with a center is above a chosen threshold, and their abundance ratio is below the maximum ratio chosen, the sequence is grouped in the cluster of this center. Otherwise, a new cluster is created with the query sequence as the center.

About the abundance ratio

An edge is created between a query sequence and a center sequence only if their abundance ratio, i.e. the query sequence's count divided by the center sequence's count, is below the maximum ratio chosen with the *–R* option. This can prevent sequences that are very abundant, and therefore likely true sequences, to be considered a variant of another true sequence that is only a little more abundant and very close to them.

See **Similarity computation** for details on how the similarities between sequences are computed.

Similarity computation

Similarity indice

A good way to evaluate the similarities between full-length sequences is to use indices based on the length of the **Longest Common Subsequence (LCS)**, and in particular, a good similarity indice is the length of the LCS divided by the length of the shortest alignment representing this LCS, giving an identity percentage. This is the similarity indice used by SUMATRA and SUMACLUST by default. Other similarity indices are available through the options.

Fast computation of the similarity

Lossless k-mer filter. Since we are usually interested in higly similar sequences, SUMATRA and SUMACLUST use similarity thresholds under which similarities are not reported. A lossless filtering step enables to only align couples of sequences that potentially have an identity greater than the chosen threshold. This filter is based on the number of overlapping k-mers that the sequences must share in order to have an identity at least equal to the threshold. With typical DNA metabarcoding datasets (a few millions sequences of 50-300 bp and threshold around 90-95% id), we empirically determined that the most efficient filtering was achieved with 4-mers and 5-mers.

Alignment within a diagonal band. Alignments are computed using a Needleman-Wunsch algorithm. In the scoring system used, matches are rewarded by one point, and mismatches and insertions/deletions are not penalised. The computation of the length of the LCS and the length of the alignment by the NWS algorithm has a quadratic complexity in time. It is responsible for most of the computation time. At high identity thresholds, the alignment computation can be done only in a diagonal band of the alignment matrix, gaining a considerable amount of time depending on the threshold.

Parallelization. There are two levels of parallelization implemented in SUMACLUST and SUMATRA. Both the filtering and the alignments steps are optimized with the use of Simple Instruction Multiple Data instructions (SIMD). Since 4-mers enable to work easily with SIMD instructions, we implemented a 4-mer filter. Moreover, both programs can be run on multiple threads.