Sumatra: fast and exact comparison of sequences

metabarcoding.org/sumatra

Introduction

With the development of next-generation sequencing, efficient tools are needed to handle millions of sequences in reasonable amounts of time. Sumatra is a program developed by the LECA. Sumatra aims to compare sequences in a way that is fast and exact at the same time. This tool has 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. Currently, Sumatra is available as a program that you can download and install on Unix-like machines.

Download and installation of Sumatra

Download

Sumatra can be downloaded from the metabarcoding.org GitLab. The archive of the latest tagged version can be downloaded on the GitLab wiki page:

https://git.metabarcoding.org/obitools/sumatra/wikis/home

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

Installation

Untar the archive, go into the newly created directory and compile:

```
tar –zxvf sumatra_v[x.x.xx].tar.gz
cd sumatra_v[x.x.xx]
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. The output can then go through a classification process with programs such as MCL or MOTHUR.

Using Sumatra

Input

If there is one dataset, the input can be either the standard input (stdin), or a file in FASTA format. If there are two datasets to compare, the input must be two files 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 n
ormalized and expressed in distance, it is (1.0 - identity), e.g. 0.05 for an iden
tity 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 expr
essed 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 Sumatra works

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

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 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 uses 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 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, the program can be run on multiple threads.