NAME

swarm — find clusters of nearly-identical nucleotide amplicons

SYNOPSIS

swarm [*options*] [*filename*]

DESCRIPTION

Environmental or clinical molecular studies generate large volumes of amplicons (e.g. SSU-rRNA sequences) that need to be clustered into molecular operational taxonomic units. Traditional clustering methods are based on greedy, input-order dependent algorithms, with arbitrary selection of global cluster size and cluster centroids. To address that problem, we developed **swarm**, a fast and robust method that recursively groups amplicons with *d* or less differences. **swarm** produces stable clusters (or "swarms"), free from centroid selection induced input-order dependency.

Exact clustering is impractical on large data sets when using a naïve all-vs-all approach (i.e. a 2-combination without repetitions), as it implies unrealistic numbers of pairwise comparisons. **swarm** is based on a maximum number of differences, and focuses only on close relationships. An efficient *k*-mer-based filtering and an astute use of comparisons results obtained during the process allows to avoid most of the amplicon comparisons needed in a naïve approach. To speed up the remaining amplicon comparisons, **swarm** implements an extremely fast Needleman-Wunsch algorithm making use of the Streaming SIMD Extensions (SSE2) of modern x86-64 CPUs. If SSE2 instructions are not available, **swarm** exits with an error message.

swarm reads the named input *filename*, a fasta file of nucleotide amplicons. The amplicon identifier is defined as the string comprised between the ">" symbol and the first space or the end of the line, whichever comes first. As **swarm** outputs lists of amplicon identifiers, amplicon identifiers must be unique to avoid ambiguity; swarm exits with an error message if identifiers are not unique. If amplicon identifiers end with a "_" followed by a number, that number is used as the amplicon copy number in the statistics output file (see option -z for usearch users). The amplicon sequence is defined as a string of [acgt] or [acgu] symbols (case insensitive), starting after the end of the identifier line and ending before the next identifier line or the file end; **swarm** exits with an error message if any other symbol is present. Default is to read from standard input if no file is named, or if the file name is "-".

Options

swarm recognizes the following command-line options:

-a, **--alternative-algorithm**

use an alternative and faster clustering algorithm, only usable when d = 1. That new algorithm produces exactly the same results than the standard swarm algorithm (for d = 1). When combined with the --no-valley option, the OTU delineation and breaking are performed in one-pass.

-b, **--break-swarms**

output all pairs of nearly-identical amplicons to standard error using a four-columns tab-delimited format:

1. "@@", a distinctive pattern to facilitate parsing.
2. amplicon A label.
3. amplicon B label.
4. number of differences between amplicons A and B (*positive integer*).

That option, designed to work with the companion script swarm_breaker.py, can also be used for swarms post-processing, network analysis and visualization.

-d, **--differences** *positive integer*

maximum number of differences allowed between two amplicons, meaning that two amplicons will be grouped if they have *integer* (or less) differences. This is **swarm**'s most important parameter. The number of differences is calculated as the number of mismatches (substitutions, insertions or deletions) between the two amplicons once the optimal pairwise global alignment has been found (see "advanced options" for

parameters influencing the pairwise alignment). Any *integer* between 1 and 256 can be used, but aligning correctly two very distant amplicons is difficult and results should be considered with caution. Default number of differences is 1.

-h, **--help** display this help and exit.

-i, **--internal-structure** *filename*
output all pairs of nearly-identical amplicons to *filename* using a five-columns tab-delimited format:

1. amplicon A label.
2. amplicon B label.
3. number of differences between amplicons A and B (*positive integer*).
4. OTU number (*positive integer*). OTUs are numbered in their order of delineation, starting from 1. All pairs of amplicons belonging to the same OTU will receive the same number.
5. number of steps from the OTU seed to amplicon B (*positive integer*).

That option will replace the option --break-swarms in future versions of swarm.

-l, **--log** *filename*
output all swarm messages to *filename* instead of stderr, with the exception of error messages of course. That option is useful in situations where using stderr is problematic (e.g. some job schedulers abort processes as soon as they write to stderr).

-n, **--no-valley**
when using --alternative-algorithm, use amplicon abundance values to prevent the formation of valleys (decreasing-increasing abundance values along a path). The option forbids the creation of a link between amplicon A and B if the abundance of B is higher than the abundance of A. In practice, that option acts as a built-in OTU breaking step, eliminating the need to use the companion script swarm_breaker.py. Please note that the built-in breaking is stricter than the OTU breaking script, and produces higher-resolution results.

-o, **--output-file** *filename*
output result to *filename*. Result is a list of swarms, one swarm per line. A swarm is a list of amplicon identifiers separated by spaces. Default is to write to standard output.

-r, **--mothur**
output results in a format compatible with Mothur. That option modifies swarm default output format.

-s, **--statistics-file** *filename*
output statistics to the specified file. Default is not to output statistics. The file is a tab-separated table with one swarm per row and seven columns of information: number of unique amplicons in the swarm, total copy number of amplicons in the swarm, identifier of the initial seed, initial seed copy number (if applicable), number of singletons (amplicons with a copy number of 1), maximum number of generations (i.e. numbers of iterations before the swarm reached its natural limits), and the maximum radius of the swarm (i.e. number of differences between the seed and the furthermost amplicon in the swarm).

-t, **--threads** *positive integer*
number of computation threads to use. The number of threads should be lesser or equal to the number of available CPU cores. Default number of threads is 1.

-u, **--uclust-file** *filename*
output results in uclust-like file format to the specified file. That option does not modify swarm default output format.

-v, **--version**
output version information and exit.

-z, **--usearch_abundance**
allows amplicon abundances to be specified using the usearch style in the sequence header (e.g. ">label;size=1").

Advanced options

swarm recognizes advanced command-line options modifying the pairwise global alignment scoring parameters:

-m, **--match-reward** *positive integer*
reward for a nucleotide match. Default is 5.

-p, **--mismatch-penalty** *positive integer*
penalty for a nucleotide mismatch. Default is 4.

-g, **--gap-opening-penalty** *positive integer*
gap open penalty. Default is 12.

-e, **--gap-extension-penalty** *positive integer*
gap extension penalty. Default is 4.

As **swarm** focuses on close relationships, final results are resilient to model parameters modifications. Modifying model parameters only impacts analysis using a high number of differences.

EXAMPLES

swarm -a -n -t 4 -o *myfile.swarms myfile.fasta*
Divide the data set *myfile.fasta* into swarms with the finest resolution possible (1 difference, built-in breaking) using 4 computation threads and the fast algorithm. OTUs are written to the file *myfile.swarms*.

zcat file.fas.gz | **swarm** | awk "{print NF}" | sort -n | uniq -c
Use swarm in a pipeline to read a compressed fasta file and to get its swarm size profile (with default parameters).

AUTHORS

Concept by Frédéric Mahé, implementation by Torbjørn Rognes.

CITATION

Mahé F, Rognes T, Quince C, de Vargas C, Dunthorn M. (2014) Swarm: robust and fast clustering method for amplicon-based studies. *PeerJ* 2:e593 <http://dx.doi.org/10.7717/peerj.593>

REPORTING BUGS

Submit suggestions and bug-reports at <https://github.com/torognes/swarm/issues>, send a pull request on <https://github.com/torognes/swarm>, or compose a friendly or curmudgeont e-mail to Frédéric Mahé <mahe@rhrk.uni-kl.de> and Torbjørn Rognes <torognes@ifi.uio.no>.

AVAILABILITY

The software is available from <https://github.com/torognes/swarm>

COPYRIGHT

SEE ALSO

swipe, an extremely fast Smith-Waterman database search tool by Torbjørn Rognes (available from <https://github.com/torognes/swipe>).

VERSION HISTORY

New features and important modifications of **swarm** (short lived or minor bug releases are not mentioned):

v1.2.20 released November 6, 2014

Version 1.2.20 presents a production-ready version of the alternative algorithm (option -a), with optional built-in OTU breaking (option -n). That alternative algorithmic approach (usable only with d = 1) is considerably faster than currently used clustering algorithms, and can deal with datasets of 100 million unique amplicons or more in a few hours. Of course, results are rigourously identical to the results previously produced with swarm. That release also introduces new options to control swarm output (options -i and -l).

v1.2.19 released October 3, 2014

Version 1.2.19 fixes a problem related to abundance information when the sequence identifier includes multiple underscore characters.

v1.2.18 released September 29, 2014

Version 1.2.18 reenables the possibility of reading sequences from stdin if no file name is specified on the command line. It also fixes a bug related to cpu features detection.

v1.2.17 released September 28, 2014

Version 1.2.17 fixes a memory allocation bug introduced in version 1.2.15.

v1.2.16 released September 27, 2014

Version 1.2.16 fixes a bug in the abundance sort introduced in version 1.2.15.

v1.2.15 released September 27, 2014

Version 1.2.15 sorts the input sequences in order of decreasing abundance unless they are detected to be sorted already. When using the alternative algorithm for d=1 it also sorts all subseeds in order of decreasing abundance.

v1.2.14 released September 27, 2014

Version 1.2.14 fixes a bug in the output with the swarm_breaker option (-b) when using the alternative algorithm (-a).

v1.2.12 released August 18, 2014

Version 1.2.12 introduces an option --alternative-algorithm to use an extremely fast, experimental clustering algorithm for the special case d = 1. Multithreading scalability of the default algorithm has been noticeably improved.

v1.2.10 released August 8, 2014

allows amplicon abundances to be specified using the usearch style in the sequence header (e.g. ">id;size=1") when the -z option is chosen.

v1.2.8 released August 5, 2014

swarm 1.2.8 fixes an error with the gap extension penalty. Previous versions used a gap penalty twice as large as intended. That bug correction induces small changes in clustering results.

v1.2.6 released May 23, 2014

Version 1.2.6 introduces an option --mothur to output swarm results in a format compatible with the microbial ecology community analysis software suite Mothur.

v1.2.5 released April 11, 2014

Version 1.2.5 removes the need for a POPCNT hardware instruction to be present. Swarm now automatically checks whether POPCNT is available and uses a slightly slower software implementation if not. Only basic SSE2 instructions are now required to run swarm.

v1.2.4 released January 30, 2014

Version 1.2.4 introduces an option --break-swarms to output all pairs of amplicons with *d* differences to standard error. That option is used by the companion script ‘swarm_breaker.py‘ to refine swarm results. The syntax of the inline assembly code is changed for compatibility with more compilers.

v1.2 released May 16, 2013

Version 1.2 greatly improves speed by using alignment-free comparisons of amplicons based on *k*-mer word content. For each amplicon, the presence-absence of all possible 5-mers is computed and recorded in a 1024-bits vector. Vector comparisons are extremely fast and drastically reduce the number of costly pairwise alignments performed by swarm. While remaining exact, swarm 1.2 can be more than 100-times faster than swarm 1.1, when using a single thread with a large set of sequences. The minor version 1.1.1, published just before, adds compatibility with Apple computers, and corrects an issue in the pairwise global alignment step that could lead to sub-optimal alignments.

v1.1 released February 26, 2013

Version 1.1 introduces two new important options: the possibility to output swarming results using the uclust output format, and the possibility to output detailed statistics on each swarms. Swarm 1.1 is also faster: new filterings based on pairwise amplicon sequence lengths and composition comparisons reduce the number of pairwise alignments needed and speed up the swarming.

v1.0 released November 10, 2012

First public release