## 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., 16S or 18S SSU-rRNA sequences) that need to be clustered into molecular operational taxonomic units. Common 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 (more precisely a 2-combination without repetitions), as it implies unrealistic numbers of pairwise comparisons. **swarm** is based on a maximum number of differences *d*, and focuses only on very close relationships. For *d* = 1 (default), swarm uses an algorithm of linear complexity that performs exact-string matching by comparing hash-values. For *d* = 2 or greater, swarm uses an algorithm of quadratic complexity that performs pairwise string comparisons. 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 positive integer, that value is used as the amplicon copy number in the statistics output file (see the option -z to change that behavior for usearch/vsearch 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.

### General options

**-b**, **--boundary** *positive integer*

when using the option --fastidious (-f) , define the minimum mass of a large OTU as the number given with this option. The default value is 3, indicating that any OTU with mass 3 or more is considered large. By default, an OTU is small if it has a mass of 2 or less, meaning that it is composed of either one amplicon of abundance 2, or two amplicons of abundance 1. Any positive value greater than 1 can be specified. Using higher boundary values will speed up the second pass, but also reduce the taxonomical resolution of swarm results.

**-c**, **--ceiling** *positive integer*

when using the option --fastidious (-f), define swarm's maximum memory footprint (in megabytes). Swarm will adjust the --bloom-bits (-y) value of the Bloom filter to fit within the specified amount of memory. That option is not active by default.

**-d**, **--differences** *zero or 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 0 and 256 can be used, but aligning correctly two very distant amplicons is difficult and results should be considered with caution. Using high *d* values will also decrease the taxonomical resolution of swarm results. When using *d* = 0, swarm will output results corresponding to a strict dereplication of the dataset. Default number of differences is 1.

**-f**, **--fastidious**
when working with $d = 1$, perform a second clustering pass to reduce the number of small OTUs (recommended option). During the stone stepping process at $d = 1$, a step can be missing for purely stochastic reasons, interrupting the agglomerative process. That option will create virtual steps to graft small OTUs upon bigger ones. By default, an OTU is small if it has a mass of 2 or less (see the --boundary option to increase that value). To speed things up, swarm uses a Bloom filter to store intermediate results. Warning, that second pass can be 2 to 3 times slower than the first pass and requires much more memory. See the options --bloom-bits (-y) or --ceiling (-c) to control the memory footprint of the Bloom filter. Warning, the fastidious option modifies OTUs: output files --log (-l), --output-file (-o), and --seeds (-w) are updated to reflect these modifications; the file --statistics-file (-s) is partially updated (columns 6 and 7 are not updated); output files --internal-structure (-i), --mothur (-r), and --uclust-file (-u) are not updated.

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

**-n**, **--no-otu-breaking**
deactivate the built-in OTU refinement (not recommended). Amplicon abundance values are used to identify transitions among in-contact OTUs and to separate them, yielding higher-resolution clustering results. That option prevents that, and in practice, allows the creation of a link between amplicon A and B, even if the abundance of B is higher than the abundance of A.

**-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.

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

**-y**, **--bloom-bits** *positive integer*
when using the option --fastidious (-f), define the size (in bits) of each entry in the Bloom filter. That option allows to balance the efficiency (i.e. speed) and the memory footprint of the Bloom filter. Large values will make the Bloom filter more efficient but will require more memory. Any value between 4 and 20 can be used. Default value is 16. See the --ceiling (-c) option for an alternative way to control the memory footprint.

**Input/output options**

**-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*).

**-l**, **--log** *filename*
output all messages to *filename* instead of *standard error*, with the exception of error messages of course. That option is useful in situations where writing to *standard error* is problematic (for example, with certain job schedulers).

**-o**, **--output-file** *filename*
output result to *filename*. Result is a list of OTUs, one OTU per line. A OTU 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's default output format.

**-s**, **--statistics-file** *filename*
output statistics to the specified file. The file is a tab-separated table with one OTU per row and seven columns of information: number of unique amplicons in the OTU, total copy number of amplicons in the OTU, 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 OTU reached its natural limits), and the theoretical maximum radius of the OTU (i.e., number of cummulated differences between the seed and the furthermost amplicon in the OTU. The actual pairwise distance and max radius is often much smaller).

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

**-w**, **--seeds** *filename*
output OTU representatives to *filename* in fasta format. The abundance value of each representative is the sum of the abundances of all the amplicons in the OTU.

**-z**, **--usearch-abundance**
accept amplicon abundances specified using the usearch/vsearch style (">label;size=*integer*[;]"). That option influences the annotation style used in output files.

**Pairwise alignment options**
when using $d > 1$, **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

Divide the data set *myfile.fasta* into OTUs with the finest resolution possible (1 difference, built-in breaking, fastidious option) using 4 computation threads. OTUs are written to the file *myfile.swarms*.

**swarm** -t 4 -f -o *myfile.swarms myfile.fasta*

## 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 curmudgeonly 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

Copyright (C) 2012, 2013, 2014, 2015 Frédéric Mahé & Torbjørn Rognes

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

## SEE ALSO

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

**vsearch**, an open-source re-implementation of the classic uclust clustering method (by Robert C. Edgar), along with other amplicon filtering and searching tools. **vsearch** is implemented by Torbjørn Rognes and documented by Frédéric Mahé, and is available at <https://github.com/torognes/vsearch>.

## VERSION HISTORY

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

**v2.1.0** released March 24, 2015
: Version 2.1.0 marks the first official release of swarm 2.

**v2.0.7** released March 18, 2015
: Version 2.0.7 writes abundance information in usearch style when using options -w (--seeds) in combination with -z (--usearch-abundance).

**v2.0.6** released March 13, 2015
: Version 2.0.6 fixes a minor bug.

**v2.0.5** released March 13, 2015
: Version 2.0.5 improves the implementation of the fastidious option and adds options to control memory usage of the Bloom filter (-y and -c). In addition, an option (-w) allows to output OTU representatives sequences with updated abundances (sum of all abundances inside each OTU). This version also enables dereplication when $d = 0$.

**v2.0.4** released March 6, 2015
: Version 2.0.4 includes a fully parallelised implementation of the fastidious option.

**v2.0.3** released March 4, 2015
: Version 2.0.3 includes a working implementation of the fastidious option, but only the initial clustering is parallelized.

**v2.0.2** released February 26, 2015
: Version 2.0.2 fixes SSSE3 problems.

**v2.0.1** released February 26, 2015
: Version 2.0.1 is a development version that contains a partial implementation of the fastidious option, but it is not usable yet.

**v2.0.0** released December 3, 2014
: Version 2.0.0 is faster and easier to use, providing new output options (--internal-structure and --log), new control options (--boundary, --fastidious, --no-otu-breaking), and built-in OTU refinement. When using default parameters, a novel and considerably faster algorithmic approach is used, guaranteeing swarm's scalability.

**v1.2.21** released February 26, 2015
Version 1.2.21 is supposed to fix some problems related to the use of the SSSE3 CPU instructions which are not always available.

**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.