NAME

swarm — find clusters of nearly-identical nucleotide amplicons

SYNOPSIS

swarm -h|v

High-precision clustering:

swarm [*filename*]

swarm [**-d** 1] [**-nrz**] [**-a** *int*] [**-i** *filename*] [**-j** *filename*] [**-l** *filename*] [**-o** *filename*] [**-s** *filename*] [**-t** *int*] [**-u** *filename*] [**-w** *filename*] [*filename*]

swarm [**-d** 1] **-f** [**-nrz**] [**-a** *int*] [**-b** *int*] [**-c|y** *int*] [**-i** *filename*] [**-j** *filename*] [**-l** *filename*] [**-o** *filename*] [**-s** *filename*] [**-t** *int*] [**-u** *filename*] [**-w** *filename*] [*filename*]

Conservative clustering:

swarm -d **2+** [**-nrxz**] [**-a** *int*] [**-e** *int*] [**-g** *int*] [**-i** *filename*] [**-l** *filename*] [**-m** *int*] [**-o** *filename*] [**-p** *int*] [**-s** *filename*] [**-t** *int*] [**-u** *filename*] [**-w** *filename*] [*filename*]

Dereplication (merge strictly identical sequences):

swarm -d **0** [**-rz**] [**-a** *int*] [**-i** *filename*] [**-l** *filename*] [**-o** *filename*] [**-s** *filename*] [**-u** *filename*] [**-w** *filename*] [*filename*]

DESCRIPTION

Environmental or clinical molecular studies generate large volumes of amplicons (e.g., 16S or 18S SSU-rRNA sequences) that need to be grouped into clusters. Traditional clustering methods are based on greedy, input-order dependent algorithms, with arbitrary selection of cluster centroids and cluster limits (often 97%-similarity). To address that problem, we developed **swarm**, a fast and robust method that recursively groups amplicons with *d* or less differences (i.e. substitutions, insertions or deletions). **swarm** produces natural and stable clusters centered on local peaks of abundance, mostly free from input-order dependency induced by centroid selection.

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* between two amplicons, and focuses only on very close local relationships. For $d = 1$, the default value, **swarm** uses an algorithm of linear complexity that generates all possible single mutations and 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 clustering process allows **swarm** 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 x86-64 CPUs, NEON instructions of ARM64 CPUs, or Altivec/VMX instructions of POWER8 CPUs. If SSE2 instructions are not available, **swarm** exits with an error message.

swarm can read nucleotide amplicons in fasta format from a normal file or from the standard input (using a pipe or a redirection). The amplicon *header* is defined as the string comprised between the '>' symbol and the first space or the end of the line, whichever comes first. Each header must end with an *abundance annotation* representing the amplicon copy number and defined as '_' followed by a positive integer. See option -z for input data using usearch/vsearch's abundance annotation format (';size=*integer*[;]'). Once stripped from the abundance annotation, the remaining part of the header is call the *label*. In summary, using regular expression patterns:

`>header[[:blank:]]` and `header = label_[1-9][0-9]*$`

Abundance annotations play a crucial role in the clustering process, and swarm exits with an error message if that information is not available. As **swarm** outputs lists of amplicon labels, amplicon labels must be unique to avoid any ambiguity; swarm exits with an error message if labels are not unique. The amplicon sequence is defined as a string of [ACGT] or [ACGU] symbols (case insensitive, 'U' is replaced with 'T' internally), starting after the end of the header line and ending before the next header line or the file end;

swarm silently removes newline symbols ('\n' or '\r') and exits with an error message if any other symbol is present. Accepted sequence lengths range from 1 nucleotide to 67 million nucleotides. Please note that processing 67-Mb sequences requires at least 32 gigabytes of memory. Lastly, if sequences are not all unique, i.e. were not properly dereplicated, swarm will exit with an error message.

Clusters are written to output files (specified with -i, -o, -s and -u) by decreasing abundance of their seed sequences, and then by alphabetical order of seed sequence labels. An exception to that is the -w (--seeds) output, which is sorted by decreasing *cluster abundance* (sum of abundances of all sequences in the cluster), and then by alphabetical order of seed sequence labels. This is particularly useful for post-clustering steps, such as *de novo* chimera detection, that require clusters to be sorted by decreasing abundances.

General options

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

-t, **--threads** *positive integer*
number of computation threads to use. Values between 1 and 512 are accepted, but we recommend to use a number of threads lesser or equal to the number of available CPU cores. Default number of threads is 1.

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

-- delimit the option list. Later arguments, if any, are treated as operands even if they begin with '-'. For example, 'swarm -- -file.fasta' reads from the file '-file.fasta'.

Clustering options

-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 'pairwise alignment advanced options' to influence that step).

Any *integer* from 0 to 255 can be used, but high *d* values will decrease the taxonomical resolution of **swarm** results. Commonly used *d* values are 1, 2 or 3, rarely higher. When using $d = 0$, **swarm** will output results corresponding to a strict dereplication of the dataset, i.e. merging identical amplicons. Warning, whatever the *d* value, **swarm** requires fasta entries to present abundance values. Default number of differences *d* is 1.

-n, **--no-otu-breaking**
when working with $d = 1$, deactivate the built-in cluster refinement (not recommended). Amplicon abundance values are used to identify transitions among in-contact clusters and to separate them, yielding higher-resolution clustering results. That option prevents that separation, and in practice, allows the creation of a link between amplicons A and B, even if the abundance of B is higher than the abundance of A.

Fastidious options

-b, **--boundary** *positive integer*
when using the option --fastidious (-f), define the minimum abundance of what should be considered a *large* cluster. By default, a cluster with a total abundance of 3 or more is considered large. Conversely, a cluster is small if it has a total abundance of 2 or less, meaning that it is composed of either one amplicon of abundance 2, or two amplicons of abundance 1, or one amplicon of abundance 1. Any positive value greater than 1 can be specified. Using higher boundary values can reduce the number of clusters (up to a point), and will reduce the taxonomical resolution of **swarm** results. It will also slightly increase computation time.

-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. Values accepted range from 40 to 1,073,741,824 megabytes. See

the --bloom-bits (-y) option for an alternative way to control the memory footprint.

-f, **--fastidious**

when working with *d* = 1, perform a second clustering pass to reduce the number of small clusters (recommended option). During the first clustering pass, an intermediate amplicon can be missing for purely stochastic reasons, interrupting the aggregation process. The fastidious option will create virtual amplicons, allowing to graft small clusters upon larger ones. By default, a cluster is considered large if it has a total abundance of 3 or more (see the --boundary option to modify that value).

To speed things up, **swarm** uses a Bloom filter to store intermediate results. Warning, the second clustering pass can be 2 to 3 times slower than the first pass and requires much more memory to store the virtual amplicons in Bloom filters. See the options --bloom-bits (-y) or --ceiling (-c) to control the memory footprint of the Bloom filter.

The fastidious option modifies clustering results: the output files produced by the options --log (-l), --output-file (-o), --mothur (-r), --uclust-file, and --seeds (-w) are updated to reflect these modifications; the file --statistics-file (-s) is partially updated (columns 6 and 7 are not updated); the output file --internal-structure (-i) is partially updated (column 5 is not updated for amplicons that belonged to the small cluster).

-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 2 and 64 can be used. Default value is 16. See the --ceiling (-c) option for an alternative way to control the memory footprint.

Input/output options

-a, **--append-abundance** *positive integer*

set abundance value to use when some or all amplicons in the input file lack abundance values (_*integer*, or ;size=*integer*; when using -z). Warning, it is not recommended to use **swarm** on datasets where abundance values are all identical. We provide that option as a courtesy to advanced users, please use it carefully. **swarm** exits with an error message if abundance values are missing and if this option is not used.

-i, **--internal-structure** *filename*

output all pairs of nearly-identical amplicons to *filename* using a five-column tab-delimited format:

1. amplicon A label (header without abundance annotations).
2. amplicon B label (header without abundance annotations).
3. number of differences between amplicons A and B (*positive integer*).
4. cluster number (*positive integer*). Clusters are numbered in their order of delineation, starting from 1. All pairs of amplicons belonging to the same cluster will receive the same number.
5. cummulated number of steps from the cluster seed to amplicon B (*positive integer*). When using the option --fastidious (-f), the actual number of steps between grafted amplicons and the cluster seed cannot be re-computed efficiently and is always set to 2 for the amplicon pair linking the small cluster to the large cluster. Cummulated number of steps in the small cluster (if any) are left unchanged.

-j, **--network-file** *filename*

(advanced users) when working with *d* = 1, dump raw amplicon network to *filename* using a two-column tab-delimited table of headers with abundance annotations. Each line represents a connection between two similar amplicons, from the most abundant to the lesser abundant.

When amplicons have the same abundance value, connections are bi-directional and are represented on two lines: A to B, then B to A.

In order to delineate clusters and to compute the equivalent of a minimal spanning tree for each cluster (see option --internal-structure), swarm first builds a network of similar amplicons. This option is for advanced users who would like to explore this raw network.

-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 clustering results to *filename*. Results consist of a list of clusters, one cluster per line. A cluster is a list of amplicon headers separated by spaces. That output format can be modified by the option --mothur (-r). Default is to write to *standard output*.

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

-s, **--statistics-file** *filename*
output statistics to *filename*. The file is a tab-separated table with one cluster per row and seven columns of information:

1. number of unique amplicons in the cluster,
2. total abundance of amplicons in the cluster,
3. label of the initial seed (header without abundance annotations),
4. abundance of the initial seed,
5. number of amplicons with an abundance of 1 in the cluster,
6. maximum number of iterations before the cluster reached its natural limit,
7. cummulated number of steps along the path joining the seed and the furthermost amplicon in the cluster. Please note that the actual number of differences between the seed and the furthermost amplicon is usually much smaller. When using the option --fastidious (-f), grafted amplicons are not taken into account.

-u, **--uclust-file** *filename*
output clustering results in *filename* using a tab-separated uclust-like format with 10 columns and 3 different type of entries (S, H or C). That option does not modify **swarm**'s default output format. Each fasta sequence in the input file can be either a cluster centroid (S) or a hit (H) assigned to a cluster. Cluster records (C) summarize information for each cluster (number of hits, centroid header). Column content varies with the type of entry (S, H or C):

1. Record type: S, H, or C.
2. Cluster number (zero-based).
3. Centroid length (S), query length (H), or number of hits (C).
4. Percentage of similarity with the centroid sequence (H), or set to '*' (S, C).
5. Match orientation + or - (H), or set to '*' (S, C).
6. Not used, always set to '*' (S, C) or to zero (H).
7. Not used, always set to '*' (S, C) or to zero (H).
8. set to '*' (S, C) or, for H, compact representation of the pairwise alignment using the CIGAR format (Compact Idiosyncratic Gapped Alignment Report): M (match), D (deletion) and I (insertion). The equal sign '=' indicates that the query

is identical to the centroid sequence.

9. Header of the query sequence (H), or of the centroid sequence (S, C).
10. Header of the centroid sequence (H), or set to '*' (S, C).

-w, **--seeds** *filename*
output cluster representative sequences to *filename* in fasta format. The abundance value of each cluster representative is the sum of the abundances of all the amplicons in the cluster. Fasta headers are formated as follows: '>label_*integer*', or '>label;size=*integer*;' if the -z option is used, and sequences are uppercased. Sequences are sorted by decreasing abundance, and then by alphabetical order of sequence labels.

-z, **--usearch-abundance**
accept amplicon abundance values in usearch/vsearch's style (>label;size=*integer*[;]). That option influences the abundance annotation style used in swarm's *standard output* (-o), as well as the output of options -r, -u and -w.

Pairwise alignment advanced options
when using $d > 1$, **swarm** recognizes advanced command-line options modifying the pairwise global alignment scoring parameters:

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

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

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

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

-x, **--disable-sse3**
On the x86-64 CPU architecture, disable SSE3 and later instructions. This option is meant for developers, not for regular users.

As **swarm** focuses on close relationships (e.g., $d = 2$ or 3), clustering results are resilient to pairwise alignment model parameters modifications. When clustering using a higher d value, modifying model parameters has a stronger impact.

EXAMPLES

Clusterize the compressed data set *myfile.fasta* using the finest resolution possible (1 difference by default, built-in breaking, fastidious option) using 4 computation threads. Clusters are written to the file *myfile.swarms*, and cluster representatives are written to *myfile.representatives.fasta*:

```
zcat myfile.fasta.gz | \
    swarm \
        -t 4 \
        -f \
        -w myfile.representatives.fasta \
        -o myfile.swarms
```

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 ⟨https://doi.org/10.7717/peerj.593⟩.

Mahé F, Rognes T, Quince C, de Vargas C, Dunthorn M. (2015) Swarm v2: highly-scalable and high-resolution amplicon clustering. *PeerJ* 3:e1420 ⟨https://doi.org/10.7717/peerj.1420⟩.

Mahé F, Czech L, Stamatakis A, Quince C, de Vargas C, Dunthorn M, Rognes T. (2021) Swarm v3:

towards tera-scale amplicon clustering. *Bioinformatics* ⟨https://doi.org/10.1093/bioinformatics/btab493⟩.

REPORTING BUGS

Submit suggestions and bug-reports at ⟨https://github.com/torognes/swarm/issues⟩, send a pull request at ⟨https://github.com/torognes/swarm/pulls⟩, or compose a friendly or curmudgeonly e-mail to Frédéric Mahé ⟨frederic.mahe@cirad.fr⟩ and Torbjørn Rognes ⟨torognes@ifi.uio.no⟩.

AVAILABILITY

Source code and binaries available at ⟨https://github.com/torognes/swarm⟩.

COPYRIGHT

SEE ALSO

swipe, an extremely fast Smith-Waterman database search tool by Torbjørn Rognes (available at ⟨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):

v3.1.5 released March 31, 2024

Version 3.1.5 changes the minimal value for the ceiling option from 8 megabytes to 40 megabytes, and fixes four minor bugs. Warning, peak RSS memory increased by 5 to 10% when $d >= 2$. Version 3.1.5 improves documentation (now covering option --network_file), adds more compilation checks and eliminates 50 compilation warnings with GCC 13, GCC 14 and clang 19, as well as 1,677 static analysis warnings.

v3.1.4 released September 20, 2023

Version 3.1.4 fixes a minor bug. It eliminates compilation warnings with GCC 13 and clang 18, as well as 1,040 static analysis warnings. The maximal number of threads swarm can run is now 512, instead of 256. Compilation with runtime checks ('-DNDEBUG') is now the default. When d > 1, overall memory allocations remain unchanged, but peak RSS memory increased by 6 to 10%, due to a change in the timing of memory deallocations. Peak RSS memory is expected to regress to its prior levels as refactoring continues.

v3.1.3 released December 5, 2022

Version 3.1.3 fixes a regression introduced in version 3.1.1 (memory over-allocation when d > 1). It also fixes a minor off-by-one error when allocating memory for a Bloom filter, compilation warnings with GCC 12 and clang 13, as well as static analysis warnings. Documentation was improved, as well as our test suite (swarm-tests).

v3.1.2 released November 10, 2022

Fix a bug with fastidious mode introduced in version 3.1.1, that could cause Swarm to crash. Probably due to allocating too much memory.

v3.1.1 released September 29, 2022
Version 3.1.1 eliminates a risk of segmentation fault with extremely long sequence headers. Documentation and error messages have been improved, and code cleaning continued.

v3.1.0 released March 1, 2021
Version 3.1.0 includes a fix for a bug in the 16-bit SIMD alignment code that was exposed with a combination of d>1, long sequences, and very high gap penalties. The code has also been been cleaned up, tested and improved substantially, and it is now fully C++11 compliant. Support for macOS on Apple Silicon (ARM64) has been added.

v3.0.0 released October 24, 2019
Version 3.0.0 introduces a faster algorithm for $d = 1$, and a reduced memory footprint. Swarm has been ported to Windows x86-64, GNU/Linux ARM 64, and GNU/Linux POWER8. Internal code has been modernized, hardened, and thoroughly tested. Strict dereplication of input sequences is now mandatory. The --seeds option (-w) now outputs results sorted by decreasing abundance, and then by alphabetical order of sequence labels.

v2.2.2 released December 12, 2017
Version 2.2.2 fixes a bug that would cause swarm to wait forever in very rare cases when multiple threads were used.

v2.2.1 released October 27, 2017
Version 2.2.1 fixes a memory allocation bug for $d = 1$ and duplicated sequences.

v2.2.0 released October 17, 2017
Version 2.2.0 fixes several problems and improves usability. Corrected output to structure and uclust files when using fastidious mode. Corrected abundance output in some cases. Added check for duplicated sequences and fixed check for duplicated sequence IDs. Checks for empty sequences. Sorts sequences by additional fields to improve stability. Improves compatibility with compilers and operating systems. Outputs sequences in upper case. Allows 64-bit abundances. Shows message when waiting for input from stdin. Improves error messages and warnings. Improves checking of command line options. Fixes remaining errors reported by test suite. Updates documentation.

v2.1.13 released March 8, 2017
Version 2.1.13 removes a bug with the progress bar when writing seeds.

v2.1.12 released January 16, 2017
Version 2.1.12 removes a debugging message.

v2.1.11 released January 16, 2017
Version 2.1.11 fixes two bugs related to the SIMD implementation of alignment that might result in incorrect alignments and scores. The bug only applies when $d > 1$.

v2.1.10 released December 22, 2016
Version 2.1.10 fixes two bugs related to gap penalties of alignments. The first bug may lead to wrong aligments and similarity percentages reported in UCLUST (.uc) files. The second bug makes swarm use a slightly higher gap extension penalty than specified. The default gap extension penalty used have actually been 4.5 instead of 4.

v2.1.9 released July 6, 2016
Version 2.1.9 fixes errors when compiling with GCC version 6.

v2.1.8 released March 11, 2016
Version 2.1.8 fixes a rare bug triggered when clustering extremely short undereplicated sequences. Also, alignment parameters are not shown when $d = 1$.

v2.1.7 released February 24, 2016
Version 2.1.7 fixes a bug in the output of seeds with the -w option when $d > 1$ that was not properly fixed in version 2.1.6. It also handles ascii character #13 (CR) in FASTA

files better. Swarm will now exit with status 0 if the -h or the -v option is specified. The help text and some error messages have been improved.

v2.1.6 released December 14, 2015
Version 2.1.6 fixes problems with older compilers that do not have the x86intrin.h header file. It also fixes a bug in the output of seeds with the -w option when $d > 1$.

v2.1.5 released September 8, 2015
Version 2.1.5 fixes minor bugs.

v2.1.4 released September 4, 2015
Version 2.1.4 fixes minor bugs in the swarm algorithm used for $d = 1$.

v2.1.3 released August 28, 2015
Version 2.1.3 adds checks of numeric option arguments.

v2.1.1 released March 31, 2015
Version 2.1.1 fixes a bug with the fastidious option that caused it to ignore some connections between large and small clusters.

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

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 cluster representatives sequences with updated abundances (sum of all abundances inside each cluster). This version also enables **swarm** to run with $d = 0$.

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 cluster refinement (no need to use the python script anymore). 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 cluster 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 label 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
Version 1.2.10 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
Version 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 clustering results in a format compatible with the microbial ecology community analysis software suite Mothur (⟨https://www.mothur.org/⟩).

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 clustering results using the uclust output format, and the possibility to output detailed statistics on each cluster. **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 clustering.

v1.0 released November 10, 2012

First public release.