NAME

vsearch — chimera detection, clustering, dereplication, masking, pairwise alignment, searching, shuffling and sorting of amplicons from metagenomic projects.

SYNOPSIS

Chimera detection:

vsearch --uchime_denovo *fastafile* (--chimeras | --nonchimeras | --uchimealns | --uchimeout) *outputfile* [*options*]

vsearch --uchime_ref *fastafile* (--chimeras | --nonchimeras | --uchimealns | --uchimeout) *outputfile* --db *fastafile* [*options*]

Clustering:

vsearch (--cluster_fast | --cluster_size | --cluster_smallmem) *fastafile* (--alnout | --blast6out | --centroids | --clusters | --msaout | --samout | --uc | --userout) *outputfile* --id *real* [*options*]

Dereplication:

vsearch --derep_fulllength *fastafile* (--output | --uc) *outputfile* [*options*]

Masking:

vsearch --maskfasta *fastafile* --output *outputfile* [*options*]

Pairwise alignment:

vsearch --allpairs_global *fastafile* (--alnout | --blast6out | --matched | --notmatched | --samout | --uc | --userout) *outputfile* (--acceptall | --id *real*) [*options*]

Searching:

vsearch --usearch_global *fastafile* --db *fastafile* (--alnout | --blast6out | --samout | --uc | --userout) *outputfile* --id *real* [*options*]

Shuffling:

vsearch --shuffle *fastafile* --output *outputfile* [*options*]

Sorting:

vsearch (--sortbylength | --sortbysize) *fastafile* --output *outputfile* [*options*]

DESCRIPTION

Environmental or clinical molecular diversity studies generate large volumes of amplicons (e.g. SSU-rRNA sequences) that need to be checked for chimeras, dereplicated, masked, sorted, searched, clustered or compared to reference sequences. The aim of **vsearch** is to offer a all-in-one open source tool to perform these tasks, using optimized algorithm implementations and harvesting the full potential of modern computers, thus providing fast and accurate data processing.

Comparing nucleotide sequences is at the core of **vsearch**. To speed up comparisons, **vsearch** implements an extremely fast implementation of the Needleman-Wunsch algorithm, making use of the Streaming SIMD Extensions (SSE2) of modern x86-64 CPUs. If SSE2 instructions are not available, **vsearch** exits with an error message. For comparisons involving sequences longer than 5,000 nucleotides, **vsearch** uses a slower alignment method with smaller memory requirements.

Input

vsearch input is a fasta file containing one or several nucleotide sequences. For each sequence, the sequence identifier is defined as the string comprised between the ">" symbol and the first space, or the end of the line, whichever comes first. Additionally, if the line starts with ">[;]size=*integer*;label", contains ">label;size=*integer*;label" or ends with ">label;size=*integer*[;]", **vsearch** will remove the pattern [;]size=*integer*[;] from the header and interpret *integer* as the number of occurrences (or abundance) of the sequence in the study. That abundance information is used or created during chimera detection, clustering, dereplication, sorting and searching.

The nucleotide sequence is defined as a string of IUPAC symbols (ACGTURYSWKMDBHVN), starting after the end of the identifier line and ending before the next identifier line, or the file end. **vsearch** silently ignores ascii characters 9 to 13, and exits with an error message if ascii characters 0 to 8, 14 to 31, "." or "-" are present. All other ascii or non-ascii characters are stripped and complained about in a non-blocking

warning message.

vsearch operations are case insensitive, except when soft masking is activated. When using clustering, masking or searching commands, the case is important if soft masking is used. Soft masking is specified with the options "--dbmask soft" (for searching) or "--qmask soft" (for searching, clustering and masking). When using soft masking, lower case letters indicate masked symbols, while upper case letters indicate regular symbols. Masked symbols are never included in the unique *k*-mers used in searching. When soft masking is not activated, all letters are converted to upper case internally and used in result files.

When comparing sequences during chimera detection, dereplication, searching and clustering, T and U are considered identical, regardless of their case. If two symbols are not identical, their alignment will result in the negative mismatch score (default -4), except if one or both of the symbols are ambiguous (RYSWKMDBHVN) in which case the score is zero. Alignment of two identical ambiguous symbols (e.g. R vs R) also receives a score of zero.

vsearch can be compiled to accepted compressed fasta files as input (gz and bzip2 formats). On the other hand, special files like pipes, named pipes, or sockets cannot be used as input. To present a progress indicator, **vsearch** needs to seek to the end of *filename* to find its length. Consequently, *filename* must be a regular file, not a stream.

Options

vsearch recognizes a large number of command-line options. For easier navigation, options are grouped below by theme (chimera detection, clustering, dereplication, masking, shuffling, sorting, and searching). We start with general options that apply to all themes.

General options:

--help Display a short help and exit.

--version Output version information and exit.

--fasta_width *positive integer*
Fasta files produced by **vsearch** are wrapped (sequences are written on lines of *integer* nucleotides, 80 by default). Set that value to 0 to eliminate the wrapping.

--maxseqlength *positive integer*
All **vsearch** operations will discard sequences of length equal or greater than *integer* (50,000 nucleotides by default).

--minseqlength *positive integer*
All **vsearch** operations will discard sequences of length smaller than *integer* (1 nucleotide by default for sorting or shuffling, 32 nucleotides for clustering, dereplication or searching).

--notrunclabels
Do not truncate sequence labels at first space, use the full header in output files.

Chimera detection options:

Chimera detection is based on a scoring function controlled by five options (--dn, --mindiffs, --mindiv, --minh, --xn). Sequences are first sorted by decreasing abundance (if available), and compared on their *plus* strand only (case insensitive).

In *de novo* mode, input fasta file should present abundance annotations (pattern [;]size=*integer*[;] in the fasta header). The input order influences the chimera detection, so we recommend to sort sequences by decreasing abundance (default of --derep_fulllength command). If your sequence set needs to be sorted, please see the --sortbysize command in the sorting section.

--abskew *real*
When using --uchime_denovo, the abundance skew is used to distinguish in a 3-way alignment which sequence is the chimera and which are the parents. The assumption is that chimeras appear later in the PCR amplification process and are therefore less abundant than their parents. The default value is 2.0, which means that the parents should be

at least 2 times more abundant than their chimera. Any positive value greater than 1.0 can be used.

--alignwidth *positive integer*
Width of the 3-way alignments in --uchimealns output. The default value is 80. Set to 0 to eliminate wrapping.

--chimeras *filename*
Output chimeric sequences to *filename*, in fasta format. Output order may vary when using multiple threads.

--db *filename*
When using --uchime_ref, detect chimeras using the fasta-formatted reference sequences contained in *filename*. Reference sequences are assumed to be chimera-free. Chimeras will not be detected if their parents (or sufficiently close relatives) are not present in the database.

--dn *real* No vote pseudo-count (parameter *n* in the chimera scoring function) (default value is 1.4).

--mindiffs *positive integer*
Minimum number of differences per segment (default value is 3).

--mindiv *real*
Minimum divergence from closest parent (default value is 0.8).

--minh *real*
Minimum score (h). Increasing this value tends to reduce the number of false positives and to decrease sensitivity. Default value is 0.28, and values ranging from 0.0 to 1.0 included are accepted.

--nonchimeras *filename*
Output non-chimeric sequences to *filename*, in fasta format. Output order may vary when using multiple threads.

--self When using --uchime_ref, ignore a reference sequence when its label matches the label of the query sequence (useful to estimate false-positive rate in reference sequences).

--selfid When using --uchime_ref, ignore a reference sequence when its nucleotide sequence is strictly identical with the query sequence.

--threads *positive integer*
Number of computation threads to use (1 to 256) with --uchime_ref. The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available ressources and to launch one thread per logical core.

--uchime_denovo *filename*
Detect chimeras present in the fasta-formatted *filename*, without external references (i.e. *de novo*). Automatically sort the sequences in *filename* by decreasing abundance beforehand (see the sorting section for details). Multithreading is not supported.

--uchime_ref *filename*
Detect chimeras present in the fasta-formatted *filename* by comparing them with reference sequences (option --db). Multithreading is supported.

--uchimealns *filename*
Write the 3-way global alignments (parentA, parentB, chimera) to *filename* using a human-readable format. Use --alignwidth to modify alignment length. Output order may vary when using multiple threads.

--uchimeout *filename*
Write chimera detection results to *filename* using the uchime tab-separated format of 18 fields (see the list below). Use --uchimeout5 to use a format compatible with

usearch v5 and earlier versions. Rows output order may vary when using multiple threads.

1. score: higher score means a more likely chimeric alignment.
2. Q: query sequence label.
3. A: parent A sequence label.
4. B: parent B sequence label.
5. T: top parent sequence label (i.e. parent most similar to the query). That field is removed when using --uchimeout5.
6. idQM: percentage of similarity of query (Q) and model (M) constructed as a part of parent A and a part of parent B.
7. idQA: percentage of similarity of query (Q) and parent A.
8. idQB: percentage of similarity of query (Q) and parent B.
9. idAB: percentage of similarity of parent A and parent B.
10. idQT: percentage of similarity of query (Q) and top parent (T).
11. LY: yes votes in the left part of the model.
12. LN: no votes in the left part of the model.
13. LA: abstain votes in the left part of the model.
14. RY: yes votes in the right part of the model.
15. RN: no votes in the right part of the model.
16. RA: abstain votes in the right part of the model.
17. div: divergence, defined as (idQM - idQT).
18. YN: query is chimeric (Y), or not (N), or is a borderline case (?).

--uchimeout5
When using --uchimeout, write chimera detection results using a tab-separated format of 17 fields (drop the 5th field of --uchimeout), compatible with usearch version 5 and earlier versions.

--xn *real* No vote weight (parameter beta in the scoring function) (default value is 8.0).

Clustering options:

vsearch implements a single-pass, greedy star-clustering algorithm, similar to the algorithms implemented in usearch, DNAclust and sumaclust for example. Important parameters are the global clustering threshold (--id) and the pairwise identity definition (--iddef).

--centroids *filename*
Output cluster centroid sequences to *filename*, in fasta format. The centroid is the sequence that seeded the cluster (i.e. the first sequence of the cluster).

--cluster_fast *filename*
Clusterize the fasta sequences in *filename*, automatically perform a sorting by decreasing sequence length beforehand.

--cluster_size *filename*
Clusterize the fasta sequences in *filename*, automatically perform a sorting by decreasing sequence abundance beforehand.

--cluster_smallmem *filename*
Clusterize the fasta sequences in *filename* without automatically modifying their order beforehand. Sequence are expected to be sorted by decreasing sequence length, unless --usersort is used.

--clusters *string*
Output each cluster to a separate fasta file using the prefix *string* and a ticker (0, 1, 2, etc.) to construct the path and filenames.

--consout *filename*
Output cluster consensus sequences to *filename*. For each cluster, a multiple alignment is computed, and a consensus sequence is constructed by taking the majority symbol (nucleotide or gap) from each column of the alignment. Columns containing a majority of gaps are skipped, except for terminal gaps.

--id *real* Do not add the target to the cluster if the pairwise identity with the centroid is lower than *real* (value ranging from 0.0 to 1.0 included). The pairwise identity is defined as the number of (matching columns) / (alignment length - terminal gaps). That definition can be modified by --iddef.

--iddef *0|1|2|3|4*
Change the pairwise identity definition used in --id. Values accepted are:

0. CD-HIT definition: (matching columns) / (shortest sequence length).
1. edit distance: (matching columns) / (alignment length).
2. edit distance excluding terminal gaps (same as --id).
3. Marine Biological Lab definition counting each extended gap (internal or terminal) as a single difference: 1.0 - [(mismatches + gaps)/(longest sequence length)]
4. BLAST definition, equivalent to --iddef 2 in a context of global pairwise alignment.

--msaout *filename*
Output a multiple sequence alignment and a consensus sequence for each cluster to *filename*, in fasta format. The consensus sequence is constructed by taking the majority symbol (nucleotide or gap) from each column of the alignment. Columns containing a majority of gaps are skipped, except for terminal gaps.

--qmask *none|dust|soft*
Mask simple repeats and low-complexity regions in sequences using the *dust* or the *soft* algorithms, or do not mask (*none*). Warning, when using *soft* masking, clustering becomes case sensitive. The default is to mask using *dust*.

--sizein Take into account the abundance annotations present in the input fasta file (search for the pattern "[>;]size=*integer*[;]" in sequence headers).

--sizeout Add abundance annotations to the output fasta files (add the pattern ";size=*integer*;" to sequence headers). If --sizein is specified, abundance annotations are reported to output files, and each cluster centroid receives a new abundance value corresponding to the total abundance of the amplicons included in the cluster (--centroids option). If --sizein is not specified, input abundances are set to 1 for amplicons, and to the number of amplicons per cluster for centroids.

--strand *plus|both*
When comparing sequences with the cluster seed, check the *plus* strand only (default) or check *both* strands.

--threads *positive integer*
Number of computation threads to use (1 to 256). The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available ressources and to launch one thread per logical core.

--uc *filename*
Output clustering results in *filename* using a uclust-like format. For a description of the format, see <http://www.drive5.com/usearch/manual/ucout.html>.

--usersort
When using --cluster_smallmem, allow any sequence input order, not just a decreasing length ordering.

Most searching options also apply to clustering:
--alnout, --blast6out, --fastapairs, --matched, --notmatched, --maxaccept, --maxreject, --samout, --userout, --userfields, score filtering, --gap penalties, masking. (see the Searching section).

Dereplication options:

--derep_fulllength *filename*
Merge strictly identical sequences contained in *filename*. Identical sequences are defined as having the same length and the same string of nucleotides (case insensitive, T and U are considered the same).

--maxuniquesize *positive integer*
Discard sequences with an abundance value greater than *integer*.

--minuniquesize *positive integer*
Discard sequences with an abundance value smaller than *integer*.

--output *filename*
Write the dereplicated sequences to *filename*, in fasta format and sorted by decreasing abundance. Identical sequences receive the header of the first sequence of their group. If --sizeout is used, the number of occurrences (i.e. abundance) of each sequence is indicated at the end of their fasta header using the pattern ";size=*integer*;".

--sizein Take into account the abundance annotations present in the input fasta file (search for the pattern "[>;]size=*integer*[;]" in sequence headers).

--sizeout Add abundance annotations to the output fasta file (add the pattern ";size=*integer*;" to sequence headers). If --sizein is specified, each unique sequence receives a new abundance value corresponding to its total abundance (sum of the abundances of its occurrences). If --sizein is not specified, input abundances are set to 1, and each unique sequence receives a new abundance value corresponding to its number of occurrences in the input file.

--strand *plus*|*both*
When searching for strictly identical sequences, check the *plus* strand only (default) or check *both* strands.

--topn *positive integer*
Output only the top *integer* sequences (i.e. the most abundant).

--uc *filename*
Output dereplication results in *filename* using a uclust-like format. For a description of the format, see <http://www.drive5.com/usearch/manual/ucout.html>. In the context of dereplication, the option --uc_allhits has no effect on the --uc output.

Masking options:

An input sequence can be composed of lower- or uppercase nucleotides. Lowercase nucleotides are silently set to uppercase before masking, unless the --qmask soft option is used. Here are the results of combined masking options --qmask (or --dbmask for database sequences) and --hardmask, assuming each input sequences contains both lower and uppercase nucleotides:

qmask	hardmask	action

none	off	no masking, all symbols uppercased
none	on	no masking, all symbols uppercased
dust	off	masked symbols lowercased, others uppercased
dust	on	masked symbols changed to Ns, others uppercased
soft	off	lowercase symbols masked, no case changes
soft	on	lowercase symbols masked and changed to Ns

--hardmask
Mask low-complexity regions by replacing them with Ns instead of setting them to lower case.

--maskfasta *filename*
Mask simple repeats and low-complexity regions in sequences contained in *filename*. The default is to mask using *dust* (use --qmask to modify that behavior).

--output *filename*
Write the masked sequences to *filename*, in fasta format.

--qmask *none*|*dust*|*soft*
Mask simple repeats and low-complexity regions in sequences using the *dust* or the *soft* algorithms, or do not mask (*none*). The default is to mask using *dust*.

--threads *positive integer*
Number of computation threads to use (1 to 256). The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available ressources and to launch one thread per logical core.

Pairwise alignment options:

The results of the n * (n - 1) / 2 pairwise alignments are written to the result files specified with --alnout, --blast6out, --fastapairs --matched, --notmatched, --samout, --uc or --userout (see Searching section below). Specify either the --acceptall option to output all pairwise alignments, or specify an identity level with --id to discard weak alignments. Most other accept/reject options (see Searching options below) may also be used. Sequences are aligned on their *plus* strand only.

--allpairs_global *filename*
Perform optimal global pairwise alignments of all vs. all fasta sequences contained in *filename*. This command is multi-threaded.

--acceptall
Write the results of all alignments to output files. This option overrides all other accept/reject options (including --id).

--id *real* Reject the sequence match if the pairwise identity is lower than *real* (value ranging from 0.0 to 1.0 included).

--threads *positive integer*
Number of computation threads to use (1 to 256). The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available ressources and to launch one thread per logical core.

Searching options:

--alnout *filename*
Write pairwise global alignments to *filename* using a human-readable format. Use --rowlen to modify alignment length. Output order may vary when using multiple threads.

--blast6out *filename*
Write search results to *filename* using a blast-like tab-separated format of twelve fields (listed below), with one line per query-target matching (or lack of matching if --output_no_hits is used). Output order may vary when using multiple threads. A similar output can be obtain with --userout *filename* and --userfields

query+target+id+alnlen+mism+opens+qlo+qhi+tlo+thi+evalue+bits. A complete list and description is available in the section "Userfields" of this manual.

1. *query*: query label.
2. *target*: target (database sequence) label. The field is set to "*" if there is no alignment.
3. *id*: percentage of identity (real value ranging from 0.0 to 100.0). The percentage identity is defined as 100 * (matching columns) / (alignment length - terminal gaps). See fields id0 to id4 for other definitions.
4. *alnlen*: length of the query-target alignment (number of columns). The field is set to 0 if there is no alignment.
5. *mism*: number of mismatches in the alignment (zero or positive integer value).
6. *opens*: number of columns containing a gap opening (zero or positive integer value).
7. *qlo*: first nucleotide of the query aligned with the target. Always equal to 1 if there is an alignment, 0 otherwise.
8. *qhi*: last nucleotide of the query aligned with the target. Always equal to the length of the pairwise alignment. The field is set to 0 if there is no alignment.
9. *tlo*: irst nucleotide of the target aligned with the query. Always equal to 1 if there is an alignment, 0 otherwise.
10. *thi*: last nucleotide of the target aligned with the query. Always equal to the length of the pairwise alignment. The field is set to 0 if there is no alignment.
11. *evalue*: expectancy-value (not computed for nucleotide alignments). Always set to -1.
12. *bits*: bit score (not computed for nucleotide alignments). Always set to 0.

--db *filename*

Compare query sequences (specified with --usearch_global) to the fasta-formatted target sequences contained in *filename*, using global pairwise alignment.

--dbmask *none*|*dust*|*soft*

Mask simple repeats and low-complexity regions in target database sequences using the *dust* or the *soft* algorithms, or do not mask (*none*). Warning, when using *soft* masking search commands become case sensitive. The default is to mask using *dust*.

--dbmatched *filename*

Write database target sequences matching at least one query sequence to *filename*, in fasta format. If the option --sizeout is used, the number of queries that matched each target sequence is indicated using the pattern ";size=*integer*;".

--dbnotmatched *filename*

Write database target sequences not matching query sequences to *filename*, in fasta format.

--fastapairs *filename*

Write pairwise alignments of query and target sequences to *filename*, in fasta format.

--fulldp Dummy option for compatibility with usearch. To maximize search sensitivity, **vsearch** uses a 8-way 16-bit SIMD vectorized full dynamic programming algorithm (Needleman-Wunsch), whether or not --fulldp is specified.

--gapext *string*

Set penalties for a gap extension. See --gapopen for a complete description of the penalty declaration system. The default is to initialize the six gap extending penalties using a penalty of 2 for extending internal gaps and a penalty of 1 for extending terminal gaps, in both query and target sequences (i.e. 2I/1E).

--gapopen *string*

Set penalties for a gap opening. A gap opening can occur in six different contexts: in the query (Q) or in the target (T) sequence, at the left (L) or right (R) extremity of the sequence, or inside the sequence (I). Sequence symbols (Q and T) can be combined with location symbols (L, I, and R), and numerical values to declare penalties for all possible contexts: aQL/bQI/cQR/dTL/eTI/fTR, where abcdef are zero or positive integers, and "/" is used as a separator.

To simplify declarations, the location symbols (L, I, and R) can be combined, the symbol (E) can be used to treat both extremities (L and R) equally, and the symbols Q and T can be omitted to treat query and target sequences equally. For instance, the default is to declare a penalty of 20 for opening internal gaps and a penalty of 2 for opening terminal gaps (left or right), in both query and target sequences (i.e. 20I/2E). If only a numerical value is given, without any sequence or location symbol, then the penalty applies to all gap openings. To forbid gap-opening, an infinite penalty value can be declared with the symbol "*". Tu use **vsearch** as a semi-global aligner, a null-penalty can be applied to the left (L) or right (R) gaps.

vsearch always initializes the six gap opening penalties using the default parameters (20I/2E). The user is then free to declare only the values he/she wants to modify. The *string* is scanned from left to right, accepted symbols are (0123456789/LIREQT*), and later values override previous values.

Please note that **vsearch**, in contrast to usearch, only allows integer gap penalties. Because the lowest gap penalties are 0.5 by default in usearch, all default scores and gap penalties in **vsearch** have been doubled to maintain equivalent penalties and to produce identical alignments.

--hardmask

Mask low-complexity regions by replacing them with Ns instead of setting them to lower case. For more information, please see the Masking section.

--id *real* Reject the sequence match if the pairwise identity is lower than *real* (value ranging from 0.0 to 1.0 included). The search process sorts target sequences by decreasing number of *k*-mers they have in common with the query sequence, using that information as a proxy for sequence similarity. That efficient pre-filtering will also prevent pairwise alignments with weakly matching targets, as there needs to be at least 6 shared *k*-mers to start the pairwise alignment, and at least one out of every 16 *k*-mers from the query needs to match the target. Consequently, using values lower than --id 0.5 is not likely to capture more weakly matching targets. The pairwise identity is by default defined as the number of (matching columns) / (alignment length - terminal gaps). That definition can be modified by --iddef.

--iddef *0|1|2|3|4*

Change the pairwise identity definition used in --id. Values accepted are:

0. CD-HIT definition: (matching columns) / (shortest sequence length).
1. edit distance: (matching columns) / (alignment length).
2. edit distance excluding terminal gaps (same as --id).
3. Marine Biological Lab definition counting each extended gap (internal or terminal) as a single difference: 1.0 - [(mismatches + gaps)/(longest sequence length)]

4. BLAST definition, equivalent to --iddef 2 in a context of global pairwise alignment.

The option --userfields accepts the fields id0 to id4, in addition to the field id, to report the pairwise identity values corresponding to the different definitions.

--idprefix *positive integer*
Reject the sequence match if the first *integer* nucleotides of the target do not match the query.

--idsuffix *positive integer*
Reject the sequence match if the last *integer* nucleotides of the target do not match the query.

--leftjust Reject the sequence match if the pairwise alignment begins with gaps.

--match *integer*
Score assigned to a match (i.e. identical nucleotides) in the pairwise alignment. The default value is 2.

--matched *filename*
Write query sequences matching database target sequences to *filename*, in fasta format.

--maxaccepts *positive integer*
Maximum number of hits to accept before stopping the search. The default value is 1. This option works in pair with --maxrejects. The search process sorts target sequences by decreasing number of *k*-mers they have in common with the query sequence, using that information as a proxy for sequence similarity. After pairwise alignments, if the first target sequence passes the acceptation criteria, it is accepted as best hit and the search process stops for that query. If --maxaccepts is set to a higher value, more hits are accepted. If --maxaccepts and --maxrejects are both set to 0, the complete database is searched.

--maxdiffs *positive integer*
Reject the sequence match if the alignment contains at least *integer* substitutions, insertions or deletions.

--maxgaps *positive integer*
Reject the sequence match if the alignment contains at least *integer* insertions or deletions.

--maxhits *positive integer*
Maximum number of hits to show once the search is terminated (hits are sorted by decreasing identity). Unlimited by default. That option applies to --alnout, --blast6out, --fastapairs, --samout, --uc, or --userout output files.

--maxid *real*
Reject the sequence match if the percentage of identity between the two sequences is greater than *real*.

--maxqsize *positive integer*
Reject query sequences with an abundance greater than *integer*.

--maxqt *real*
Reject if the query/target sequence length ratio is greater than *real*.

--maxrejects *positive integer*
Maximum number of non-matching target sequences to consider before stopping the search. The default value is 32. This option works in pair with --maxaccepts. The search process sorts target sequences by decreasing number of *k*-mers they have in common with the query sequence, using that information as a proxy for sequence similarity. After pairwise alignments, if none of the first 32 examined target sequences pass the acceptation criteria, the search process stops for that query (no hit). If --maxrejects

is set to a higher value, more target sequences are considered. If --maxaccepts and --maxrejects are both set to 0, the complete database is searched.

--maxsizeratio *real*
Reject if the query/target abundance ratio is greater than *real*.

--maxsl *real*
Reject if the shorter/longer sequence length ratio is greater than *real*.

--maxsubs *positive integer*
Reject the sequence match if the pairwise alignment contains more than *integer* substitutions.

--mid *real*
Reject the sequence match if the percentage of identity is lower than *real* (ignoring all gaps, internal and terminal).

--mincols *positive integer*
Reject the sequence match if the alignment length is shorter than *integer*.

--minqt *real*
Reject if the query/target sequence length ratio is lower than *real*.

--minsizeratio *real*
Reject if the query/target abundance ratio is lower than *real*.

--minsl *real*
Reject if the shorter/longer sequence length ratio is lower than *real*.

--mintsize *positive integer*
Reject target sequences with an abundance lower than *integer*.

--mismatch *integer*
Score assigned to a mismatch (i.e. different nucleotides) in the pairwise alignment. The default value is -4.

--notmatched *filename*
Write query sequences not matching database target sequences to *filename*, in fasta format.

--output_no_hits
Write both matching and non-matching queries to --alnout, --blast6out, --samout or --userout output files (--uc and --uc_allhits output files always feature non-matching queries). Non-matching queries are labelled "No hits" in --alnout files.

--qmask *none*|*dust*|*soft*
Mask simple repeats and low-complexity regions in query sequences using the *dust* or the *soft* algorithms, or do not mask (*none*). Warning, when using *soft* masking search commands become case sensitive. The default is to mask using *dust*.

--query_cov *real*
Reject if the fraction of the query aligned to the target sequence is lower than *real*. The query coverage is computed as (matches + mismatches) / query sequence length. Internal or terminal gaps are not taken into account.

--rightjust
Reject the sequence match if the pairwise alignment ends with gaps.

--rowlen *positive integer*
Width of alignment lines in --alnout output. The default value is 64. Set to 0 to eliminate wrapping.

--samout *filename*
Write alignment results to *filename* in the SAM format. For a description of the format, see <https://github.com/samtools/hts-specs>. Output order may vary when using

multiple threads.

--self Reject the sequence match if the query and target labels are identical.

--selfid Reject the sequence match if the query and target sequences are strictly identical.

--sizeout Add abundance annotations to the output of the option --dbmatched (using the pattern ";size=*integer*;"), to report the number of queries that matched each target.

--strand *plus*|*both*
When searching for similar sequences, check the *plus* strand only (default) or check *both* strands.

--target_cov *real*
Reject the sequence match if the fraction of the target sequence aligned to the query sequence is lower than *real*. The target coverage is computed as (matches + mismatches) / target sequence length. Internal or terminal gaps are not taken into account.

--threads *positive integer*
Number of computation threads to use (1 to 256). The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available ressources and to launch one thread per logical core.

--top_hits_only
Output only the hits with the highest percentage of identity with the query.

--uc *filename*
Output searching results in *filename* using a uclust-like format. For a description of the format, see <http://www.drive5.com/usearch/manual/ucout.html>. Output order may vary when using multiple threads.

--uc_allhits
When using the --uc option, show all hits, not just the top hit for each query.

--usearch_global *filename*
Compare target sequences (--db) to the fasta-formatted query sequences contained in *filename*, using global pairwise alignment.

--userfields *string*
When using --userout, select and order the fields written to the output file. Fields are separated by "+" (e.g. query+target+id). See the "Userfields" section for a complete list of fields.

--userout *filename*
Write user-defined tab-separated output to *filename*. Select the fields with the option --userfields. Output order may vary when using multiple threads. If --userfields is empty or not present, *filename* is empty.

--weak_id *real*
Show hits with percentage of identity of at least *real*, without terminating the search. A normal search stops as soon as enough hits are found (as defined by --maxaccepts, --maxrejects, and --id). As --weak_id reports weak hits that are not deduced from --maxaccepts, high --id values can be used, hence preserving both speed and sensitivity. Logically, *real* must be smaller than the value indicated by --id.

--wordlength *positive integer*
Length of words (i.e. *k*-mers) for database indexing. The range of possible values goes from 3 to 15, but values near 8 are generally recommended. Longer words may reduce the sensitivity for weak similarities, but can increase accuracy. On the other hand, shorter words may increase sensitivity, but can reduce accuracy. Computation time will generally increase with shorter words and decrease with longer words. Memory requirements for a part of the index increase with a factor of 4 each time word length increases by one nucleotide, and this generally becomes significant for long words (12

or more). The default value is 8.

Shuffling options:

--output *filename*
Write the shuffled sequences to *filename*, in fasta format.

--seed *positive integer*
When shuffling sequence order, use *integer* as seed. A given seed will always produce the same output order (useful for replicability). Set to 0 to use a pseudo-random seed (default behavior).

--shuffle *filename*
Pseudo-randomly shuffle the order of sequences contained in *filename*.

--topn *positive integer*
Output only the top *integer* sequences.

Sorting options:

Fasta entries are sorted by decreasing abundance (--sortbysize) or sequence length (--sortbylength). To obtain a stable sorting order, ties are sorted by decreasing abundance and label increasing alpha-numerical order (--sortbylength), or just by label increasing alpha-numerical order (--sortbysize). Label sorting assumes that all sequences have unique labels. The same applies to the automatic sorting performed during chimera checking (--uchime_denovo), dereplication (--derep_fulllength), and clustering (--cluster_fast and --cluster_size).

--maxsize *positive integer*
When using --sortbysize, discard sequences with an abundance value greater than *integer*.

--minsize *positive integer*
When using --sortbysize, discard sequences with an abundance value smaller than *integer*.

--output *filename*
Write the sorted sequences to *filename*, in fasta format.

--relabel *string*
Relabel sequence using the prefix *string* and a ticker (1, 2, 3, etc.) to construct the new headers. Use --sizeout to conserve the abundance annotations.

--sizeout When using --relabel, report abundance annotations to the output fasta file (using the pattern ";size=*integer*;").

--sortbylength *filename*
Sort by decreasing length the sequences contained in *filename*. See the general options --minseqlength and --maxseqlength to eliminate short and long sequences.

--sortbysize *filename*
Sort by decreasing abundance the sequences contained in *filename* (the pattern "[>;]size=*integer*[;]" has to be present). See the options --minsize and --maxsize to eliminate rare and dominant sequences.

--topn *positive integer*
Output only the top *integer* sequences (i.e. the longest or the most abundant).

Userfields (fields accepted by the --userfields option):

aln Print a string of M (match), D (delete, i.e. a gap in the query) and I (insert, i.e. a gap in the target) representing the pairwise alignment. Empty field if there is no alignment.

alnlen Print the length of the query-target alignment (number of columns). The field is set to 0 if there is no alignment.

bits Bit score (not computed for nucleotide alignments). Always set to 0.

caln Compact representation of the pairwise alignment using the CIGAR format (Compact Idiosyncratic Gapped Alignment Report): M (match), D (deletion) and I (insertion). Empty field if there is no alignment.

evalue E-value (not computed for nucleotide alignments). Always set to -1.

exts Number of columns containing a gap extension (zero or positive integer value).

gaps Number of columns containing a gap (zero or positive integer value).

id Percentage of identity (real value ranging from 0.0 to 100.0). The percentage identity is defined as 100 * (matching columns) / (alignment length - terminal gaps).

id0 CD-HIT definition of the percentage of identity (real value ranging from 0.0 to 100.0) using the length of the shortest sequence in the pairwise alignment as denominator: 100 * (matching columns) / (shortest sequence length).

id1 The percentage of identity (real value ranging from 0.0 to 100.0) is defined as the edit distance: 100 * (matching columns) / (alignment length).

id2 The percentage of identity (real value ranging from 0.0 to 100.0) is defined as the edit distance, excluding terminal gaps. The field id2 is an alias for the field id.

id3 Marine Biological Lab definition of the percentage of identity (real value ranging from 0.0 to 100.0), counting each extended gap (internal or terminal) as a single difference and using the length of the longest sequence in the pairwise alignment as denominator: 100 * (1.0 - [(mismatches + gaps) / (longest sequence length)]).

id4 BLAST definition of the percentage of identity (real value ranging from 0.0 to 100.0), equivalent to --iddef 2 in a context of global pairwise alignment.

ids Number of matches in the alignment (zero or positive integer value).

mism Number of mismatches in the alignment (zero or positive integer value).

opens Number of columns containing a gap opening (zero or positive integer value).

pairs Number of columns containing only nucleotides. That value corresponds to the length of the alignment minus the gap-containing columns (zero or positive integer value).

pctgaps Number of columns containing gaps expressed as a percentage of the alignment length (real value ranging from 0.0 to 100.0).

pctpv Percentage of positive columns. When working with nucleotide sequences, this is equivalent to the percentage of matches (real value ranging from 0.0 to 100.0).

pv Number of positive columns. When working with nucleotide sequences, this is equivalent to the number of matches (zero or positive integer value).

qcov Fraction of the query sequence that is aligned with the target sequence (real value ranging from 0.0 to 100.0). The query coverage is computed as 100.0 * (matches + mismatches) / query sequence length. Internal or terminal gaps are not taken into account. The field is set to 0.0 if there is no alignment.

qframe Query frame (-3 to +3). That field only concerns coding sequences and is not computed by **vsearch**. Always set to +0.

qhi Last nucleotide of the query aligned with the target. Always equal to the length of the pairwise alignment. The field is set to 0 if there is no alignment.

qihi Last nucleotide of the query aligned with the target (ignoring terminal gaps). Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

qilo First nucleotide of the query aligned with the target (ignoring initial gaps). Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

ql Query sequence length (positive integer value). The field is set to 0 if there is no alignment.

qlo First nucleotide of the query aligned with the target. Always equal to 1 if there is an alignment, 0 otherwise.

qrow Print the sequence of the query segment as seen in the pairwise alignment (i.e. with gap insertions if need be). Empty field if there is no alignment.

qs Query segment length. Always equal to query sequence length.

qstrand Query strand orientation (+ or - for nucleotide sequences). Empty field if there is no alignment.

query Query label.

raw Raw alignment score (negative, null or positive integer value). The score is the sum of match rewards minus mismatch penalties, gap openings and gap extensions. The field is set to 0 if there is no alignment.

target Target label. The field is set to "*" if there is no alignment.

tcov Fraction of the target sequence that is aligned with the query sequence (real value ranging from 0.0 to 100.0). The target coverage is computed as 100.0 * (matches + mismatches) / target sequence length. Internal or terminal gaps are not taken into account. The field is set to 0.0 if there is no alignment.

tframe Target frame (-3 to +3). That field only concerns coding sequences and is not computed by **vsearch**. Always set to +0.

thi Last nucleotide of the target aligned with the query. Always equal to the length of the pairwise alignment. The field is set to 0 if there is no alignment.

tihi Last nucleotide of the target aligned with the query (ignoring terminal gaps). Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

tilo First nucleotide of the target aligned with the query (ignoring initial gaps). Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

tl Target sequence length (positive integer value). The field is set to 0 if there is no alignment.

tlo First nucleotide of the target aligned with the query. Always equal to 1 if there is an alignment, 0 otherwise.

trow Print the sequence of the target segment as seen in the pairwise alignment (i.e. with gap insertions if need be). Empty field if there is no alignment.

ts Target segment length. Always equal to target sequence length. The field is set to 0 if there is no alignment.

tstrand Target strand orientation (+ or - for nucleotide sequences). Always set to "+", so reverse strand matches have tstrand "+" and qstrand "-". Empty field if there is no alignment.

DELIBERATE CHANGES

If you are a usearch user, our objective is to make you feel at home. That's why **vsearch** was designed to behave like usearch, to some extent. Like any complex software, usearch is not free from quirks and inconsistencies. We decided not to reproduce some of them, and for complete transparency, to document here the deliberate changes we made.

During a search with usearch, when using the options --blast6out and --output_no_hits, for queries with no match the number of fields reported is 13, where it should be 12. This is corrected in **vsearch**.

The field raw of the --userfields option is not informative in usearch. This is corrected in **vsearch**.

The fields qlo, qhi, tlo, thi now have counterparts (qilo, qihi, tilo, tihi) reporting alignment coordinates

ignoring terminal gaps.

In usearch, when using the option --output_no_hits, queries that receive no match are reported in blast6out file, but not in the alignment output file. This is corrected in **vsearch**.

vsearch introduces a new --cluster_size command that sorts sequences by decreasing abundance before clustering.

vsearch reintroduces --iddef alternative pairwise identity definitions that were removed from usearch.

vsearch extends the --topn option to sorting commands.

vsearch extends the --sizein option to dereplication (--derep_fulllength) and clustering (--cluster_fast).

vsearch treats T and U as identical nucleotides during dereplication.

vsearch sorting is stabilized by using sequence abundances or sequences labels as secondary or tertiary keys.

NOVELTIES

vsearch introduces new options not present in usearch 7. They are described in the "Options" section of this manual. Here is a short list:

- alignwidth (chimera checking)
- cluster_size (clustering)
- fasta_width (general option)
- iddef (clustering, pairwise alignment, searching)
- maxuniquesize (dereplication)
- shuffle (shuffling)

EXAMPLES

Align all sequences in a database with each other and output all pairwise alignments:

vsearch --allpairs_global *database.fas* --alnout *results.aln* --acceptall

Check for the presence of chimeras (*de novo*); parents should be at least 1.5 times more abundant than chimeras. Output non-chimeric sequences in fasta format (no wrapping):

vsearch --uchime_denovo *queries.fas* --nonchimeras *results.fas* --fasta_width 0 --abskew 1.5

Cluster with a 97% similarity threshold, collect cluster centroids, and write cluster descriptions using a uclust-like format:

vsearch --cluster_fast *queries.fas* --id 0.97 --centroids *centroids.fas* --uc *clusters.uc*

Dereplicate the sequences contained in queries.fas, take into account the abundance information already present, write unwrapped sequences to output with the new abundance information, discard all sequences with an abundance of 1:

vsearch --derep_fulllength *queries.fas* --output *queries_masked.fas* --sizein --sizeout --fasta_width 0 --minuniquesize 2

Mask simple repeats and low complexity regions in the input fasta file (masked regions are lowercased), and write the results to the output file:

vsearch --maskfasta *queries.fas* --output *queries_masked.fas* --qmask dust

Search queries in a reference database, with a 80%-similarity threshold, take terminal gaps into account when calculating pairwise similarities:

vsearch --usearch_global *queries.fas* --db *references.fas* --alnout *results.aln* --id 0.8 --iddef 1

Search a sequence dataset against itself (ignore self hits), get all matches with at least 60% identity, and collect results in a blast-like tab-separated format:

vsearch --usearch_global *queries.fas* --db *queries.fas* --id 0.6 --self --blast6out *results.blast6*

--maxaccepts 0 --maxrejects 0

Shuffle the input fasta file (change the order of sequences) in a repeatable fashion (fixed seed), and write unwrapped fasta sequences to the output file:

vsearch --shuffle *queries.fas* --output *queries_shuffled.fas* --seed 13 --fasta_width 0

Sort by decreasing abundance the sequences contained in queries.fas (using the "size=*integer*" information), relabel the sequences while preserving the abundance information (with --sizeout), keep only sequences with an abundance equal to or greater than 2:

vsearch --sortbysize *queries.fas* --output *queries_sorted.fas* --relabel sampleA_ --sizeout --minsize 2

AUTHORS

Implementation by Torbjørn Rognes and Tomas Flouri, documentation by Frédéric Mahé.

REPORTING BUGS

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

AVAILABILITY

Source code and binaries are available at <https://github.com/torognes/vsearch>.

COPYRIGHT

SEE ALSO

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

swarm, a fast and accurate amplicon clustering method by Frédéric Mahé and Torbjørn Rognes, available at <https://github.com/torognes/swarm>.

VERSION HISTORY

New features and important modifications of **vsearch** (short lived or minor bug releases may not be mentioned):

v1.0.0 released November 28th, 2014
: First public release.

v1.0.1 released December 1st, 2014
: Bug fixes (sortbysize, semicolon after size annotation in headers) and minor changes (labels as secondary sort key for most sorts, treat T and U as identical for dereplication, only output size in dbmatched file if sizeout specified).

v1.0.2 released December 6th, 2014
Bug fixes (ssse3/sse4.1 requirement, memory leak).

v1.0.3 released December 6th, 2014
Bug fix (now writes help to stdout instead of stderr).

v1.0.4 released December 8th, 2014
Added --allpairs_global option. Reduced memory requirements slightly. Removed memory leaks.

v1.0.5 released December 9th, 2014
Fixes a minor bug with --allpairs_global and --acceptall options.

v1.0.6 released December 14th, 2014
Fixes a memory allocation bug in chimera detection (--uchime_ref option).

v1.0.7 released December 19th, 2014
Fixes a bug in the output from chimera detection with the --uchimeout option.

v1.0.8 released January 22nd, 2015
Introduces several changes and bug fixes:

- a new linear memory aligner for alignment of sequences longer than 5,000 nucleotides,
- a new --cluster_size command that sorts sequences by decreasing abundance before clustering,
- meaning of userfields qlo, qhi, tlo, thi changed for compatibility with usearch,
- new userfields qilo, qihi, tilo, tihi gives alignment coordinates ignoring terminal gaps,
- in --uc output files, a perfect alignment is indicated with a "=" sign,
- the option --cluster_fast will now sort sequences by decreasing length, then by decreasing abundance and finally by sequence identifier,
- default --maxseqlength value set to 50,000 nucleotides,
- fix for bug in alignment in rare cases,
- fix for lack of detection of under- or overflow in SIMD aligner.

v1.0.9 released January 22nd, 2015
Fixes a bug in the function sorting sequences by decreasing abundance (--sortbysize).

v1.0.10 released January 23rd, 2015
Fixes a bug where the sizein option was ignored and always treated as on, affecting clustering and dereplication commands.

v1.0.11 released February 5th, 2015
Introduces the possibility to output results in SAM format (for clustering, pairwise alignment and searching).

v1.0.12 released February 6th, 2015
Temporarily fixes a problem with long headers in FASTA files.

v1.0.13 released February 17th, 2015
Fix a memory allocation problem when computing multiple sequence alignments with the --msaout and --consout options, as well as a memory leak. Also increased line buffer for reading FASTA files to 4MB.

v1.0.14 released February 17th, 2015
Fix a bug where the multiple alignment and consensus sequence computed after clustering ignored the strand of the sequences. Also decreased size of line buffer for reading FASTA files to 1MB again due to excessive stack memory usage.