## NAME

vsearch — dereplicate, filter, sort, search, compare and clusterize amplicons from metagenomic projects

## SYNOPSIS

**vsearch** [ *options* ] *filename*

## DESCRIPTION

Environmental or clinical molecular studies generate large volumes of amplicons (e.g. SSU-rRNA sequences) that need to be filtered, dereplicated, searched, clustered or compared to sequences from other studies. 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 to guarantee a fast and accurate data processing.

Nucleotidic sequence comparisons is at the core of **vsearch**. To speed up comparisons, **vsearch** implements an efficient *k*-mer filtering, and 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, **vsearch** exits with an error message.

### Input

**vsearch** input is a fasta file containing one or several nucleotidic 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 ends with the pattern ";size=*integer*;", **vsearch** will interpret *integer* as the abundance of the sequence (in a dereplicated fasta file for instance). The nucleotidic 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; **vsearch** exits with an error message if any other symbol is present in the sequence. Optionally, **vsearch** can be compiled to accepted compressed fasta files as input (gz and bzip2 formats).

### Options

**vsearch** recognizes a large number of command-line options. For an easier navigation, options are grouped by theme (dereplication, filtering, sorting, searching, comparison, clustering). 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 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 dereplication or searching).

**--notrunclabels**
do not truncate sequence labels at first space, use the full header.

**--strand** *plus|both*
when searching or dereplicating, check the *plus* strand only (default) or check *both* strands.

**--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. The default is to launch one thread per available CPU core.

**--uc** *filename*
when searching, clustering or dereplicating, output results in *filename* using a uclust-like format.

**--uc_allhits**
when searching, clustering or dereplicating, and when using the --uc option, show all hits, not just top hit.

Clustering options:

**--centroids** *filename*
output cluster centroid sequences to *filename* file.

**--cluster_fast** *filename*
use the fast clustering algorithm and write the results to *filename*.

**--cluster_smallmem** *filename*
use a slower clustering algorithm (consumes less memory) and write the results to *filename*.

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

**--usersort**
when using --cluster_smallmem, conserve the initial input order of sequences, do not sort sequences by decreasing length before clustering.

Dereplication, masking, shuffling and sorting options:

**--derep_fulllength** *filename*
merge strictly identical sequences contained in *filename*. Redundant sequences receive the header of the sequence of their group, and the number of occurrences (abundance) is indicated at the end of the fasta header using the pattern ";size=X;".

**--maskfasta** *filename*
mask sequences contained in *filename*.

**--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*.

**--minuniquesize** *positive integer*
when dereplicating, discard sequences with an abundance value smaller than *integer*.

**--output** *filename*
when dereplicating, sorting or shuffling, write the results to *filename*.

**--relabel** *string*
when sorting, relabel sequence headers using *string* as suffix.

**--seed** *positive integer*
when shuffling, use *integer* as seed. Set to 0 to use a pseudo-random seed.

**--sizein** read abundance annotation from input

**--sizeout** add abundance annotation to output

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

**--sortbylength** *filename*
sort by decreasing length the sequences contained in *filename*.

**--sortbysize** *filename*
sort by decreasing abundance the sequences contained in *filename*.

**--topn positive integer**
when dereplicating, sorting or shuffling, output just the top *integer* sequences.

Searching options:

**--alnout** *filename*
write pairwise global alignments to *filename* using a human-readable format.

**--blast6out** *filename*
write search results to *filename* using a blast-like tab-separated format.

**--db** *filename*
compare query sequences to the fasta-formatted subject sequences contained in *filename*, using global pairwise alignment.

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

**--dbmatched** *filename*
write database subject sequences matching at least one query sequence to *filename*, in fasta format.

**--dbnotmatched** *filename*
write database subject sequences not matching query sequences to *filename*, in fasta format.

**--fastapairs** *filename*
write pairwise alignments of query and subject sequences to *filename*, in fasta format.

**--fulldp** dummy option. To maximize search sensitivity, vsearch uses a 8-way SIMD vectorized full dynamic programming algorithm (Needleman-Wunsch), whether or not --fulldp is specified.

**--gapext** *string*
penalties for gap extension (2I/1E)

**--gapopen** *string*
penalties for gap opening (20I/2E)

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

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

**--idprefix** *positive integer*
reject the subject sequence if the first *integer* nucleotides do not match the query sequence.

**--idsuffix** *positive integer*
reject the subject sequence if the last *integer* nucleotides do not match the query sequence.

**--leftjust** reject the subject sequence if the alignment begins with gaps.

**--leftjust** reject the subject sequence if the 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 subject sequences to *filename*, in fasta format.

**--maxaccepts** *positive integer*
maximum number of hits to accept before stopping the search. The default value is 1. That option works in pair with maxrejects. The search process sorts subject sequences by decreasing number of kmers they have in common with the query sequence, using that information as a proxy for sequence similarity. If the first subject 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 subject sequence if the alignment contains at least *integer* substitutions, insertions or deletions.

**--maxgaps** *positive integer*
reject the subject sequence 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). The default value is 1. Set to 0 to ignore the option.

**--maxid** *real*
reject the subject sequence if its percentage of identity with the query is equal or greater than *real*.

**--maxqsize** *positive integer*
reject query sequences with an abundance equal or greater than *integer*.

**--maxqt** *real*
reject if the query/subject length ratio is equal or greater than *real*.

**--maxrejects** *positive integer*
maximum number of non-matching subject sequences to consider before stopping the search. The default value is 32. That option works in pair with maxaccepts. The search process sorts subject sequences by decreasing number of kmers they have in common with the query sequence, using that information as a proxy for sequence similarity. If none of the first 32 subject sequences pass the acceptation criteria, the search process stops for that query (no hit). If maxrejects is set to a higher value, more subject sequences are considered. If maxaccepts and maxrejects are both set to 0, the complete database is searched.

**--maxsizeratio** *real*
reject if the query/subject abundance ratio is equal or greater than *real*.

**--maxsl** *real*
reject if the shorter/longer length ratio is equal or greater than *real*.

**--maxsubs** *positive integer*
reject the subject sequence if the alignment contains at least *integer* substitutions.

**--mid** *real*
reject the subject sequence if its percentage of identity with the query is lower than *real* (ignoring gaps).

**--mincols** *positive integer*
reject the subject sequence if the alignment length is shorter than *integer*.

**--minqt** *real*
reject if the query/subject length ratio is lower than *real*.

**--minsizeratio** *real*
reject if the query/subject abundance ratio is lower than *real*.

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

**--mintsize** *positive integer*
reject subject 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 subject sequences to *filename*, in fasta format.

**--output_no_hits** *filename*
write both matching and non-matching queries to output files. Non-matching queries are labelled "no hit" **(to be verified)**.

**--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*). The default is to mask using *dust*.

**--query_cov** *real*
reject if the fraction of the query aligned to the subject sequence is lower than *real*.

**--rightjust**
reject the subject sequence if the alignment ends with gaps.

**--rowlen** *positive integer*
width of alignment lines in alnout output. The default value is 64.

**--self** reject the alignment if the query and subject labels are identical.

**--selfid** reject the alignment if the query and subject sequences are identical.

**--target_cov** *real*
reject if the fraction of the subject sequence aligned to the query sequence is lower than *real*.

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

**--userfields** *string*
when using --userout, select and order the fields written to the output file. See the next section for a complete list of fields.

**--userout** *filename*
write user-defined tab-separated output to *filename*. See "userfields".

**--vsearch_global** *filename*
*filename* of queries for global alignment search.

**--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 index. The default value is 8.

Fields: (evalue and bits are not calculated for nucleotide sequences, as does usearch. Value is set to 0)

**--blabla** vavavtat a a a a a a.

## EXAMPLES

(in progress)

Search queries in a reference database, with a 80%-similarity threshold:

**vsearch** --vsearch_global *queries.fas* --db *references.fas* --alnout *results.aln* --id 0.8

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** --vsearch_global *queries.fas* --db *queries.fas* --id 0.6 --alnout *results.aln* --self --blast6out *results.blast6* --maxaccepts 0 --maxrejects 0

clusterize 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*

## LIMITATIONS

**vsearch** does not yet perform chimera detection.

## 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

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

## COPYRIGHT

**vsearch** binaries may include code from the bzip2 library copyright Julian R. Seward.

## SEE ALSO

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

## VERSION HISTORY

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

**v1.0** released November 1st, 2014
First public release