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 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, exits with an error message if ascii characters 0 to 8, 14 to 31, "." and "-" are present. All other characters are stripped and complained about in a warning message (non-blocking).

vsearch operations are case insensitive, except when soft masking is activated. For --vsearch_global (searching), --cluster_fast and --cluster_smallmem (clustering), and --maskfasta (masking) 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 aligning sequences during searching and clustering, T and U are considered identical, regardless of their case. If two symbols are non-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) is considered a match, and given a positive match score (+2).

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, clustering or searching).

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

--strand *plus|both*
when searching, clustering or dereplicating, 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 launch one thread per available logical 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.

--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*.

--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 of twelve fields (listed below), with one line per query-target matching (or lack of matching if --output_no_hits is used). 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 "Fields" of this manual.

1. Query label.
2. Target (database sequence or cluster centroid) label. The field is set to "*" if there is no alignment.
3. 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).
4. Length of the query-target alignment (number of columns). The field is set to 0 if there is no alignment.
5. Number of mismatches in the alignment (zero or positive integer value).

6. Number of columns containing a gap opening (zero or positive integer value).
7. First nucleotide of the query aligned with the target. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.
8. Last nucleotide of the query aligned with the target. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.
9. First nucleotide of the target aligned with the query. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.
10. Last nucleotide of the target aligned with the query. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.
11. E-value (not computed for nucleotidic alignments). Always set to 0.
12. Bit score (not computed for nucleotidic alignments). Always set to 0.

--db *filename*
compare query sequences 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.

--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. 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). The pairwise identity is defined as the number of (matching columns) / (alignment length - terminal gaps).

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

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

--leftjust reject the target 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 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. That 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. 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 target sequence if the alignment contains at least *integer* substitutions, insertions or deletions.

--maxgaps *positive integer*
reject the target 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 target 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/target length ratio is equal or greater than *real*.

--maxrejects *positive integer*
maximum number of non-matching target sequences to consider before stopping the search. The default value is 32. That 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. If none of the first 32 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 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 target sequence if the alignment contains at least *integer* substitutions.

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

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

--minqt *real*
reject if the query/target 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 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 output files (--alnout, --blast6out, and --userout. Output files --uc and --uc_allhits always feature non-matching queries). Non-matching queries are labelled "no hit" in --alnout files **(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*). 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*.

--rightjust
reject the target sequence if the alignment ends with gaps.

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

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

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

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

--top_hits_only
output only the 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 indexing. The default value is 8.

Fields:

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 nucleotidic 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).

evalue E-value (not computed for nucleotidic 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).

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 nucleotidic 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 nucleotidic 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). **(what is the formula?)**.

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. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

ql Query sequence length.

qlo First nucleotide of the query aligned with the target. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

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 nucleotidic sequences).

query Query label.

raw Raw alignment score (not computed for nucleotidic alignments). Always set to 0.

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). **(what is the formula?)**. The field is set to 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. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

tl Target sequence length.

tlo First nucleotide of the target aligned with the query. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.

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 nucleotidic sequences). Always set to "+", so reverse strand matches have tstrand "+" and qstrand "-".

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 extend. 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. **vsearch** outputs 12 fields.

NOVELTIES

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

- shuffle
- fasta_width

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

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