NAME

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

SYNOPSIS

Clustering:

vsearch [*options*] (--cluster_fast|--cluster_smallmem) *fastafile* --uc *outputfile* --id *real*

Dereplication:

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

Masking:

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

Searching:

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

Shuffling:

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

Sorting:

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

DESCRIPTION

Environmental or clinical molecular studies generate large volumes of amplicons (e.g. SSU-rRNA sequences) that need to be dereplicated, filtered, sorted, searched, clusterized 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 guaranteeing 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 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.

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 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 dereplication, sorting and searching.

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, and 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 non-blocking warning message.

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) also receives a score of zero.

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 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.

--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.

Clustering options:

--centroids *filename*
output cluster centroid sequences to *filename* file **in fasta format?**.

--cluster_fast *filename*
clusterize the fasta sequences in *filename*, perform a sorting by decreasing sequence length first.

--cluster_smallmem *filename*
clusterize the fasta sequences in *filename* without modifying their order first. Sequence must 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 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. Use --construncate to take terminal gaps into account.

--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 using shortest sequence as numerator.
1. edit distance.
2. edit distance excluding terminal gaps (default value).
3. Marine Biological Lab definition counting each extended gap as a single difference.
4. BLAST, same as 2 for global pairwise alignments.

--msaout *filename*
output multiple sequence alignments of each cluster to *filename*.

--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*.

--strand *plus*|*both*
when comparing sequences to the cluster seeds, check the *plus* strand only (default) or check *both* strands.

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

--uc_allhits
when using the --uc option, show all hits, not just the top hit for each seed. **TO BE TESTED**.

--usersort
when using --cluster_smallmem, allows to use any sequence input order, not only a decreasing length sorting.

Many searching options also apply to clustering:
--alnout, --blast6out, --userout, --userfields, fastapairs, --matched, --notmatched, --max-accept, --maxreject, score filtering, gap --penalties, masking, etc. (documentation in progress).

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 as different). As **vsearch** needs to read *filename* twice, *filename* must be a real file, not a stream.

--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) is indicated at the end of the 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*[;]").

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

--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.

--uc *filename*
output dereplication results in *filename* using a uclust-like format. See <http://www.drive5.com/usearch/manual/ucout.html> for a description of the format.

--uc_allhits
when using the --uc option, show all hits, not just the top hit for each query. In the context of dereplication, that option has no effect.

Masking options:

An input sequence can be composed of lower- or uppercase nucleotides. Lowercase nucleotides are silently converted 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, lowercase converted to uppercase
none	on	idem
dust	off	masked symbols converted to lowercase, the rest converted to uppercase
dust	on	masked symbols replaced by Ns, the rest converted to upper case
soft	off	lowercase symbols considered masked, no case changes
soft	on	lowercase symbols considered masked and converted 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* (see --qmask to modify that behavior).

--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*.

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

Shuffling options:

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

--seed *positive integer*
when shuffling sequence order, use *integer* as seed. 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:

--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.

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 "Fields" 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. Nucleotide numbering starts from 1. The field is set to 0 if there is no alignment.
8. *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.

9. *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.
10. *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.
11. *evalue*: expectancy-value (not computed for nucleotidic alignments). Always set to -1.
12. *bits*: bit score (not computed for nucleotidic alignments). Always set to 0.

--db *filename*
compare query sequences (--vsearch_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. 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 penaly 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 null or positive integers, and "/" is use as 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 ommited to treat both 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 declare an infinite penalty value, the symbol "*" can be used to indicate that gap openings are forbidden in that context.
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 wants to modify. The *string* is scanned from left to right, accepted symbols are (0123456789/LIREQT*), and later values override previous values.

--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 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 using shortest sequence as numerator.
1. edit distance.
2. edit distance excluding terminal gaps (default value of --id).
3. Marine Biological Lab definition counting each extended gap as a single difference.
4. BLAST, same as 2 for global pairwise alignments.

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 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). Unlimited by default value. **It applies to alnout, blast6out, uc, userout, fastapairs**.

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

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

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

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

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

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

--mincols *positive integer*
reject the target sequence 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, and --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 such: 100.0 * (matches + mismatches) / query sequence length. Internal or terminal gaps are not taken into account.

--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 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 strictly identical.

--sizeout add abundance annotations to the output of the option --dbmatched (using the pattern ";size=*integer*").

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

--target_cov *real*
reject if the fraction of the target sequence aligned to the query sequence is lower than *real*. The target coverage is computed as such: 100.0 * (matches + mismatches) / target sequence length. Internal or terminal gaps are not taken into account.

--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. See <http://www.drive5.com/usearch/manual/ucout.html> for a description of the format. 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. **TO BE TESTED**.

--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 next 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.

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

--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.

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 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). Empty field if there is no alignment.

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).

id0 CD-HIT definition of the percentage of identity, using the shortest sequence in the pairwise alignment as numerator (real value ranging from 0.0 to 100.0).

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 as a single difference.

id4 BLAST definition of the percentage of identity (real value ranging from 0.0 to 100.0), same as the field id2 for global pairwise alignments.

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). The query coverage is computed as such: 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. 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. 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). 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 such: 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. 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. 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 "-". 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 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. This is corrected in **vsearch**.

The fields qlo, qhi, tlo, thi and raw of the --userfields option are not informative in usearch. This is corrected in **vsearch**.

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

vsearch extends the --topn option to sorting commands.

NOVELTIES

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

- shuffle
- fasta_width
- iddef
- maxuniquesize

EXAMPLES

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*

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 soft

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

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

> **vsearch** --vsearch_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** --vsearch_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

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

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

COPYRIGHT

SEE ALSO

swipe, an extremely fast Smith-Waterman database search tool by Torbjørn Rognes and available at <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 December 1st, 2014
First public release