vsearch(1) USER COMMANDS vsearch(1)

NAME

vsearch — a versatile open-source tool for microbiome analysis, including chimera detection, clustering, dereplication and rereplication, extraction, FASTA/FASTQ/SFF file processing, masking, orienting, pairwise alignment, restriction site cutting, searching, shuffling, sorting, subsampling, and taxonomic classification of amplicon sequences for metagenomics, genomics, and population genetics.

SYNOPSIS

Chimera detection:

vsearch (--uchime_denovo | --uchime2_denovo | --uchime3_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 | --cluster_unoise) fastafile (--alnout | --biomout | --blast6out | --centroids | --clusters | --mothur_shared_out | --msaout | --otutabout | --profile | --samout | --uc | --userout) outputfile --id real [options]

Dereplication and rereplication:

vsearch --fastx_uniques (fastafile | fastqfile) (--fastaout | --fastqout | --tabbedout | --uc) outputfile [options]

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

vsearch --derep_smallmem (fastafile | fastqfile) --fastaout outputfile [options]

vsearch --rereplicate fastafile --output outputfile [options]

Extraction of sequences:

vsearch --fastx_getseq fastafile (--fastaout | --fastqout | --notmatched | --notmatchedfq) outputfile --label label [options]

vsearch --fastx_getseqs fastafile (--fastaout | --fastqout | --notmatched | --notmatchedfq) outputfile (--label label --labels labelfile | --label_word label | --label_words labelfile) [options]

vsearch --fastx_getsubseq fastafile (--fastaout | --fastqout | --notmatched | --notmatchedfq) outputfile --label label [--subseq_start position] [--subseq_end position] [options]

```
       FASTA/FASTQ/SFF file processing:
              vsearch --fasta2fastq fastafile --fastqout outputfile [options]

              vsearch --fastq_chars fastqfile [options]

              vsearch --fastq_convert fastqfile --fastqout outputfile [options]

              vsearch (--fastq_eestats | --fastq_eestats2) fastqfile --output outputfile
[options]

              vsearch --fastq_filter fastqfile [--reverse fastqfile] (--fastaout | --
fastaout_dis-
              carded | --fastqout | --fastqout_discarded --fastaout_rev | --
fastaout_discarded_rev |
              --fastqout_rev | --fastqout_discarded_rev) outputfile [options]

              vsearch --fastq_join fastqfile --reverse fastqfile (--fastaout | --fastqout)
outputfile
              [options]

              vsearch --fastq_mergepairs fastqfile --reverse fastqfile (--fastaout | --fastqout
|
              --fastaout_notmerged_fwd | --fastaout_notmerged_rev | --fastqout_notmerged_fwd |
              --fastqout_notmerged_rev | --eetabbedout) outputfile [options]

              vsearch --fastq_stats fastqfile [--log logfile] [options]

              vsearch --fastx_filter inputfile [--reverse inputfile] (--fastaout | --
fastaout_dis-
              carded | --fastqout | --fastqout_discarded --fastaout_rev | --
fastaout_discarded_rev |
              --fastqout_rev | --fastqout_discarded_rev) outputfile [options]

              vsearch --fastx_revcomp inputfile (--fastaout | --fastqout) outputfile [options]

              vsearch --sff_convert sff-file --fastqout outputfile [options]

       Masking:
              vsearch --fastx_mask fastxfile (--fastaout | --fastqout) outputfile [options]

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

       Orienting:
              vsearch --orient fastxfile --db fastxfile (--fastaout | --fastqout | --notmatched
|
              --tabbedout) outputfile [options]

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

       Restriction site cutting:
              vsearch --cut fastafile --cut_pattern pattern (--fastaout | --fastaout_rev | --
fas-
              taout_discarded | --fastaout_discarded_rev) outputfile [options]

       Searching:
              vsearch --search_exact fastafile --db fastafile (--alnout | --biomout | --
blast6out |
```

```
              --mothur_shared_out | --otutabout | --samout | --uc | --userout | --lcaout)
outputfile
              [options]

              vsearch --usearch_global fastafile --db (fastafile | udbfile) (--alnout | --
biomout |
              --blast6out | --mothur_shared_out | --otutabout | --samout | --uc | --userout |
              --lcaout) outputfile --id real [options]

       Shuffling and sorting:
              vsearch (--shuffle | --sortbylength | --sortbysize) fastafile --output outputfile
[op-
              tions]

       Subsampling:
              vsearch --fastx_subsample fastafile (--fastaout | --fastqout) outputfile (--
sample_pct
              real | --sample_size positive integer) [options]

       Taxonomic classification:
              vsearch --sintax fastafile --db (fastafile | udbfile) --tabbedout outputfile [--
sin-
              tax_cutoff real] [options]

       UDB database handling:
              vsearch --makeudb_usearch fastafile --output outputfile [options]

              vsearch --udb2fasta udbfile --output outputfile [options]

              vsearch (--udbinfo | --udbstats) udbfile [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  pro-
       cessing.

       Comparing nucleotide sequences is at the core of vsearch. To speed up comparisons,
vsearch im-
       plements an extremely fast Needleman-Wunsch algorithm, making use of the Streaming SIMD
Exten-
       sions  (SSE2) of post-2003 x86-64 CPUs.  If SSE2 instructions are not available, vsearch
exits
       with an error message. On Power8 CPUs it will use AltiVec/VSX/VMX instructions, and  on
ARMv8
       CPUs  it  will  use Neon instructions. On other systems it can use the SIMD Everywhere
(simde)
       library, if available. Memory usage increases rapidly with sequence length: for
example  com-
       paring  two  sequences of length 1 kb requires 8 MB of memory per thread, and comparing
two 10
       kb sequences requires 800 MB of memory per thread. For comparisons involving sequences
with  a
```

length product greater than 25 million (for example two sequences of length 5 kb), vsearch
uses a slower alignment method described by Hirschberg (1975) and Myers and Miller (1988),
with much smaller memory requirements.

Input

vsearch accept as input fasta or fastq files containing one or several nucleotidic entries. In
fasta files, each entry is made of a header and a sequence. The header is defined as the
string comprised between the initial '>' symbol and the first space, tab or the end of the
line, unless the --notrunclabels option is in effect, in which case the entire line is in-
cluded. The header should contain printable ascii characters (33-126). The program will termi-
nate with a fatal error if there are unprintable ascii characters. A warning will be issued if
non-ascii characters (128-255) are encountered.

If the header matches the pattern '>[;]size=integer;label', the pattern '>label;size=inte-
ger;label', or the pattern '>label;size=integer[;]', vsearch will 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 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 warning message.

In fastq files, each entry is made of sequence header starting with a symbol '@', a nucleo-
tidic sequence (same rules as for fasta sequences), a quality header starting with a symbol
'+' and a string of ASCII characters (offset 33 or 64), each one encoding the quality value of
the corresponding position in the nucleotidic sequence.

vsearch operations are case insensitive, except when soft masking is activated. DUST masking
is automatically applied during chimera detection, clustering, masking, pairwise alignment and
searching. Soft masking is specified with the options '--dbmask soft' (for searching and
chimera detection with a reference) or '--qmask soft' (for searching, de novo chimera detec-
tion, 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 index words used for sequence comparisons, otherwise they are treated as normal symbols.

When comparing sequences during chimera detection, dereplication, searching and clustering, T and U are considered identical, regardless of their case. When aligning sequences, identical symbols will receive a positive match score (default +2). If two symbols are not identical, their alignment result in a negative mismatch score (default -4). Aligning a pair of symbols where at least one of them is an ambiguous symbol (BDHKMNRSVWY) will always result in a score of zero. Alignment of two identical ambiguous symbols (for example, R vs R) also receives a score of zero. When computing the amount of similarity by counting matches and mismatches af-ter alignment, ambiguous nucleotide symbols will count as matching to other symbols if they have at least one of the nucleotides (ACGTU) they may represent in common. For example: W will match A and T, but also any of MRVHDN. When showing alignments (for example with the --alnout option) matches involving ambiguous symbols will be shown with a plus character (+) between them while exact matches between non-ambiguous symbols will be shown with a vertical bar char-acter (|).

vsearch can read data from standard files and write to standard files, but it can also read from pipes and write to pipes! For example, multiple fasta files can be piped into vsearch for dereplication. To do so, file names can be replaced with:

- the symbol '-', representing '/dev/stdin' for input files or '/dev/stdout' for output files (with an exception for '--db -', see * below),

- a named pipe created with the command mkfifo,

- a process substitution '<(command)' as input or '>(command)' as output.

* --db - is not accepted, to prevent potential concurrent reads from stdin. A workaround for advanced users is to call '--db /dev/stdin' directly.

vsearch can automatically read compressed gzip or bzip2 files if the appropriate libraries are present during the compilation. vsearch can also read pipes streaming compressed gzip or bzip2 data if the options --gzip_decompress or --bzip2_decompress are selected. When reading from a pipe, the progress indicator is not updated.

Options

vsearch recognizes a large number of command-line commands and options. For easier navigation, options are grouped below by theme (chimera detection, clustering, dereplication and

rerepli-
cation, FASTA/FASTQ file processing, masking, pairwise alignment, searching, shuffling, sort-
ing, and subsampling). We start with the general options that apply to all themes. Options
start with a double dash (--). A single dash (-) may also be used, except on NetBSD systems.
Option names may be shortened as long as they are not ambiguous (e.g. --derep_f).

Help and version commands:

--help -h
Display help text with brief information about all commands and options.

--version -v
Output version information and a citation for the VSEARCH publication. Show
the status of the support for gzip- and bzip2-compressed input files.

General options:

--bzip2_decompress
When reading from a pipe streaming bzip2-compressed data, decompress the data.
This option is not needed when reading from a standard bzip2-compressed file.

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

--gzip_decompress
When reading from a pipe streaming gzip-compressed data, decompress the data.
This option is not needed when reading from a standard gzip-compressed file.

--label_suffix string
When writing FASTA or FASTQ files, add the suffix string to sequence headers.

--log filename
Write messages to the specified log file. Information written includes program
version, amount of memory available, number of cores and command line options,
and if need be, informational messages, warnings and fatal errors. The start
and finish times are also recorded as well as the elapsed time and the maximum
amount of memory consumed. The different vsearch commands can also write addi-
tional information to the log file.

--maxseqlength positive integer
All vsearch operations discard sequences longer than integer (50,000 nu-

cleotides by default).

--minseqlength positive integer
All vsearch operations discard sequences shorter than integer: 1 nucleotide by default for sorting or shuffling, 32 nucleotides for clustering and dereplication as well as the commands --makeudb_usearch, --sintax, and --usearch_global.

--no_progress
Do not show the gradually increasing progress indicator.

--notrunclabels
Do not truncate sequence labels at first space or tab, but use the full header in output files. Turned off by default for all commands except the sintax command.

--quiet Suppress all messages to stdout and stderr except for warnings and fatal error messages.

--sample string
When writing FASTA or FASTQ files, add the the given sample identifier string to sequence headers. For instance, if the given string is ABC, the text ";sample=ABC" will be added to the header. Note that string will be truncated at the first ';' or blank character. Other characters (alphabetical, numerical and punctuations) are accepted.

--threads positive integer
Number of computation threads to use (1 to 1024). The number of threads should be less than or equal to the number of available CPU cores. The default is to use all available resources and to launch one thread per core. The following commands are multi-threaded: allpairs_global, cluster_fast, cluster_size, cluster_smallmem, cluster_unoise, fastq_mergepairs, fastx_mask, maskfasta, search_exact, sintax, uchime_ref, and usearch_global. Only one thread is used for the other commands.

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).

Input sequences are masked as specified with the --qmask and --hardmask options.

Mask-
ing of the database for reference based chimera detection is specified with the --db-
mask option.

In de novo mode, input fasta file must present abundance annotations (i.e. a pattern
[;]size=integer[;] in the fasta header). Input order matters for 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
three-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. For --uchime3_denovo the
default value is 16.0. For the other commands, 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 equal or greater than 1.0 can be used.

--alignwidth positive integer
When using --uchimealns, set the width of the three-way alignments (80 nu-
cleotides by default). Set to zero to eliminate wrapping.

--borderline filename
Output borderline chimeric sequences to filename, in fasta format. Borderline
chimeric sequences are sequences that have a high enough score but which are
not sufficiently different from their closest parent.

--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 reference sequences con-
tained in filename. Reference sequences are assumed to be chimera-free.
Chimeras cannot be detected if their parents, or sufficiently close relatives,
are not present in the database. The file name must refer to a FASTA file or
to a UDB file. If a UDB file is used, it should be created using the
--makeudb_usearch command with the --dbmask dust option.

--dn strictly positive real number
pseudo-count prior on the number of no votes, corresponding to the parameter n in the chimera scoring function (default value is 1.4). Increasing --dn reduces the likelihood of tagging a sequence as a chimera (less false positives, but also more false negatives).

--fasta_score
Add the chimera score to the headers in the fasta output files for chimeras, non-chimeras and borderline sequences, using the format ';uchime_denovo=float;'.

--lengthout
Write sequence length information to the output files in FASTA format by adding a ";length=integer" attribute in the header.

--mindiffs positive integer
Minimum number of differences per segment (default value is 3). The parameter is ignored with --uchime2_denovo and --uchime3_denovo.

--mindiv real
Minimum divergence from closest parent (default value is 0.8). The parameter is ignored with --uchime2_denovo and --uchime3_denovo.

--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. The parameter is ignored with --uchime2_denovo and --uchime3_denovo.

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

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

--relabel_keep
When relabelling, keep the old identifier in the header after a space.

--relabel_md5
Relabel sequences using the MD5 message digest algorithm applied to each sequence. Former sequence headers are discarded. The sequence is converted to upper case and each 'U' is replaced by a 'T' before computation of the

digest.
The MD5 digest is a cryptographic hash function designed to minimize the prob-
ability that two different inputs give the same output, even for very similar,
but non-identical inputs. Still, there is a very small, but non-zero, proba-
bility that two different inputs give the same digest (i.e. a collision). MD5
generates a 128-bit (16-byte) digest that is represented by 16 hexadecimal
numbers (using 32 symbols among 0123456789abcdef). Use --sizeout to conserve
the abundance annotations.

--relabel_self
Relabel sequences using each sequence itself as a label.

--relabel_sha1
Relabel sequences using the SHA1 message digest algorithm applied to each se-
quence. It is similar to the --relabel_md5 option but uses the SHA1 algorithm
instead of the MD5 algorithm. SHA1 generates a 160-bit (20-byte) digest that
is represented by 20 hexadecimal numbers (40 symbols). The probability of a
collision (two non-identical sequences resulting in the same digest) is
smaller for the SHA1 algorithm than it is for the MD5 algorithm.

--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 se-
quence is strictly identical to the nucleotidic sequence of the query.

--sizein In de novo mode, abundance annotations (pattern '[>;]size=integer[;]') present
in sequence headers are taken into account by default (--sizein is always im-
plied). This option is ignored by --uchime_ref.

--sizeout
When relabelling, add abundance annotations to fasta headers (using the format
';size=integer;').

--uchime_denovo filename
Detect chimeras present in the fasta-formatted filename, without external ref-
erences (i.e. de novo). Automatically sort the sequences in filename by de-
creasing abundance beforehand (see the sorting section for details). Multi-
threading is not supported.

--uchime2_denovo filename
Detect chimeras present in the fasta-formatted filename, using the UCHIME2 algorithm. This algorithm is designed for denoised amplicons (see --cluster_unoise). Automatically sort the sequences in filename by decreasing abundance beforehand (see the sorting section for details). Multithreading is not supported.

--uchime3_denovo filename
Detect chimeras present in the fasta-formatted filename, using the UCHIME2 algorithm. The only difference from --uchime2_denovo is that the default minimum abundance skew (--abskew) is set to 16.0 rather than 2.0.

--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 three-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. All sequences are converted to upper case before alignment. Lower case letters indicate disagreement in the alignment.

--uchimeout filename
Write chimera detection results to filename using a 18-field, tab-separated uchime-like format. 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 17-field, tab-separated uchime-like format (drop the 5th field of --uchimeout), compatible with usearch version 5 and earlier versions.

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute is added to output sequences by the --lengthout option.

--xn strictly positive real number
weight of no votes, corresponding to the parameter beta in the scoring function (default value is 8.0). Increasing --xn reduces the likelihood of tagging a sequence as a chimera (less false positives, but also more false negatives).

--xsize Strip abundance information from the headers when writing the output file.

Clustering options:

vsearch implements a single-pass, greedy centroid-based 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).

Input sequences are masked as specified with the --qmask and --hardmask options.

--biomout filename
Generate an OTU table in the biom version 1.0 JSON file format as specified at (link) ⟨https://biom-format.org/documentation/format_versions/biom-1.0.html⟩ <https://biom-format.org/documentation/format_versions/biom-1.0.html>. The format describes how to store a sparse matrix containing the abundances of the OTUs in the different samples. This format is much more efficient than the classic and mothur OTU table formats available with the --otutabout and --mothur_shared_out options, respectively, and is recommended at least for large tables. The OTUs are represented by the cluster centroids. Taxonomy information will be included for the OTUs if available. Sample identifiers will be extracted from the headers of all sequences in the input file. If the header contains ';sample=abc123;' or ';barcodelabel=abc123;' or a similar string somewhere, then the given sample identifier (here 'abc123') will be used. The semicolon is not mandatory at the beginning or end of the header. The sample identifier may contain any printable character except semicolons. If no such sample label is found, the identifier in the initial part of the header will be used, but only letters, digits and underscores are allowed. OTU identifiers will be extracted from the headers of the cluster centroid sequences. If the header contains ';otu=def789;' or a similar string somewhere, then the given OTU identifier (here 'def789') will be used. The semicolon is not mandatory at the beginning or end of the header. The OTU identifier may contain any printable character except semicolons. If no such OTU label is found, the identifier in the initial part of the header will be used, and all characters except semicolons are allowed. Alternatively, OTU identifiers can be generated using the relabelling options (--relabel, --relabel_self, --relabel_sha1, or --relabel_md5). Taxonomy information, if present, will also be extracted from the headers of the centroid sequences. If the header contains ';tax=Homo_sapiens;' or a similar string somewhere, then the given taxonomy information (here 'Homo_sapiens') will be used. The semicolon is not mandatory at the beginning or end of the header. The taxonomy information may contain

any printable character except semicolons. If an OTU table in the biom version
2.1 HDF5 file format is required, the biom utility may be used as described at
(link) ⟨https://biom-format.org/documentation/biom_conversion.html⟩
<https://biom-format.org/documentation/biom_conversion.html>.

--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 clus-
ter).

--clusterout_id
Add cluster identifier information to the output files when using the --cen-
troids, --consout and --profile options.

--clusterout_sort
Sort some output files by decreasing abundance instead of input order. It ap-
plies to the --consout, --msaout, --profile, --centroids, and --uc options.
For --uc, the sorting applies only to the centroid information part (the C
lines).

--cluster_fast filename
Clusterize the fasta sequences in filename, automatically sort by decreasing
sequence length beforehand.

--cluster_size filename
Clusterize the fasta sequences in filename, automatically sort 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 se-
quence length, unless --usersort is used.

--cluster_unoise filename
Perform denoising of the fasta sequences in filename according to the UNOISE
version 3 algorithm by Robert Edgar, but without the de novo chimera removal
step, which may be performed afterwards with --uchime3_denovo. The options
--minsize (default 8) and --unoise_alpha (default 2.0) may be specified. In
the this algorithm, clustering of sequences depend on both the sequence dis-
tance and the abundance ratio. The abundance ratio (skew) is the abundance of
a new sequence divided by the abundance of the centroid sequence.

This skew must not be larger than beta if the sequences should be clustered together. Beta is calculated as 2 raised to the power of minus 1 minus alpha times the sequence distance. The sequence distance used is the number of mismatches in the alignment, ignoring gaps. This means that the abundance must be exponen- tially lower as the distance increases from the centroid for a new sequence to be included in the cluster. Nearer sequences with higher abundances will form their own new clusters.

--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 center- star multiple sequence alignment is computed with the centroid as the center, using a fast algorithm (not accurate when using low pairwise identity thresh- olds). A consensus sequence is constructed by taking the majority symbol (nu- cleotide or gap) from each column of the alignment. Columns containing a ma- jority of gaps are skipped, except for terminal gaps. If the --sizein option is specified, sequence abundances will be taken into account.

--cons_truncate
This command is ignored. A warning is issued.

--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 ex- tending 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 con- texts: 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 sym- bols (Q and T) can be combined with location symbols (L, I, and R), and numer- ical 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 '*'. To 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.

--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 gap opening (internal or terminal) as a single mismatch, whether or not the

gap was
extended: 1.0 - [(mismatches + gap openings)/(longest sequence
length)]

4. BLAST definition, equivalent to --iddef 1 in a context of global
pairwise alignment.

--lengthout
Write sequence length information to the output files in FASTA format by
adding a ";length=integer" attribute in the header.

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

--minsize positive integer
Specify the minimum abundance of sequences for denoising using --cluster_un-
oise. The default is 8.

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

--msaout filename
Output a multiple sequence alignment and a consensus sequence for each cluster
to filename, in fasta format. Be warned that vsearch computes center star mul-
tiple sequence alignments using a fast method whose accuracy can decrease sig-
nificantly when using low pairwise identity thresholds. 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. If the --sizein option is specified, sequence abun-
dances will be taken into account when computing the consensus.

--mothur_shared_out filename
Output an OTU table in the mothur 'shared' tab-separated plain text format as
described at (link) ⟨https://www.mothur.org/wiki/Shared_file⟩
<https://www.mothur.org/wiki/Shared_file>. The format describes how a matrix
containing the abundances of the OTUs in the different samples is stored. The
first line will start with the strings 'label', 'group' and 'numOtus' and is
followed by a list of all OTU identifiers. The following lines, one for each
sample, starts with the string 'vsearch' followed by the sample

identifier,
the total number of OTUs, and a list of abundances for each OTU in
that sam-
ple, in the order given on the first line. The OTU and sample
identifiers are
extracted from the FASTA headers of the sequences. The OTUs are
represented by
the cluster centroids. See the --biomout option for further details.

--otutabout filename
Output an OTU table in the classic tab-separated plain text format as a
matrix
containing the abundances of the OTUs in the different samples. The
first line
will start with the string '#OTU ID' and is followed by a tab-
separated list
of all sample identifiers. The following lines, one for each OTU,
starts with
the OTU identifier and is followed by a tab-separated list of
abundances for
that OTU in each sample, in the order given on the first line. The
OTU and
sample identifiers are extracted from the FASTA headers of the
sequences (see
the --sample option). The OTUs are represented by the cluster
centroids. An
extra column is added to the right of the table if taxonomy
information is
available for at least one of the OTUs. This column will be labelled
'taxon-
omy' and each row will then contain the taxonomy information
extracted for
that OTU. See the --biomout option for further details.

--profile filename
Output a sequence profile to a text file with the frequency of each
nucleotide
in each position in the multiple alignment for each cluster. There is a
FASTA-
like header line for each cluster, followed by the profile information
in a
tab-separated format. The eight columns are: position (0-based),
consensus nu-
cleotide, number of As, number of Cs, number of Gs, number of Ts or Us,
number
of gap symbols, and finally the total number of ambiguous nucleotide
symbols
(B, D, H, K, M, N, R, S, Y, V or W). All numbers are integers. If the --
sizein
option is specified, sequence abundances will be taken into account.

--qmask none|dust|soft
Mask regions in sequences using the dust or the soft methods, or do
not mask
(none). Warning, when using soft masking, clustering becomes case
sensitive.
The default is to mask using dust.

--qsegout filename
Write the aligned part of each query sequence to filename in FASTA

format.

--relabel string
Relabel sequence identifiers in the output files produced by --consout, --pro-
file and --centroids options. Please see the description of the same option
under Chimera detection for details.

--relabel_keep
When relabelling, keep the old identifier in the header after a space.

--relabel_md5
Relabel sequence identifiers in the output files produced by --consout, --pro-
file and --centroids options. Please see the description of the same option
under Chimera detection for details.

--relabel_self
Relabel sequence identifiers in the output files produced by --consout, --pro-
file and --centroids options. Please see the description of the same option
under Chimera detection for details.

--relabel_sha1
Relabel sequence identifiers in the output files produced by --consout, --pro-
file and --centroids options. Please see the description of the same option
under Chimera detection for details.

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

--sizeorder
When an amplicon is close to 2 or more centroids, both within the distance
specified with the --id option, resolve the ambiguity by clustering it with
the centroid having the highest abundance, not necessarily the closest one.
The option only has effect when the value specified with --maxaccepts is
higher than one. The --sizeorder option turns on what is sometimes referred to
as abundance-based greedy clustering (AGC), in contrast to the default dis-
tance-based greedy clustering (DGC).

--sizeout
Add abundance annotations to the output fasta files (add the pattern
';size=integer;' to sequence headers). If --sizein is specified, abundance an-
notations are reported to output files, and each cluster centroid receives a
new abundance value corresponding to the total abundance of the

amplicons in-
cluded in the cluster (--centroids option). If --sizein is not
specified, in-
put 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.

--tsegout filename
Write the aligned part of each target sequence to filename in FASTA
format.

--uc filename
Output clustering results in filename using a tab-separated uclust-like
format
with 10 columns and 3 different type of entries (S, H or C). Each
fasta se-
quence in the input file can be either a cluster centroid (S) or a hit
(H) as-
signed to a cluster. Cluster records (C) summarize information (size,
centroid
label) for each cluster. In the context of clustering, the option --
uc_allhits
has no effect on the --uc output. Column content varies with the type of
entry
(S, H or C):

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

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

--unoise_alpha real
Specify the alpha parameter to the --cluster_unoise command. The default is 2.0.

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

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute is added to output sequences by the --lengthout option.

--xsize Strip abundance information from the headers when writing the output file.

... Most searching options as well as score filtering, gap penalties and masking also apply to clustering (see the Searching section for definitions): --alnout, --blast6out, --fastapairs, --matched, --notmatched, --maxaccepts, --maxrejects, --samout, --userout, --userfields

Dereplication and rereplication options:

VSEARCH can dereplicate sequences with the commands --derep_fulllength, --derep_id, --derep_smallmem, --derep_prefix and --fastx_uniques. The --derep_fulllength command is depreciated and is replaced by the new --fastx_uniques command that can also handle FASTQ files in addition to FASTA files. The --derep_fulllength, --derep_smallmem, and --fastx_uniques commands requires strictly identical sequences of the same length, but ignores upper/lower case and treats T and U as identical symbols. The --derep_id command requires both identical sequences and identical headers/labels. The --derep_prefix command will group sequences with a common prefix and does not require them to be equally long. The --derep_smallmem uses a much smaller amount of memory when dereplicating than the other files, and may be a bit slower and cannot read the input from a pipe. It takes both FASTA and FASTQ files as input but only writes FASTA output to the file specified with the --fastaout option. The --fastx_uniques command can write FASTQ output (specified with --fastqout) or FASTA output (specified with --fastaout) as well as a special tab-separated column text format (with --tabbedout). The other commands can write FASTA output to the file specified with the --output option. All

dereplica-
tion commands, except --derep_smallmem, can write output to a special UCLST-like file
specified with the --uc option. The --rereplicate command can duplicate sequences in
the input file according to the abundance of each input sequence. Other valid options
are --fastq_ascii, --fastq_asciiout, --fastq_qmax, --fastq_qmaxout, --fastq_qmin,
--fastq_qminout, --fastq_qout_max, --lengthout, --maxuniquesize, --minuniquesize, --re-
label, --relabel_keep, --relabel_md5, --relabel_self, --relabel_sha1, --sizein, --size-
out, --strand, --topn, --xlength, and --xsize.

--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). See the options --sizein and
--sizeout to take into account and compute abundance values. This command does
not support multithreading.

--derep_id filename
Merge strictly identical sequences contained in filename, as with the
--derep_fulllength command, but the sequence labels (identifiers) on the
header line need to be identical too.

--derep_smallmem filename
Merge strictly identical sequences contained in filename, as with the
--derep_fulllength command, but using much less memory. The output is written
to a FASTA file specified with the --fastaout option. The output is written in
the order that the sequences first appear in the input, and not in descending
abundance order as with the other dereplication commands. It can read, but not
write FASTQ files. This command cannot read from a pipe, it must be a proper
file, as it is read twice. Dereplication is performed with a 128 bit hash
function and it is not verified that grouped sequences are identical, however
the probability that two different sequences are grouped in a dataset of one
billion unique sequences is approximately 1e-21. Memory footprint is appr. 24
bytes times the number of unique sequence. Multithreading and the options
--topn, --uc, or --tabbedout are not supported.

--derep_prefix filename
Merge sequences with identical prefixes contained in filename. A

short se-
quence identical to an initial segment (prefix) of another sequence is
consid-
ered a replicate of the longer sequence. If a sequence is identical
to the
prefix of two or more longer sequences, it is clustered with the
shortest of
them. If they are equally long, it is clustered with the most
abundant. Re-
maining ties are solved using sequence headers and sequence input
order. Se-
quence comparisons are case insensitive, and T and U are considered
identical.
This command does not support multithreading.

--fastaout filename
Write the dereplicated sequences to filename, in fasta format and
sorted by
decreasing abundance. Identical sequences receive the header of the
first se-
quence 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 us-
ing the pattern ';size=integer;'. This option is only
valid for
--fastx_uniques and --derep_smallmem.

--fastqout filename
Write the dereplicated sequences to filename, in fastq format and
sorted by
decreasing abundance. Identical sequences receive the header of the
first se-
quence of their group. If --sizeout is used, the number of occurrences
(i.e.
abundance) of each sequence is indicated at the end of their fastq
header us-
ing the pattern ';size=integer;'. This option is only
valid for
--fastx_uniques.

--fastq_ascii positive integer
Define the ASCII character number used as the basis for the FASTQ
quality
score. The default is 33, which is used by the Sanger / Illumina 1.8+
FASTQ
format (phred+33). The value 64 is used by the Solexa, Illumina 1.3+ and
Illu-
mina 1.5+ formats (phred+64). Only 33 and 64 are valid arguments.

--fastq_asciiout positive integer
When using --fastq_convert, --sff_convert or --fasta2fastq, define the
ASCII
character number used as the basis for the FASTQ quality score when
writing
FASTQ output files. The default is 33. Only 33 and 64 are valid
arguments.

--fastq_qmax positive integer
Specify the maximum quality score accepted when reading FASTQ files.

The de-
fault is 41, which is usual for recent Sanger/Illumina 1.8+ files.

--fastq_qmaxout positive integer
Specify the maximum quality score used when writing FASTQ files. The default
is 41, which is usual for recent Sanger/Illumina 1.8+ files. Older formats may
use a maximum quality score of 40.

--fastq_qmin positive integer
Specify the minimum quality score accepted for FASTQ files. The default is 0,
which is usual for recent Sanger/Illumina 1.8+ files. Older formats may use
scores between -5 and 2.

--fastq_qminout positive integer
Specify the minimum quality score used when writing FASTQ files. The default
is 0, which is usual for Sanger/Illumina 1.8+ files. Older versions of the
format may use scores between -5 and 2.

--fastq_qout_max
For --fastx_uniques, indicate that the new quality scores computed when
dereplicating FASTQ files should be equal to the maximum (best) of the input
quality scores for each position (corresponding to the lowest error probabil-
ity). The default is to output a quality score corresponding to the average of
the error probabilities for each position.

--fastx_uniques filename
Merge strictly identical sequences contained in FASTA or FASTQ file 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). See the
options --sizein and --sizeout to take into account and compute abundance val-
ues. This command does not support multithreading. By default, the quality
scores in FASTQ output files will correspond to the average error probability
of the nucleotides in the each position. If the --fastq_qout_max option is
given, the quality score will be the highest (best) quality score observed in
each position.

--lengthout
Write sequence length information to the output files in FASTA and FASTQ for-
mat by adding a ";length=integer" attribute in the header.

--maxuniquesize positive integer

Discard sequences with a post-dereplication abundance value greater than inte-
ger.

--minuniquesize positive integer
Discard sequences with a post-dereplication abundance value smaller than inte-
ger.

--output filename
Write the dereplicated sequences to filename, in fasta format and sorted by
decreasing abundance. Identical sequences receive the header of the first se-
quence 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 us-
ing the pattern ';size=integer;'. This option is not allowed for
--fastx_uniques or --derep_smallmem.

--relabel string
Please see the description of the same option under Chimera detection for de-
tails.

--relabel_keep
When relabelling, keep the old identifier in the header after a space.

--relabel_md5
Please see the description of the same option under Chimera detection for de-
tails.

--relabel_self
Please see the description of the same option under Chimera detection for de-
tails.

--relabel_sha1
Please see the description of the same option under Chimera detection for de-
tails.

--rereplicate filename
Duplicate each sequence the number of times indicated by the abundance of each
sequence in the specified file (option --sizein is always implied). The se-
quence labels are identical for the same sequence, unless --relabel,
--rela-
bel_self, --relabel_sha1 or --relabel_md5 is used to create unique labels.
Output is written to the file specified with the --output option, in FASTA
format. The output file does not contain abundance information unless
--size-
out is specified, in which case an abundance of 1 is used.

--sizein Take into account the abundance annotations present in the input fasta file
(search for the pattern '[>;]size=integer[;]' in sequence headers). That option is active by default when rereplicating.

--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.

--tabbedout filename
Output clustering info to the specified tab-separated text file with 6 columns and a row for each input sequence. Column 1 contains the original label/header of the sequence. Column 2 contains the label of the output sequence which is equal to the label/header of the first sequence in each cluster, but potentially relabelled. Column 3 contains the cluster number, starting from 0. Column 4 contains the sequence number within each cluster, starting at 0. Column 5 contains the number of sequences in the cluster. Column 6 contains the original label/header of the first sequence in the cluster before any potential relabelling. This option is only valid for the --fastx_uniques command.

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

--uc filename
Output full-length or prefix-dereplication results in filename using a tab-separated uclust-like format with 10 columns and 3 different type of entries (S, H or C). Each fasta sequence in the input file can be either a centroid (S) or a hit (H) assigned to a cluster. Cluster records (C) summarize information (size, centroid label) for each cluster. In the context of dereplication, the option --uc_allhits has no effect on the --uc output. Column content varies with the type of entry (S, H or C):

1. Record type: S, H, or C.
2. Cluster number (zero-based).
3. Sequence length (S, H), or cluster size (C).
4. Percentage of similarity with the centroid sequence (H), or set to '*' (S, C).
5. Match orientation + or - (H), or set to '*' (S, C).
6. Not used, always set to '*' (S, C) or 0 (H).
7. Not used, always set to '*' (S, C) or 0 (H).
8. Not used, always set to '*'.
9. Label of the query sequence (H), or of the centroid sequence (S, C).
10. Label of the centroid sequence (H), or set to '*' (S, C).

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute is added to output sequences by the --lengthout option.

--xsize
Strip abundance information from the headers when writing the output file.

Extraction options:

Sequences with headers matching certain criteria can be extracted from FASTA and FASTQ files using the --fastx_getseq, --fastx_getseqs and --fastx_getsubseq commands.

The --fastx_getseq command requires the header to match a label specified with the --label option. If the --label_substr_match option is given, the label may be a substring located anywhere in the header, otherwise the entire header must match the label. These matches are not case-sensitive. The headers in the input file are truncated at the first space or tab character unless the --notrunclabels option is given. The matching sequences will be written to the files specified with the --fastaout and --fastqout options, in FASTA and FASTQ format, respectively. Sequences that do not match are written to the files specified with the --notmatched and --notmatchedfq options, respectively.

The --fastx_getsubseq command is similar to the --fastx_getseq command, but will extract a subsequence of the matching sequences. The start position is specified

with the
--subseq_start option and the end position is specified with the --subseq_end
option.
The positions are 1-based, meaning that the first symbol of the sequence is at
position
1. If the start or end position option is not specified, the default is to start
at the
first position and end at the last position in the sequence.

The --fastx_getseqs command is similar to the --fastx_getseq command but
allows more
flexibility in specifying the label(s) to be matched. A single label may be
specified
using the --label option as described above. Alternatively, a file containing a
list of
labels to be matched may be specified with the --labels option. The file
must be a
plain text file with one label on each line. The --label_word and --label_words
options
may be used to specify either a single word or a file containing a list of
words, re-
spectively, to be matched. Words are defined as character sequences delimited
either by
a character that is not alpha-numeric (A-Z, a-z, or 0-9) or by the beginning or
end of
the header. Word matching is case-sensitive. The --label_field option will
limit the
matching of words to a certain field in the header.

--fastaout filename
Write the extracted sequences in FASTA format to the file with the given
name.

--fastqout filename
Write the extracted sequences in FASTQ format to the file with the given
name.
This option is illegal if the input is in FASTA format.

--fastx_getseq filename
Extract sequences from the given FASTA or FASTQ file. Specify a label to
match
using the --label option. Output files are specified with the --
fastaout,
--fastqout, --notmatched and --notmatchedfq options.

--fastx_getseqs filename
Extract sequences from the given FASTA or FASTQ file. Specify the label
or la-
bels to match using one of the following options: --label, --labels,
--la-
bel_word, or --label_words. Output files are specified with the --
fastaout,
--fastqout, --notmatched and --notmatchedfq options.

--fastx_getsubseq filename
Extract a certain part of some of the sequences in the given FASTA or
FASTQ
file. Specify labels to match using the --label option. Specify the
subse-
quence range to be extracted with the --subseq_start and --subseq_end

options.
Output files are specified with the --fastaout, --fastqout, --notmatched and
--notmatchedfq options.

--label string
Specify the label to match in the sequence header. Unless the --label_sub-
str_match option is given, the label must match the entire header. The compar-
ison is not case-sensitive.

--label_field string
Specify a field name to be used when matching using the --label_word or --la-
bel_words option. The field name is a string like "abc" that must precede the
word to be matched with an equals sign (=) in between. The field must be de-
limited by semicolons or the beginning or end of the header. The following
header will match the label 123 in the field abc: "seq1;abc=123".

--label_substr_match
The labels specified with the --label or the --labels option may match any-
where in the header if this option is given. Otherwise a label needs to match
the entire header.

--label_word string
Specify a word to match in the sequence header. Words are defined as strings
delimited by either the start or end of the header or by any symbol that is
not a letter (A-Z, a-z) or digit (0-9). The comparison is case-sensitive.

--label_words filename
Specify a file containing words to be matched against the sequence headers.
The plain text file must contain one word on each line. Words are defined as
strings delimited by either the start or end of the header or by any symbol
that is not a letter (A-Z, a-z) or digit (0-9). The comparison is case-sensi-
tive.

--labels filename
Specify a file containing labels to be matched against the sequence headers.
The plain text file must contain one label on each line. Unless the --la-
bel_substr_match option is given, a label must match the entire header. The
comparison is not case-sensitive.

--notmatched filename
Write the sequences that were not extracted to the file with the given

name,
in FASTA format.

--notmatchedfq filename
Write the sequences that were not extracted to the file with the given name,
in FASTQ format. This option is illegal if the input is in FASTA format.

--subseq_end positive integer
Specify the end position in the sequences when extracting subsequences using
the --fastx_getsubseq command. Positions are 1-based, so the sequences start
at position 1. The default is to end at the end of the sequence if this option
is not specified.

--subseq_start positive integer
Specify the starting position in the sequences when extracting subsequences
using the --fastx_getsubseq command. Positions are 1-based, so the sequences
start at position 1. The default is to start at the beginning of the sequence
(position 1), if this option is not specified.

FASTA/FASTQ/SFF file processing options:

Analyse, trim, filter, convert, merge, join or reverse complement sequences in FASTA,
FASTQ or SFF files. The --fastq_chars command can be used to analyse FASTQ files to
identify the quality encoding and the range of quality score values used. To convert
between different FASTQ file variants, use the --fastq_convert command. Statistical
analysis of the quality and length of the sequences in a FASTQ file may be performed
with the --fastq_stats, --fastq_eestats, and --fastq_eestats2 commands. Sequences may
be trimmed, filtered and converted by the --fastq_filter or --fastx_filter commands.
The --sff_convert command can be used to convert SFF files to FASTQ, while the
--fasta2fastq command will convert a FASTA file to a FASTQ file with fake quality
scores. Paired-end reads can be merged using the --fastq_mergepairs command or joined
with the --fastq_join command. The --fastx_revcomp command will reverse-complements
sequences.

--eeout When using --fastq_filter, --fastx_filter or --fastq_mergepairs, include the
number of expected errors (ee) in the sequence header of FASTQ and FASTA out-
put files. This option is a synonym of the --fastq_eeout option. Use the --xee
option to remove this information from headers.

--eetabbedout filename
When specified with the --fastq_mergepairs command, write statistics with expected errors of each merged read to the given file. The file is a tab separated file with four columns: The number of expected errors in the forward read, the number of expected errors in the reverse read, the number of observed errors in the forward read, and the number of observed errors in the reverse read. The observed number of errors are the number of differences in the overlap region of the merged sequence relative to each of the reads in the pair.

--fasta2fastq filename
Add a fake nucleotide quality score to the sequences in the given FASTA file and write them to the FASTQ file specified with the --fastqout option. The quality score may be adjusted using the --fastq_qmaxout option (default 41). The --fastq_asciiout option may be used to adjust the FASTQ output quality ASCII base character (default 33).

--fastaout filename
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, write to the given FASTA-formatted file the sequences passing the filter, or the merged sequences.

--fastaout_rev filename
When using --fastq_filter, or --fastx_filter, write to the given FASTA-formatted file the reverse reads passing the filter.

--fastaout_notmerged_fwd filename
When using --fastq_mergepairs, write forward reads not merged to the specified FASTA file.

--fastaout_notmerged_rev filename
When using --fastq_mergepairs, write reverse reads not merged to the specified FASTA file.

--fastaout_discarded filename
Write sequences that do not pass the filter of the --fastq_filter or --fastx_filter command to the given FASTA-formatted file.

--fastaout_discarded_rev filename
Write reverse reads that do not pass the filter of the --fastq_filter or --fastx_filter command to the given FASTA-formatted file.

--fastq_allowmergestagger
When using --fastq_mergepairs, allow merging of staggered read pairs. Stag-
gered pairs are pairs where the 3' end of the reverse read has an overhang to
the left of the 5' end of the forward read. This situation can occur when a
very short fragment is sequenced. The 3' overhang of the reverse read is not
included in the merged sequence. The opposite option is the --fastq_nostagger
option. The default is to discard staggered pairs.

--fastq_ascii positive integer
Define the ASCII character number used as the basis for the FASTQ quality
score. The default is 33, which is used by the Sanger / Illumina 1.8+ FASTQ
format (phred+33). The value 64 is used by the Solexa, Illumina 1.3+ and Illu-
mina 1.5+ formats (phred+64). Only 33 and 64 are valid arguments.

--fastq_asciiout positive integer
When using --fastq_convert, --sff_convert or --fasta2fastq, define the ASCII
character number used as the basis for the FASTQ quality score when writing
FASTQ output files. The default is 33. Only 33 and 64 are valid arguments.

--fastq_chars filename
Summarize the composition of sequence and quality strings contained in the in-
put FASTQ file. For each sequence symbol, --fastq_chars gives the number of
occurrences of the symbol, its relative frequency and the length of the
longest run of that symbol. For each character present in the quality strings,
--fastq_chars gives the ASCII value of the character, its relative frequency,
and the number of times a k-mer of that character appears at the end of qual-
ity strings. The length of the k-mer can be set using --fastq_tail (4 by de-
fault). The command --fastq_chars tries to automatically detect the quality
encoding (Solexa, Illumina 1.3+, Illumina 1.5+ or Illumina 1.8+/Sanger) by an-
alyzing the range of observed quality score values. In case of success,
--fastq_chars suggests values for the --fastq_ascii (33 or 64), --fastq_qmin
and --fastq_qmax options to be used with the other commands that require a
FASTQ input file.

--fastq_convert filename
Convert between the different variants of the FASTQ file format. The quality

encoding of the input file must be specified with the --fastq_ascii option (either 33 or 64, the default is 33), and the output quality encoding must be specified with the --fastq_asciiout option (default 33). The minimum and maximum output quality scores may be limited using the --fastq_qminout and --fastq_qmaxout options. The output file is specified with the --fastqout option.

--fastq_eeout
When using --fastq_filter, --fastx_filter or --fastq_mergepairs, include the number of expected errors (ee) in the sequence header of FASTQ and FASTA files. This option is a synonym of the --eeout option. Use the --xee option to remove this information from headers.

--fastq_eestats filename
Analyze a FASTQ file and report statistics on the distributions of quality scores, error probabilities and expected accumulated errors. The report, a table of 21 tab-separated columns, is written to the file specified with the --output option. The first column corresponds to the position in the reads (Pos). The second and third columns correspond to the number of reads (Reads) and percentage of reads (PctRecs) that include this position. The remaining columns include information about the distribution of quality scores in this position (Q), error probabilities in this position (Pe), and finally the expected number of accumulated errors from the beginning of the reads and until the current position (EE). For each of the Q, Pe and EE distributions, the following statistics are included: minimum value (Min), lower quartile (Low), median (Med), mean (Mean), upper quartile (Hi), and maximum value (Max). The quality encoding and the range of quality values may be specified with --fastq_ascii --fastq_qmin and --fastq_qmax.

--fastq_eestats2 filename
Analyze the specified FASTQ file and report statistics on the number of sequences that would be retained at a combination of selected cutoffs for length truncation and maximum expected errors, that could potentially be used as arguments to the --fastq_trunclen and --fastq_maxee options to the --fastq_filter command. The result, a table of two or more columns, is written

to the
file specified with the --output option. There is a line for each length
trun-
cation cutoff. The first column on each line contains the selected
truncation
length, while the following columns contain the number of sequences
and, in
parenthesis, the percentage of sequences that would be retained at
the se-
lected EE levels. The truncation length cutoffs may be specified
with the
--length_cutoffs option and requires a list of three comma-separated
integers
indicating the shortest cutoff, the longest cutoff, and the increment
between
cutoffs. The longest cutoff may be specified with a star (*) which
indicates
that the limit is equal to the longest sequence in the input file. The
default
setting is "50,*,50" meaning that truncation lengths of 50, 100, 150 and
so on
up to the longest sequence length should be used. The maximum expected
error
(EE) cutoffs may be specified with the --ee_cutoffs option which
requires a
comma-separated list of floating point numbers as its argument. The
default
setting is "0.5,1.0,2.0" that indicates that expected error levels of
0.5, 1.0
and 2.0 should be used.

--fastq_filter filename
Trim and/or filter sequences in the given FASTQ file. Similar
to the
--fastx_filter command, but works only on FASTQ files. See --
fastx_filter for
details.

--fastq_join filename
Join paired-end sequence reads into one sequence and add a gap
between them
using a padding sequence. The sequences are not merged as
with the
fastq_mergepairs command, but simply joined with a gap. The forward
reads are
specified as the argument to this option and the reverse reads are
specified
with the --reverse option. The resulting sequences consist of the
forward
read, the padding sequence and the reverse complement of the reverse
read. The
padding sequence is specified with the --join_padgap option and the
padding
quality is specified with the --join_padgapq option. The default
padding se-
quence string is NNNNNNNN and the default padding quality string is
IIIIIIII,
corresponding to a base quality score of 40 (a very high quality
score with
error probability 0.0001). The joined sequences are output to the

file(s)
specified with the --fastaout or --fastqout options.

--fastq_maxdiffs positive integer
When using --fastq_mergepairs, specify the maximum number of non-matching nucleotides allowed in the overlap region. That option has a strong influence on the merging success rate. The default value is 10.

--fastq_maxdiffpct real
When using --fastq_mergepairs, specify the maximum percentage of non-matching nucleotides allowed in the overlap region. The default value is 100.0%. There are other more sophisticated rules in the merging algorithm that will discard read pairs with a high fraction of mismatches.

--fastq_maxee real
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, discard sequences with an expected error greater than the specified number (value ranging from 0.0 to infinity). For a given sequence, the expected error is the sum of error probabilities for all the positions in the sequence. Since error probabilities can be small but not null, the expected error is always greater than zero, and at most equal to the length of the sequence when all positions in the sequence have an error probability of 1.0.

Using the expected error as the lambda parameter in the Poisson distribution, it is possible to compute the probability of observing k errors. For instance, a read with an expected error of 1.0 has:

- 36.8% chance of having zero error,
- 36.8% chance of having one error,
- 18.4% chance of having two errors,
- 6.1% chance of having three errors,
- 1.5% chance of having four errors,
- 0.3% chance of having five errors,
- etc.

--fastq_maxee_rate real
When using --fastq_filter or --fastx_filter, discard sequences with an average expected error greater than the specified number (value ranging from 0.0 to 1.0 included). For a given sequence, the average expected error is the sum of

error
probabilities for all the positions in the sequence, divided by the length of
the sequence.

--fastq_maxlen positive integer
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, discard se-
quences with more than the specified number of bases.

--fastq_maxmergelen positive integer
When using --fastq_mergepairs, specify the maximum length of the merged sequence
(default is 1,000,000).

--fastq_maxns positive integer
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, discard se-
quences with more than the specified number of N's.

--fastq_mergepairs filename
Merge paired-end sequence reads into one sequence. The forward reads are speci-
fied as the argument to this option and the reverse reads are specified with the
--reverse option. Reads with the same index/position in the forward and reverse
files are considered to form a pair, even if their labels are different. Thus,
forward and reverse reads must appear in the same order and total number in both
files. A warning is emitted if the forward and reverse files contain different
numbers of reads. The merged sequences are written to the file(s) specified with
the --fastaout or --fastqout options. The non-merged reads can be output to the
files specified with the --fastaout_notmerged_fwd, --fastaout_notmerged_rev,
--fastqout_notmerged_fwd and --fastqout_notmerged_rev options. Statistics may be
output to the file specified with the --eetabbedout option. Sequences are trun-
cated as specified with the --fastq_truncqual option to remove low-quality bases
in the 3' end. Sequences shorter than specified with --fastq_minlen (after trun-
cation) are discarded (1 by default). Sequences with too many ambiguous bases
(N's), as specified with the --fastq_maxns are also discarded (no limit by de-
fault). Staggered reads are not merged unless the --fastq_allowmergestagger op-
tion is specified. The minimum length of the overlap region between the reads
may be specified with the --fastq_minovlen option (at least 5, default 10). The
overlap region may not include more mismatches than specified with the
--fastq_maxdiffs option (10 by default) or a higher percentage of

mismatches
than specified with the --fastq_maxdiffpct option (100.0% by default), otherwise
the read pair is discarded. Additional rules will avoid merging of reads that
cannot be aligned reliably and unambiguously. The minimum and maximum length of
the merged sequence may be specified with the --fastq_minmergelen and
--fastq_maxmergelen options, respectively. The quality value limits for output
files may be specified with the --fastq_qminout and --fastq_qmaxout options, but
they apply only to the merged region. Other relevant options are:
--fastq_ascii, --fastq_maxee, --fastq_nostagger, --fastq_qmax, --fastq_qmin, and
--label_suffix.

--fastq_minlen positive integer
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, discard input
sequences with less than the specified number of bases (default 1).

--fastq_minmergelen positive integer
When using --fastq_mergepairs, specify the minimum length of the merged se-
quence. The default is 1.

--fastq_minovlen positive integer
When using --fastq_mergepairs, specify the minimum overlap between the merged
reads. The default is 10. Must be at least 5.

--fastq_minqual positive integer
When using --fastq_filter or --fastx_filter, discard reads having any base with
a quality score below the given value. The default is 0, which discards none.

--fastq_nostagger
When using --fastq_mergepairs, forbid the merging of staggered read pairs. This
is the default behaviour of --fastq_mergepairs. To change that behaviour, see
the --fastq_allowmergestagger option.

--fastq_qmax positive integer
Specify the maximum quality score accepted when reading FASTQ files. The default
is 41, which is usual for recent Sanger/Illumina 1.8+ files.

--fastq_qmaxout positive integer
When using --fastq_mergepairs, --fastq_convert, --sff_convert or --fasta2fastq,
specify the maximum quality score used when writing FASTQ files. For the
--fasta2fastq command, the value specified here is the fake quality score used
for the FASTQ output file. The default is 41, which is usual for

recent
Sanger/Illumina 1.8+ files. Older formats may use a maximum quality score of 40.
The limit only applies to the merged region when using --fastq_mergepairs.

--fastq_qmin positive integer
Specify the minimum quality score accepted for FASTQ files. The default is 0,
which is usual for recent Sanger/Illumina 1.8+ files. Older formats may use
scores between -5 and 2.

--fastq_qminout positive integer
When using --fastq_mergepairs, --fastq_convert or --sff_convert, specify the
minimum quality score used when writing FASTQ files. The default is 0, which is
usual for Sanger/Illumina 1.8+ files. Older versions of the format may use
scores between -5 and 2. The limit applies only to the merged region when using
--fastq_mergepairs.

--fastq_stats filename
Analyze a FASTQ file and report the number of reads it contains. The quality en-
coding and the range of quality values may be specified with --fastq_ascii
--fastq_qmin and --fastq_qmax. That command requires the --log option and out-
puts the following detailed statistics on read length, quality score, length vs.
quality distributions, and length / quality filtering:

Read length distribution:

1. L: read length.
2. N: number of reads.
3. Pct: fraction of reads with this length.
4: AccPct: fraction of reads with this length or longer.

Quality score distribution:

1. ASCII: character encoding the quality score.
2. Q: Phred quality score.
3. Pe: probability of error associated with the quality score.
4. N: number of bases with this quality score.
5. Pct: fraction of bases with this quality score.
6: AccPct: fraction of bases with this quality score or higher.

Length vs. quality distribution:

1. L: position in reads (starting from position 2).

2. PctRecs: fraction of reads with at least this length.

3. AvgQ: average quality score at this position.

4. P(AvgQ): error probability corresponding to AvgQ.

5. AvgP: average error probability at this position.

6: AvgEE: average expected error over all reads up to this position.

7: Rate: growth rate of AvgEE between this position and position - 1.

8: RatePct: Rate (as explained above) expressed as a percentage.

Effect of expected error and length filtering:
The first column indicates read lengths (L). The next four columns indicate the number of reads that would be retained by the --fastq_filter command if the reads were truncated at length L (option --fastq_trunclen L) and filtered to have a maximum expected error of 1.0, 0.5, 0.25 or 0.1 (with the option --fastq_maxee float). The last four columns indicate the fraction of reads that would be retained by the --fastq_filter command using the same length and maximum expected error parameters.

Effect of minimum quality and length filtering:
The first column indicates read lengths (Len). The next four columns indicate the fraction of reads that would be retained by the --fastq_filter command if the reads were truncated at length Len (option --fastq_trunclen Len) or at the first position with a quality Q equal to or lesser than 5, 10, 15 or 20 (option --fastq_truncqual Q).

--fastq_stripleft positive integer
When using --fastq_filter or --fastx_filter, strip the specified number of bases from the left end of the reads. If the length of the resulting read is null, then the read is discarded.

--fastq_stripright positive integer
When using --fastq_filter or --fastx_filter, strip the specified number of bases from the right end of the reads. If the length of the resulting read is null, then the read is discarded.

--fastq_tail positive integer
When using --fastq_chars, count the number of times a series of

characters of
length k appears at the end of quality strings. By default, k = 4.

--fastq_truncee real
When using --fastq_filter or --fastx_filter, truncate sequences so that their
total expected error is not higher than the specified value.

--fastq_truncee_rate real
When using --fastq_filter or --fastx_filter, truncate sequences so that their
average expected error per base is not higher than the specified value. The
truncation will happen at the first occurrence. The average expected error per
base is calculated as the total expected number of errors divided by the length
of the sequence after truncation.

--fastq_trunclen positive integer
When using --fastq_filter or --fastx_filter, truncate sequences to the specified
length. Shorter sequences are discarded.

--fastq_trunclen_keep positive integer
When using --fastq_filter or --fastx_filter, truncate sequences to the specified
length. Shorter sequences are not discarded.

--fastq_truncqual positive integer
When using --fastq_filter, --fastq_mergepairs or --fastx_filter, truncate se-
quences starting from the first base with the specified base quality score value
or lower.

--fastqout filename
When using --fastq_filter, --fastq_mergepairs, --fastx_filter or --fasta2fastq,
write to the given FASTQ-formatted file the sequences passing the filter, or the
merged or converted sequences.

--fastqout_rev filename
When using --fastq_filter or --fastx_filter, write to the given FASTQ-formatted
file the reverse reads passing the filter.

--fastqout_discarded filename
When using --fastq_filter or --fastx_filter, write sequences that do not pass
the filter to the given FASTQ-formatted file.

--fastqout_discarded_rev filename
When using --fastq_filter or --fastx_filter, write reverse reads that do not
pass the filter to the given FASTQ-formatted file.

--fastqout_notmerged_fwd filename
When using --fastq_mergepairs, write forward reads not merged to the

specified
FASTQ file.

--fastqout_notmerged_rev filename
When using --fastq_mergepairs, write reverse reads not merged to the specified
FASTQ file.

--fastx_filter filename
Trim and/or filter the sequences in the given FASTA or FASTQ file and output the
remaining sequences to the FASTQ file specified with the --fastqout option
and/or to the FASTA file specified with the --fastaout option. Discarded se-
quences are written to the files specified with the --fastaout_discarded and
--fastqout_discarded options. The input format (FASTA or FASTQ) is automatically
detected. If the input consists of paired sequences, an input file with reverse
reads may be specified with the --reverse option, and corresponding output will
be written to the files specified with the --fastqout_rev, --fastaout_rev,
--fastqout_discarded_rev, and --fastaout_discarded_rev options. Output can not
be written to FASTQ files if the input is in FASTA format. The sequences are
first trimmed and then filtered based on the remaining bases. Sequences may be
trimmed using the options --fastq_stripleft, --fastq_stripright,
--fastq_truncee, --fastq_truncee_rate, --fastq_trunclen, --fastq_trunclen_keep
and --fastq_truncqual. The sequences may be filtered using the options
--fastq_maxee, --fastq_maxee_rate, --fastq_maxlen, --fastq_maxns, --fastq_minlen
(default 1), --fastq_minqual, --fastq_trunclen, --maxsize, and --minsize. Se-
quences not satisfying the requirements are discarded. For pairs of sequences,
both sequences in a pair must satisfy the requirements, otherwise both are dis-
carded. If no shortening or filtering options are given, all sequences are writ-
ten to the output files, possibly after conversion from FASTQ to FASTA format.
The --relabel option may be used to relabel the output sequences. The --eeout
option may be used to output the expected number of errors in each sequence. Af-
ter all sequences have been processed, the number of kept and discarded se-
quences will be shown, as well as how many of the kept sequences were trimmed.
When the input is in FASTA format, the following options are not accepted be-
cause quality scores are not available: --eeout, --fastq_ascii, --

fastq_eeout,
--fastq_maxee, --fastq_maxee_rate, --fastq_minqual, --fastq_out, --fastq_qmax,
--fastq_qmin, --fastq_truncee, --fastq_truncee_rate, --fastq_truncqual,
--fastqout_discarded, --fastqout_discarded_rev, --fastqout_rev.

--fastx_revcomp filename
Reverse-complement the sequences in the given FASTA or FASTQ file to a file
specified with the --fastaout and/or --fastqout options. If the input file is in
FASTA format, the output can not be written back to a FASTQ file due to missing
base quality scores.

--join_padgap string
When running --fastq_join, use the string as a sequence padding string. The de-
fault is NNNNNNNN (8 N's). Option accepts an empty string, or any other combina-
tion of ASCII characters.

--join_padgapq string
When running --fastq_join, use the string as a quality padding string. The de-
fault is a string of I's equal in length to the sequence padding string. The
letter I corresponds to a base quality score of 40 indicating a very high qual-
ity base with error probability of 0.0001. Option accepts an empty string, or
any other combination of ASCII characters.

--lengthout
Write sequence length information to the output files in FASTA or FASTQ format
by adding a ";length=integer" attribute in the header.

--maxsize positive integer
When using --fastq_filter or --fastx_filter, discard sequences with an abundance
higher than the specified value.

--minsize positive integer
When using --fastq_filter or --fastx_filter, discard sequences with an abundance
lower than the specified value.

--output filename
When using --fastq_eestats or --fastq_eestats2, write tabulated results to file-
name. See --fastq_eestats's and --fastq_eestats2's documentation for a complete
description of the table.

--relabel_keep
When using --relabel, keep the old identifier in the header after a space.

--relabel string

Please see the description of the same option under Chimera detection for de-
tails.

--relabel_md5
Please see the description of the same option under Chimera detection for de-
tails.

--relabel_self
Please see the description of the same option under Chimera detection for de-
tails.

--relabel_sha1
Please see the description of the same option under Chimera detection for de-
tails.

--reverse filename
When using --fastq_filter, --fastx_filter, --fastq_mergepairs or --fastq_join,
specify the FASTQ file containing containing the reverse reads.

--sff_convert filename
Convert the given SFF file to FASTQ. The FASTQ output file is specified with the
--fastqout option. The sequence may be clipped as specified in the SFF file if
the option --sff_clip is specified, otherwise no clipping occurs. Bases that
would have been clipped are converted to lower case, while the rest is in upper
case. The output quality encoding may be specified with the --fastq_asciiout op-
tion (default 33). The minimum and maximum output quality scores may be limited
using the --fastq_qminout and --fastq_qmaxout options.

--sff_clip
Specifies that the sequences converted by the --sff_convert command should be
clipped in both ends as indicated in the SFF file. By default no clipping is
performed.

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute is
added to output sequences by the --lengthout option.

--xsize
Strip abundance information from the headers when writing the output file.

--xee Strip information about expected errors (ee) from the output file headers. This
information is added by the --fastq_eeout and --eeout options.

Masking options:

An input sequence can be composed of lower- or uppercase letters. When soft masking is specified, lower case letters are treated as symbols that should be masked. Otherwise the case of the input sequences is ignored.

DUST masking is performed automatically by the commands for chimera detection (uchime_denovo, uchime_ref), clustering (cluster_fast, cluster_smallmem, cluster_size), masking (maskfasta, fastx_mask), pairwise alignment (allpairs_global) and searching (search_exact, usearch_global).

Masking is usually specified with the --qmask option, while the --dbmask option is used for the database sequences specified with the --db option with the --usearch_global, --search_exact and --uchime_ref commands.

The argument to the --qmask and --dbmask option may be none, soft or dust. If the argument is none, the no masking is performed. If the argument is soft the lower case symbols are masked. Finally, if the argument is dust, the sequence is masked using the DUST algorithm by Tatusov and Lipman to mask low-complexity regions.

If the --hardmask option is specified, all masked regions are converted to N's, otherwise masked regions are indicated by lower case letters.

If any sequence is masked, the masked version of the sequence (with lower case letters or N's) is used in all output files. Otherwise the sequence is unmodified. The exception is the sequences in the output file specified with the --uchimealns option, where the input sequences are converted to upper case first and lower case letters indicate disagreement between the aligned sequences.

The --qmask option (or --dbmask for database sequences) may be combined with the --hardmask option. The results of using the none, dust or soft argument to --qmask or --dbmask are presented below, assuming each input sequence contains both lower and uppercase symbols.

Results if the --hardmask option is off (default):

none: no masking, all symbols used, no change

dust: masked symbols lowercased, rest uppercased

soft: lowercase symbols masked, no case changes

Results if the --hardmask option is on:

none: no masking, all symbols used, no change

dust: masked symbols changed to Ns, rest unchanged

soft: lowercase symbols masked and changed to Ns

When a sequence region is masked, words in the region are not included in the indices used in the heuristic search algorithm. In all other aspects, the region is treated as other regions.

Regions in sequences that are hardmasked (with N's) have a zero alignment score and do not contribute to an alignment.

--fastaout filename
Write the masked sequences to filename, in fasta format. Applies only to the --fastx_mask command.

--fastqout filename
Write the masked sequences to filename, in fastq format. Applies only to the --fastx_mask command.

--fastx_mask filename
Mask regions in sequences contained in the specified fasta or fastq file. The default is to mask using DUST (use --qmask to modify that behaviour). The output files are specified with the --fastaout and --fastqout options. The minimum and maximum percentage of unmasked residues may be specified with the --min_unmasked_pct and --max_unmasked_pct options, respectively.

--hardmask
Symbols in masked regions are replaced by N's. The default is to replace the masked regions by lower case letters.

--maskfasta filename
Mask regions in sequences contained in the fasta file filename. The default is to mask using dust (use --qmask to modify that behaviour). The output file is specified with the --output option. This command is depreciated, please use --fastx_mask instead.

--max_unmasked_pct real
Discard sequences with more than the specified maximum percentage of unmasked residues. Works only with --fastx_mask.

--min_unmasked_pct real
Discard sequences with less than the specified minimum percentage of unmasked residues. Works only with --fastx_mask.

--output filename
Write the masked sequences to filename, in fasta format. Applies only to the --mask_fasta command.

--qmask none|dust|soft
If the argument is dust, mask regions in sequences using the DUST algorithm that detects simple repeats and low-complexity regions. This is the default for chimera detection, clustering, masking, pairwise alignment, and searching. If the argument is soft, mask the lower case letters in the input sequence. If the argument is none, do not mask.

Orienting options:

The --orient command can be used to orient the sequences in a given file in either the forward or the reverse complementary direction based on a reference database specified with the --db option. The two strands of each input sequence are compared to the reference database using nucleotide words. If one of the strands shares many more words with at least one sequence in the database than the other, that strand is chosen. The correctly oriented sequences may be written to a FASTA file specified with the --fastaout, and to a FASTQ file specified with the --fastqout option (as long as the input was also in FASTQ format). If the result is uncertain, because the number of matching words is too similar, the original sequence is written to the file specified with the --notmatched option. The results may also be written to a tab-delimited text file specified with the --tabbedout option. This file will contain the query label, the direction (+, - or ?), the number of matching words on the forward strand, and the number of matching words on the reverse complementary strand. By default, a word length of 12 is used for this command. The word length may be adjusted using the --wordlength option. There has to be at least 4 times as many matches on one strand than the other for a strand to be selected. In addition to the common options, the following options may also be specified for this command: --dbmask, --qmask, --relabel, --relabel_keep, --relabel_md5, --relabel_self, --relabel_sha1, --sizein, and --sizeout.

--db filename
Read the reference database from the given file. It may be in FASTA, FASTQ or UDB format. If an UDB file is used it should have been created with a

wordlength of 12.

--fastaout filename
Write the correctly oriented sequences to filename, in fasta format.

--fastqout filename
Write the correctly oriented sequences to filename, in fastq format.

--notmatched filename
Write the sequences with undetermined direction to filename, in the original format.

--orient filename
Orient the sequences in the given file.

--tabbedout filename
Write the resuls to a tab-delimited text file with the specified filename. This file will contain the query label, the direction (+, - or ?), the number of matching words on the forward strand, and the number of matching words on the reverse complementary strand.

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, --qsegout, --samout, --tsegout, --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. Mask-ing is performed as usual and specified with --qmask and --hardmask.

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

--allpairs_global filename
Perform optimal global pairwise alignments of the fasta sequences contained in filename. Each sequence is compared to all sequencs that come after it in the file, resulting in a total of n * (n-1) / 2 pairwise alignments, where n is the total number of sequences. This command is multi-threaded.

--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 1024). The number of threads should be lesser or equal to the number of available CPU cores. The default is to use all available resources and to launch one thread per logical core.

--uc filename
Output pairwise alignment results in filename using a tab-separated uclust-like format with 10 columns. Each sequence is compared to all other sequences, and all hits (--acceptall) or only some hits (--id float) are reported, with one pairwise comparison per line:

1. Record type, always set to 'H'.
2. Ordinal number of the target sequence (based on input order, starting from zero).
3. Sequence length.
4. Percentage of similarity with the target sequence.
5. Match orientation, always set to '+'.
6. Not used, always set to zero.
7. Not used, always set to zero.
8. Compact representation of the pairwise alignment using the CIGAR format (Compact Idiosyncratic Gapped Alignment Report): M (match/mismatch), D (deletion) and I (insertion). The equal sign '=' indicates that the query is identical to the centroid sequence (ignoring terminal gaps).
9. Label of the query sequence.
10. Label of the target sequence.

Restriction site cutting options:

The input sequences in the file specified with the --cut command are cut into fragments at all restriction sites matching the pattern given with the --cut_pattern option. The fragments on the forward strand are written to the file specified with the --fastaout file and the fragments on the reverse strand are written to the file specified with the --fastaout_rev option. Input sequences that do not match are written to the file specified with the option --fastaout_discarded, and their reverse complement are also writ-

ten to the file specified with the --fastaout_discarded_rev option. The relabel options (--relabel, --relabel_self, --relabel_keep, --relabel_md5, and --relabel_sha1) may be used to relabel the output sequences).

--cut filename
Specify the input file with sequences in FASTA format.

--cut_pattern string
Specify the restriction site cutting pattern and positions. The pattern is a string of lower- or uppercase letters specifying the nucleotides that must match, and may include ambiguous nucleotide symbols. The special characters "^" (circumflex) and "_" (underscore) are used to indicate the cutting position on the forward and reverse strand, respectively. For example, the pattern "G^AATT_C" is the pattern for the EcoRI restriction site. For such palindromic patterns (identical to its reverse complement) the command will output all possible fragments on both strands. For non-palindromic sites, it may be necessary to run the command also on the reverse complemented input sequences. Exactly one cutting site on each strand must be indicated.

--fastaout filename
Specify the output file for the resulting fragments on the forward strand.

--fastaout_rev filename
Specify the output file for the resulting fragments on the reverse strand.

--fastaout_discarded filename
Specify the output file for the non-matching sequences.

--fastaout_discarded_rev filename
Specify the output file for the non-matching sequences, reverse complemented.

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.

--biomout filename
Write search results to an OTU table in the biom version 1.0 file format. The query file contains the samples, while the database file contains the OTUs. Sample and OTU identifiers are extracted from the header of these

sequences.
See the --biomout option in the Clustering section for further details.

--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). Warning, vsearch uses global pair-wise alignments, not blast's seed-and-extend algorithm. Therefore, some common blast output values (alignment start and end, evalue, bit score) are reported differently. Output order may vary when using multiple threads. A similar out-put can be obtain with --userout filename and --userfields query+tar-get+id+alnlen+mism+opens+qlo+qhi+tlo+thi+evalue+bits. A complete list and de-scription 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 inte-ger value).
6. opens: number of columns containing a gap opening (zero or positive integer value, excluding terminal gaps).
7. qlo: first nucleotide of the query aligned with the target. Always equal to 1 if there is an alignment, 0 otherwise (see qilo to ig-nore initial gaps).
8. qhi: last nucleotide of the query aligned with the target. Always equal to the length of the pairwise alignment, 0 otherwise (see qihi to ignore terminal gaps).

9. tlo: first nucleotide of the target aligned with the query. Always equal to 1 if there is an alignment, 0 otherwise (see tilo to ignore initial gaps).

10. thi: last nucleotide of the target aligned with the query. Always equal to the length of the pairwise alignment, 0 otherwise (see tihi to ignore terminal gaps).

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 target sequences contained in filename in FASTA or FASTQ format, using global pairwise alignment. Alternatively, the name of a preformatted UDB database created using the makeudb_usearch command (see below) may be specified.

--dbmask none|dust|soft
Mask regions in the target database sequences using the dust method or the soft method, 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 '*'. To 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 sequence regions by replacing them with Ns instead of setting them to lower case as is the default. 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 also prevents pairwise alignments with very short, or with weakly matching targets, as there needs to be by default at least 12 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 (see options --wordlength and --minwordmatches to change that behaviour). 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 (default definition for --id).
3. Marine Biological Lab definition counting each gap opening (inter-nal or terminal) as a single mismatch, whether or not the gap was extended: 1.0 - [(mismatches + gap openings)/(longest sequence length)]
4. BLAST definition, equivalent to --iddef 1 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

differ-
ent 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.

--lca_cutoff real
Adjust the fraction of matching hits required for the last common ancestor
(LCA) output with the --lcaout option during searches. The default value is
1.0 which requires all hits to match at each taxonomic rank for that rank to
be included. If a lower cutoff value is used, e.g. 0.95, a small fraction of
non-matching hits are allowed while that rank will still be reported. The ar-
gument to this option must be larger than 0.5, but not larger than 1.0.

--lcaout filename
Output last common ancestor (LCA) information about the hits of each query to
a text file in a tab-separated format. The first column contains the query id,
while the second column contains the taxonomic information. The headers of the
sequences in the database must contain taxonomic information in the same for-
mat as used with the --sintax command, e.g. "tax=k:Archaea,p:Eur-
yarchaeota,c:Halobacteria". Only the initial parts of the taxonomy that are
common to a large fraction of the hits of each query will be output. It is
necessary to set the --maxaccepts option to a value different from 1 for this
information to be useful. The --top_hits_only option may also be useful. The
fraction of matching hits required may be adjusted by the --lca_cutoff option
(default 1.0).

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

--lengthout
Write sequence length information to the output files in FASTA format by
adding a ";length=integer" attribute in the header.

--match integer
Score assigned to a match (i.e. identical nucleotides) in the pairwise align-

ment. 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 matching target sequences to accept before stopping the search for a given query. 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 matching targets are accepted. If --maxaccepts and --maxrejects are both set to 0, the complete database is searched. See --maxhits option for a control on the number of hits reported per query when search is done on both strands.

--maxdiffs positive integer
Reject the sequence match if the alignment contains more than integer substitutions, insertions or deletions.

--maxgaps positive integer
Reject the sequence match if the alignment contains more than integer insertions or deletions.

--maxhits non-negative integer
Maximum number of hits to show once the search is terminated for a given query (hits are sorted by decreasing identity). When searching only on the plus strand (default situation, see --strand), the number of matching targets (--maxaccepts) and the number of hits (--maxhits) are the same. However, when searching on both strands, there could be two hits per target (one per strand): --maxhits then controls the overall number of reported hits per query. Unlimited by default or if the argument is zero. This 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 se-
quences 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 for a given query. The default value is 32. This option works in
pair with --maxaccepts. The search process sorts target sequences by decreas-
ing 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.

--minwordmatches non-negative integer
Minimum number of k-mers or word matches required for a sequence to be considered further. Default value is 12 for the default word length 8. For word lengths 3-15, the default minimum word matches are 18, 17, 16, 15, 14, 12, 11, 10, 9, 8, 7, 5 and 3, respectively. If the query sequence has fewer unique words than the number specified, all words in the query must match. If the argument is 0, no word matches are required.

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

--mothur_shared_out filename
Write search results to an OTU table in the mothur 'shared' tab-separated plain text file format. The query file contains the samples, while the database file contains the OTUs. Sample and OTU identifiers are extracted from the header of these sequences. See the --otutabout option in the Clustering section for further details.

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

--otutabout filename
Write search results to an OTU table in the classic tab-separated plain text format. The query file contains the samples, while the database file contains the OTUs. Sample and OTU identifiers are extracted from the header of these sequences (--sample option). See the --mothur_shared_out option in the Clustering section for further details.

--output_no_hits
Write both matching and non-matching queries to --alnout, --blast6out, --samout or --userout output files. Non-matching queries are labelled 'No hits' in --alnout files.

--pattern string
This option is ignored. It is provided for compatibility with usearch.

--qmask none|dust|soft
Mask regions in the 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.

--qsegout filename
Write the aligned part of each query sequence to filename in FASTA format.

--query_cov real
Reject if the fraction of the query aligned to the target sequence is lower than real (value ranging from 0.0 to 1.0 included). 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.

--samheader
Include header lines to the SAM file when --samout is specified. The header includes lines starting with @HD, @SQ and @PG, but no @RG lines (see (link) ⟨https://github.com/samtools/hts-specs⟩ <https://github.com/samtools/hts-specs>). By default no header line is written.

--samout filename
Write alignment results to filename using the SAM format (a tab-separated text file). When using the --samheader option, the SAM file starts with header lines. Each non-header line is a SAM record, which represents either a query-target alignment or the absence of match for a query (output order may vary when using multiple threads). Each record contains 11 mandatory fields and optional fields (see (link) ⟨https://github.com/samtools/hts-specs⟩ <https://github.com/samtools/hts-specs> for a complete description of the format):

1. query sequence label.
2. combination of bitwise flags. Possible values are: 0 (top hit), 4 (no hit), 16 (reverse-complemented hit), 256 (secondary hit, i.e. all hits except the top hit).
3. target sequence label.

4. first position of a target aligned with the query (always 1 for global pairwise alignments, 0 if there is no match).
5. mapping quality (ignored, always set to '*').
6. CIGAR string (set to '*' if there is no match).
7. name of the target sequence matching with the next read of the query (for mate reads only, ignored and always set to '*').
8. position of the primary alignment of the next read of the query (for mate reads only, ignored and always set to 0).
9. target sequence length (for multi-segment targets, ignored and always set to 0).
10. query sequence (complete, not only the segment aligned to the target as usearch does).
11. quality string (ignored, always set to '*').

Optional fields for query-target matches (number and order of fields may vary):

12. AS:i:? alignment score (i.e. percentage of identity).
13. XN:i:? next best alignment score (always set to 0).
14. XM:i:? number of mismatches.
15. XO:i:? number of gap openings (excluding terminal gaps).
16. XG:i:? number of gap extensions (excluding terminal gaps).
17. NM:i:? edit distance to the target (sum of XM and XG).
18. MD:Z:? string for mismatching positions.
19. YT:Z:UU string representing the alignment type.

--search_exact filename
Search for exact full-length matches to the query sequences contained in filename in the fasta database of target sequences (--db). Only 100% exact matches are reported and this command is much faster than --usearch_global. The --id, --maxaccepts and --maxrejects options are ignored, but the rest of the searching options may be specified.

--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 iden-

tical.

--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.

--top_hits_only
Only the top hits with an equally high percentage of identity between the query and database sequence sets are written to the output specified with the options --lcaout, --alnout, --samout, --userout, --blast6out, --uc, --fastapairs, --matched or --notmatched (but not --dbmatched and --dbnot-matched). For each query, the top hit is the one presenting the highest percentage of identity (see the --iddef option to change the way identity is measured). For a given query, if several top hits present exactly the same percentage of identity, the number of matching targets reported is controlled by the --maxaccepts value (1 by default), and the number of hits is controlled by the --maxhits option.

--tsegout filename
Write the aligned part of each target sequence to filename in FASTA format.

--uc filename
Output searching results in filename using a tab-separated uclust-like format with 10 columns. When using the --search_exact command, the table layout is the same than with the --allpairs_global. When using the --usearch_global command, the table present two different type of entries: hit (H) or no hit (N). Each query sequence is compared to all other sequences, and the best hit (--maxaccepts 1) or several hits (--maxaccepts > 1 and --uc_allhits)

are re-
ported (H). Output order may vary when using multiple threads. Column
content
varies with the type of entry (H or N):

1. Record type: H, or N ('hit' or 'no hit').
2. Ordinal number of the target sequence (based on input order,
start-
ing from zero). Set to '*' for N.
3. Sequence length. Set to '*' for N.
4. Percentage of similarity with the target sequence. Set to
'*' for
N.
5. Match orientation + or -. . Set to '.' for N.
6. Not used, always set to zero for H, or '*' for N.
7. Not used, always set to zero for H, or '*' for N.
8. Compact representation of the pairwise alignment using the
CIGAR
format (Compact Idiosyncratic Gapped Alignment
Report): M
(match/mismatch), D (deletion) and I (insertion). The
equal sign
'=' indicates that the query is identical to the centroid
sequence
(ignoring terminal gaps). Set to '*' for N.
9. Label of the query sequence.
10. Label of the target centroid sequence. Set to '*' for N.

--uc_allhits
With the commands --search_exact and --usearch_global, when using the --
uc op-
tion, show all hits, not just the top hit for each query.

--usearch_global filename
Compare target sequences (--db) to the query sequences contained in
filename
in FASTA or FASTQ format, 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' sec-
tion 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 (but count towards --maxrejects), 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 or 9 are generally recommended. Longer words may reduce the sensitivity/recall for weak similarities, but can increase precision. On the other hand, shorter words may increase sensitivity or recall, but may reduce precision. Computation time generally increases with shorter words and decreases with longer words, but it increases again for very long 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.

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute is added to output sequences by the --lengthout option.

Shuffling options:
Fasta entries in the input file are outputted in a pseudo-random order.

--lengthout
Write sequence length information to the output files in FASTA format by adding a ";length=integer" attribute in the header.

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

--randseed integer
When shuffling sequence order, use integer as seed. A given seed always produces the same output order (useful for replicability). Set to 0 to use a pseudo-random seed (default behaviour).

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

annotations.

--relabel_keep
When relabelling, keep the old identifier in the header after a space.

--relabel_md5
Relabel sequences using the MD5 message digest algorithm applied to each sequence. Former sequence headers are discarded. The sequence is converted to upper case and U is replaced by T before the digest is computed. The MD5 digest is a cryptographic hash function designed to minimize the probability that two different inputs gives the same output, even for very similar, but non-identical inputs. Still, there is always a very small, but non-zero probability that two different inputs give the same result. The MD5 digest generates a 128-bit (16-byte) digest that is represented by 16 hexadecimal numbers (using 32 symbols among 0123456789abcdef). Use --sizeout to conserve the abundance annotations.

--relabel_self
Relabel sequences using the sequence itself as the label.

--relabel_sha1
Relabel sequences using the SHA1 message digest algorithm applied to each sequence. It is similar to the --relabel_md5 option but uses the SHA1 algorithm instead of the MD5 algorithm. The SHA1 digest generates a 160-bit (20-byte) result that is represented by 20 hexadecimal numbers (40 symbols). The probability of a collision (two non-identical sequences having the same digest) is smaller for the SHA1 algorithm than it is for the MD5 algorithm. Use --sizeout to conserve the abundance annotations.

--sizeout
When using --relabel, --relabel_self, --relabel_md5 or --relabel_sha1, preserve and report abundance annotations to the output fasta file (using the pattern ';size=integer;').

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

--topn positive integer
Output only the first integer sequences after pseudo-random reordering.

--xlength
Strip header attribute ";length=integer" from input sequences. This attribute

```
                      is added to output sequences by the --lengthout option.

              --xsize  Strip abundance information from the headers when writing the output
file.

       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
abun-
              dance (if present) 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).

              --lengthout
                     Write sequence length information to the output files in FASTA format by
adding
                     a ";length=integer" attribute in the header.

              --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
                       Please  see the description of the same option under Chimera detection
for de-
                       tails.

              --relabel_keep
                       When relabelling, keep the old identifier in the header after a space.

              --relabel_md5
                       Please see the description of the same option under Chimera detection
for  de-
                       tails.

              --relabel_self
                       Please  see the description of the same option under Chimera detection
for de-
                       tails.

              --relabel_sha1
                       Please see the description of the same option under Chimera detection
for  de-
                       tails.
```

```
              --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  se-
                       quences.

              --sortbysize filename
                       Sort  by  decreasing  abundance  the  sequences contained in filename
(missing
                       abundance values are assumed to be ';size=1'). 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).

              --xlength
                       Strip  header attribute ";length=integer" from input sequences. This
attribute
                       is added to output sequences by the --lengthout option.

              --xsize  Strip abundance information from the headers when writing the output
file.

       Subsampling options:
              Subsampling randomly extracts a certain number or a certain percentage of the
sequences
              in the input file. If the --sizein option is in effect, the abundances of the
input se-
              quences is taken into account and the sampling is performed as if the  input
sequences
              were rereplicated, subsampled and dereplicated before being written to the output
file.
              The  extraction is performed as a random sampling with a uniform distribution
among the
              input sequences and is performed without replacement. The input file is
specified  with
              the  --fastx_subsample  option,  the output files are specified with the --
fastaout and
              --fastqout options and the amount of sequences to be  sampled  is  specified
with  the
              --sample_pct  or  --sample_size  options.  The  sequences not sampled may be
written to
              files  specified  with  the  options  --fasta_discarded  and   --
fastq_discarded.   The
              --fastq_ascii, --fastq_qmin and --fastq_qmax options are also available.

              --fastaout filename
                       Write the sampled sequences to filename, in fasta format.

              --fastaout_discarded filename
                       Write the sequences not sampled to filename, in fasta format.
```

--fastq_ascii positive integer
Define the ASCII character number used as the basis for the FASTQ quality score. The default is 33, which is used by the Sanger / Illumina 1.8+ FASTQ format (phred+33). The value 64 is used by the Solexa, Illumina 1.3+ and Illumina 1.5+ formats (phred+64). Only 33 and 64 are valid arguments.

--fastq_qmax positive integer
Specify the maximum quality score accepted when reading FASTQ files. The default is 41, which is usual for recent Sanger/Illumina 1.8+ files.

--fastq_qmin positive integer
Specify the minimum quality score accepted for FASTQ files. The default is 0, which is usual for recent Sanger/Illumina 1.8+ files. Older formats may use scores between -5 and 2.

--fastqout filename
Write the sampled sequences to filename, in fastq format. Requires input in fastq format.

--fastqout_discarded filename
Write the sequences not sampled to filename, in fastq format. Requires input in fastq format.

--fastx_subsample filename
Perform subsampling from the sequences in the specified input file that is in FASTA or FASTQ format.

--lengthout
Write sequence length information to the output files in FASTA format by adding a ";length=integer" attribute in the header.

--randseed positive integer
When subsampling, use integer as a seed for the pseudo-random generator. A given seed always produces the same output, which is useful for replicability. Set to 0 to use a pseudo-random seed (default behaviour).

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

--relabel_keep
When relabelling, keep the old identifier in the header after a space.

--relabel_md5
Relabel sequences using the MD5 message digest algorithm applied to

each se-
quence. Former sequence headers are discarded. The sequence is
converted to
upper case and U is replaced by T before the digest is computed. The
MD5 di-
gest is a cryptographic hash function designed to minimize the
probability
that two different inputs give the same output, even for very
similar, but
non-identical inputs. Still, there is always a very small, but non-zero
proba-
bility that two different inputs give the same result. The MD5 digest
gener-
ates a 128-bit (16-byte) digest that is represented by 16 hexadecimal
numbers
(using 32 symbols among 0123456789abcdef). Use --sizeout to conserve the
abun-
dance annotations.

--relabel_self
Relabel sequences using the sequence itself as the label.

--relabel_sha1
Relabel sequences using the SHA1 message digest algorithm applied to
each se-
quence. It is similar to the --relabel_md5 option but uses the SHA1
algorithm
instead of the MD5 algorithm. The SHA1 digest generates a 160-bit
(20-byte)
result that is represented by 20 hexadecimal numbers (40 symbols). The
proba-
bility of a collision (two non-identical sequences having the same
digest) is
smaller for the SHA1 algorithm than it is for the MD5 algorithm. Use --
sizeout
to conserve the abundance annotations.

--sample_pct real
Subsample the given percentage of the input sequences. Accepted values
range
from 0.0 to 100.0.

--sample_size positive integer
Extract the given number of sequences.

--sizein Take the abundance information of the input file into account,
otherwise the
abundance of each sequence is considered to be 1.

--sizeout
Write abundance information to the output file.

--xlength
Strip header attribute ";length=integer" from input sequences. This
attribute
is added to output sequences by the --lengthout option.

--xsize Strip abundance information from the headers when writing the output
file.

Taxonomic classification options:

The vsearch command --sintax will classify the input sequences according to the Sintax algorithm as described by Robert Edgar (2016) in SINTAX: a simple non-Bayesian taxonomy classifier for 16S and ITS sequences, BioRxiv, 074161. Preprint. doi: 10.1101/074161 (link) ⟨https://doi.org/10.1101/074161⟩

The name of the fasta file containing the input sequences to be classified is given as an argument to the --sintax command. The reference sequence database is specified with the --db option. The results are written in a tab delimited text file whose name is specified with the --tabbedout option. The --sintax_cutoff option may be used to set a minimum level of bootstrap support for the taxonomic ranks to be reported. The --rand-seed option may be included to specify a seed for initialisation of the random number generator used by the algorithm. Please note that when using multiple threads, the --randseed option may not work as intended, because sequences may be processed in a random order by different threads. To ensure the same results each time, use a single thread --threads 1) in combination with a fixed random seed specified with --randseed.

Multithreading is supported. Databases in UDB files are supported. The strand option may be specified.

The reference database must contain taxonomic information in the header of each se-quence in the form of a string starting with ";tax=" and followed by a comma-separated list of up to nine taxonomic identifiers. Each taxonomic identifier must start with an indication of the rank by one of the letters d (for domain) k (kingdom), p (phylum), c (class), o (order), f (family), g (genus), s (species), or t (strain). The letter is followed by a colon (:) and the name of that rank. Commas and semicolons are not al-lowed in the name of the rank. Non-ascii characters should be avoided in the names.

Example:

```
>X80725_S000004313;tax=d:Bacteria,p:Proteobacteria,c:Gammaproteobacteria,o:Enterobacte-
riales,f:Enterobacteriaceae,g:Escherichia/
Shigella,s:Escherichia_coli,t:str._K-12_sub-
str._MG1655
```

The option --notrunclabels is turned on by default for this command, allowing spaces in the taxonomic identifiers.

If two sequences in the reference database has equally many kmer matches with the query, the shortest sequence will be chosen by default. If they are equally long, the sequence appearing first in the database will be chosen. If the recommended option --sintax_random is specified, sequences with an equal number of kmer matches will instead be chosen by a random draw.

--db filename
Read the reference sequences from filename, in FASTA, FASTQ or UDB format. These sequences need to be annotated with taxonomy.

--randseed positive integer
Use integer as seed for the random number generator used in the Sintax algorithm. A given seed always produces the same output order (useful for replicability). Set to 0 to use a pseudo-random seed (default behaviour). Does not work correctly with multiple threads; please use --threads 1 to ensure correct behaviour.

--sintax filename
Read the input sequences from filename, in FASTA or FASTQ format.

--sintax_cutoff real
Specify a minimum level of bootstrap support for the taxonomic ranks that will be included in column 4 of the output file. For instance 0.9, corresponding to 90%.

--sintax_random
Break ties between sequences with equally many kmer matches by a random draw. This option is recommended and may be made the default in the future.

--tabbedout filename
Write the results to filename, in a tab-separated text format. Column 1 contains the query label. Column 2 contains the predicted taxonomy in the same format as for the reference data, with bootstrap support indicated in parentheses after each rank. Column 3 contains the strand. If the --sintax_cutoff option is used, the predicted taxonomy will be repeated in column 4 while omitting the bootstrap values and including only the ranks with support at or above the threshold.

UDB options:
Databases to be used with the --usearch_global or --sintax commands may be prepared

from FASTA files and stored to a binary UDB formatted file in order to speed up search-
ing. This may be worthwhile when searching a large database repeatedly. The sequences
are indexed and stored in a way that can be quickly loaded into memory. The commands
and options below can be used to create and inspect UDB files.

--dbmask none|dust|soft
Specify the sequence masking method used with the --makeudb_usearch command,
either none, dust or soft. No masking is performed when none is specified.
When dust is specified, the DUST algorithm will be used for masking low com-
plexity regions (short repeats and skewed composition). Lower case letters in
the input file will be masked when soft is specified (soft masking).

--hardmask
Mask sequences by replacing letters with N for the --makeudb_usearch command.
The default is to use lower case letters (soft masking).

--makeudb_usearch filename
Create an UDB database file from the FASTA-formatted sequences in the file
with the given filename. The UDB database is written to the file specified
with the --output option.

--output filename
Specify the filename of a FASTA or UDB output file for the --makeudb_usearch
or the --udb2fasta command, respectively.

--udb2fasta filename
Read the UDB database in the file with the given filename and output the se-
quences in FASTA format in the file specified by the --output option.

--udbinfo filename
Show information about the UDB database in the file with the given filename.

--udbstats filename
Report statistics about the indexed words in the UDB database in the file with
the given filename.

--wordlength positive integer
Specify the length of the words to be used when creating the UDB database in-
dex using the --makeudb_usearch command. Valid numbers range from 3 to 15. The
default is 8.

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

aln Print a string of M (match/mismatch, i.e. not a gap), 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
(Com-
pact Idiosyncratic Gapped Alignment Report): M (match/mismatch), D
(deletion)
and I (insertion). The equal sign '=' indicates that the query is
identical to
the centroid sequence (ignoring terminal gaps). 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,
excluding
terminal gaps).

id The percentage of identity, according to the identity definition
specified by
the --iddef option. Equal to id0, id1, id2, id3 or id4 below. By
default the
same as id2.

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.

id3 Marine Biological Lab definition of the percentage of identity (real
value
ranging from 0.0 to 100.0), counting each gap opening (internal or
terminal)
as a single mismatch, whether or not the gap was extended, 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 1 in a context of global pairwise alignment. The field id4 is always equal to the field id1.

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, excluding terminal gaps).

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, 0 otherwise (see qihi to ignore terminal gaps).

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 (see qilo to ignore initial gaps).

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, 0 otherwise (see tihi to ignore terminal gaps).

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 (see tilo to ignore initial gaps).

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.

vsearch by default uses the DUST algorithm for masking low-complexity regions. Masking behav-
iour is also slightly changed to be more consistent.

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

- uchime2_denovo, uchime3_denovo, alignwidth, borderline, fasta_score (chimera check-
ing)

- cluster_size, cluster_unoise, clusterout_id, clusterout_sort, profile (clustering)

- fasta_width, gzip_decompress, bzip2_decompress (general option)

- iddef (clustering, pairwise alignment, searching)

- maxuniquesize (dereplication)

- relabel_md5, relabel_self and relabel_sha1 (chimera detection, dereplication, FASTQ
processing, shuffling, sorting)

- shuffle (shuffling)

- fastq_eestats, fastq_eestats2, fastq_maxlen, fastq_truncee (FASTQ processing)

- fastaout_discarded, fastqout_discarded (subsampling)

- rereplicate (dereplication/rereplication)

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 abun-
dant than chimeras. Output non-chimeric sequences in fasta format (no wrapping):

vsearch --uchime_denovo queries.fas --abskew 1.5 --nonchimeras results.fas

```
              --fasta_width 0

       Cluster with a 97% similarity threshold, collect cluster centroids, and write cluster
descrip-
       tions 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
informa-
       tion already present, write unwrapped fasta sequences to queries_unique.fas with the new
abun-
       dance information, discard all sequences with an abundance of 1:

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

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

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

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

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

       Search a sequence dataset against itself (ignore self hits), get all matches with at
least 60%
       similarity, and collect results in a blast-like tab-separated format. Accept an
unlimited num-
       ber  of  hits  (--maxaccepts  0), and compare each query to all other sequences,
including un-
       likely candidates (--maxrejects 0):

              vsearch --usearch_global queries.fas --db queries.fas --self --id 0.6  --
blast6out  re-
              sults.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 --randseed 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
--size-
       out), keep only sequences with an abundance equal to or greater than 2:

              vsearch --sortbysize queries.fas --output queries_sorted.fas --relabel sampleA_
```

```
--size-
             out --minsize 2

AUTHORS
       Implementation and documentation by Torbjørn Rognes, Frédéric Mahé and Tomás Flouri.

CITATION
       Rognes T, Flouri T, Nichols B, Quince C, Mahé F. (2016) VSEARCH: a versatile open
source  tool
       for metagenomics.  PeerJ 4:e2584 doi: 10.7717/peerj.2584 (link) ⟨https://doi.org/
10.7717/
       peerj.2584⟩

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

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

COPYRIGHT
       Copyright (C) 2014-2025, Torbjørn Rognes, Frédéric Mahé and Tomás Flouri

       All rights reserved.

       Contact: Torbjørn Rognes <torognes@ifi.uio.no>, Department of Informatics, University of
Oslo,
       PO Box 1080 Blindern, NO-0316 Oslo, Norway

       This software is dual-licensed and available under a choice of one of two licenses,
either un-
       der the terms of the GNU General Public License version 3 or the BSD 2-Clause License.

       GNU General Public License version 3

       This program is free software: you can redistribute it and/or modify it under the terms
of the
       GNU  General  Public License as published by the Free Software Foundation, either
version 3 of
       the License, or (at your option) any later version.

       This program is distributed in the hope that it will be  useful,  but  WITHOUT  ANY
WARRANTY;
       without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.  See
       the GNU General Public License for more details.

       You should have received a copy of the GNU General Public License along with this
program.  If
       not, see (link) ⟨https://www.gnu.org/licenses/⟩ <https://www.gnu.org/licenses/>.

       The BSD 2-Clause License

       Redistribution and use in source and binary forms, with or without modification, are
permitted
```

SEE ALSO

swipe, an extremely fast pairwise local (Smith-Waterman) database search tool by Torbjørn

Rognes, available at (link) ⟨https://github.com/torognes/swipe⟩
<https://github.com/torognes/swipe>.

swarm, a fast and accurate amplicon clustering method by Frédéric Mahé and Torbjørn Rognes,
available at (link) ⟨https://github.com/torognes/swarm⟩ <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 dereplica-
tion, 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. Reduce memory requirements slightly and eliminate mem-
ory 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 give alignment coordinates ignoring terminal
gaps,

- in --uc output files, a perfect alignment is indicated with a '=' sign,

- the option --cluster_fast now sorts 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.

v1.0.15 released February 18th, 2015
Fix bug in calculation of identity metric between sequences when using the MBL definition (--iddef 3).

v1.0.16 released February 19th, 2015
Integrated patches from Debian for increased compatibility with various architectures.

v1.1.0 released February 20th, 2015
Added the --quiet option to suppress all output to stdout and stderr except for warnings and fatal errors. Added the --log option to write messages to a log file.

v1.1.1 released February 20th, 2015
Added info about --log and --quiet options to help text.

v1.1.2 released March 18th, 2015
Fix bug with large datasets. Fix format of help info.

v1.1.3 released March 18th, 2015
Fix more bugs with large datasets.

v1.2.0-1.2.19 released July 6th to September 8th, 2015
Several new commands and options added. Bugs fixed. Documentation updated.

v1.3.0 released September 9th, 2015
Changed to autotools build system.

v1.3.1 released September 14th, 2015
Several new commands and options. Bug fixes.

v1.3.2 released September 15th, 2015
Fixed memory leaks. Added '-h' shortcut for help. Removed extra 'v' in version number.

v1.3.3 released September 15th, 2015
Fixed bug in hexadecimal digits of MD5 and SHA1 digests. Added --samheader option.

v1.3.4 released September 16th, 2015
Fixed compilation problems with zlib and bzip2lib.

v1.3.5 released September 17th, 2015
Minor configuration/makefile changes to compile to native CPU and simplify makefile.

v1.4.0 released September 25th, 2015
Added --sizeorder option.

v1.4.1 released September 29th, 2015
Inserted public domain MD5 and SHA1 code to eliminate dependency on crypto and openssl
libraries and their licensing issues.

v1.4.2 released October 2nd, 2015
Dynamic loading of libraries for reading gzip and bzip2 compressed files if available.
Circumvention of missing gzoffset function in zlib 1.2.3 and earlier.

v1.4.3 released October 3rd, 2015
Fix a bug with determining amount of memory on some versions of Apple OS X.

v1.4.4 released October 3rd, 2015
Remove debug message.

v1.4.5 released October 6th, 2015
Fix memory allocation bug when reading long FASTA sequences.

v1.4.6 released October 6th, 2015
Fix subtle bug in SIMD alignment code that reduced accuracy.

v1.4.7 released October 7th, 2015
Fixes a problem with searching for or clustering sequences with repeats. In this new
version, vsearch looks at all words occurring at least once in the sequences

in the
initial step. Previously only words occurring exactly once were considered. In addi-
tion, vsearch now requires at least 10 words to be shared by the sequences, previously
only 6 were required. If the query contains less than 10 words, all words must be
present for a match. This change seems to lead to slightly reduced recall, but somewhat
increased precision, ending up with slightly improved overall accuracy.

v1.5.0 released October 7th, 2015
This version introduces the new option --minwordmatches that allows the user to specify
the minimum number of matching unique words before a sequence is considered further.
New default values for different word lengths are also set. The minimum word length is
increased to 7.

v1.6.0 released October 9th, 2015
This version adds the relabeling options (--relabel, --relabel_md5 and --relabel_sha1)
to the shuffle command. It also adds the --xsize option to the clustering, dereplica-
tion, shuffling and sorting commands.

v1.6.1 released October 14th, 2015
Fix bugs and update manual and help text regarding relabelling. Add all relabelling op-
tions to the subsampling command. Add the --xsize option to chimera detection, derepli-
cation and fastq filtering commands. Refactoring of code.

v1.7.0 released October 14th, 2015
Add --relabel_keep option.

v1.8.0 released October 19th, 2015
Added --search_exact, --fastx_mask and --fastq_convert commands. Changed most commands
to read FASTQ input files as well as FASTA files. Modified --fastx_revcomp and
--fastx_subsample to write FASTQ files.

v1.8.1 released November 2nd, 2015
Fixes for compatibility with QIIME and older OS X versions.

v1.9.0 released November 12th, 2015
Added the --fastq_mergepairs command and associated options. This command has not been
tested well yet. Included additional files to avoid dependency of autoconf for compila-
tion. Fixed an error where identifiers in fasta headers where not truncated at tabs,
just spaces. Fixed a bug in detection of the file format (FASTA/FASTQ) of a gzip com-
pressed input file.

v1.9.1 released November 13th, 2015
Fixed memory leak and a bug in score computation in --fastq_mergepairs, and

improved
speed.

v1.9.2 released November 17th, 2015
Fixed a bug in the computation of some values with --fastq_stats.

v1.9.3 released November 19th, 2015
Workaround for missing x86intrin.h with old compilers.

v1.9.4 released December 3rd, 2015
Fixed incrementation of counter when relabeling dereplicated sequences.

v1.9.5 released December 3rd, 2015
Fixed bug resulting in inferior chimera detection performance.

v1.9.6 released January 8th, 2016
Fixed bug in aligned sequences produced with --fastapairs and --userout (qrow, trow)
options.

v1.9.7 released January 12th, 2016
Masking behaviour is changed somewhat to keep the letter case of the input sequences
unchanged when no masking is performed. Masking is now performed also during chimera
detection. Documentation updated.

v1.9.8 released January 22nd, 2016
Fixed bug causing segfault when chimera detection is performed on extremely short se-
quences.

v1.9.9 released January 22nd, 2016
Adjusted default minimum number of word matches during searches for improved perfor-
mance.

v1.9.10 released January 25th, 2016
Fixed bug related to masking and lower case database sequences.

v1.10.0 released February 11th, 2016
Parallelized and improved merging of paired-end reads and adjusted some defaults. Re-
moved progress indicator when stderr is not a terminal. Added --fasta_score option to
report chimera scores in FASTA files. Added --rereplicate and --fastq_eestats commands.
Fixed typos. Added relabelling to files produced with --consout and --profile options.

v1.10.1 released February 23rd, 2016
Fixed a bug affecting the --fastq_mergepairs command causing FASTQ headers to be trun-
cated at first space (despite the bug fix release 1.9.0 of November 12th, 2015). Full
headers are now included in the output (no matter if --notrunclabels is in effect or
not).

v1.10.2 released March 18th, 2016

Fixed a bug causing a segmentation fault when running --usearch_global with an empty query sequence. Also fixed a bug causing imperfect alignments to be reported with an alignment string of '=' in uc output files. Fixed typos in man file. Fixed fasta/fastq processing code regarding presence or absence of compression library header files.

v1.11.1 released April 13th, 2016

Added strand information in UC file for --derep_fulllength and --derep_prefix. Added expected errors (ee) to header of FASTA files specified with --fastaout and --fastaout_discarded when --eeout or --fastq_eeout option is in effect for fastq_filter and fastq_mergepairs. The options --eeout and --fastq_eeout are now equivalent.

v1.11.2 released June 21st, 2016

Two bugs were fixed. The first issue was related to the --query_cov option that used a different coverage definition than the qcov userfield. The coverage is now defined as the fraction of the whole query sequence length that is aligned with matching or mismatching residues in the target. All gaps are ignored. The other issue was related to the consensus sequences produced during clustering when only N's were present in some positions. Previously these would be converted to A's in the consensus. The behaviour is changed so that N's are produced in the consensus, and it should now be more compatible with usearch.

v2.0.0 released June 24th, 2016

This major new version supports reading from pipes. Two new options are added: --gzip_decompress and --bzip2_decompress. One of these options must be specified if reading compressed input from a pipe, but are not required when reading from ordinary files. The vsearch header that was previously written to stdout is now written to stderr. This enables piping of results for further processing. The file name '-' now represent standard input (/dev/stdin) or standard output (/dev/stdout) when reading or writing files, respectively. Code for reading FASTA and FASTQ files has been refactored.

v2.0.1 released June 30th, 2016

Avoid segmentation fault when masking very long sequences.

v2.0.2 released July 5th, 2016

Avoid warnings when compiling with GCC 6.

v2.0.3 released August 2nd, 2016

Fixed bad compiler options resulting in Illegal instruction errors when running

precom-
piled binaries.

v2.0.4 released September 1st, 2016
Improved error message for bad FASTQ quality values. Improved manual.

v2.0.5 released September 9th, 2016
Add options --fastaout_discarded and --fastqout_discarded to output discarded sequences
from subsampling to separate files. Updated manual.

v2.1.0 released September 16th, 2016
New command: --fastx_filter. New options: --fastq_maxlen, --fastq_truncee. Allow --min-
wordmatches down to 3.

v2.1.1 released September 23rd, 2016
Fixed bugs in output to UC-files. Improved help text and manual.

v2.1.2 released September 28th, 2016
Fixed incorrect abundance output from fastx_filter and fastq_filter when relabelling.

v2.2.0 released October 7th, 2016
Added OTU table generation options --biomout, --mothur_shared_out and --otutabout to
the clustering and searching commands.

v2.3.0 released October 10th, 2016
Allowed zero-length sequences in FASTA and FASTQ files. Added --fastq_trunclen_keep op-
tion. Fixed bug with output of OTU tables to pipes.

v2.3.1 released November 16th, 2016
Fixed bug where --minwordmatches 0 was interpreted as the default minimum word matches
for the given word length instead of zero. When used in combination with --maxaccepts 0
and --maxrejects 0 it will allow complete bypass of kmer-based heuristics.

v2.3.2 released November 18th, 2016
Fixed bug where vsearch reported the ordinal number of the target sequence instead of
the cluster number in column 2 on H-lines in the uc output file after clustering. For
search and alignment commands both usearch and vsearch reports the target sequence num-
ber here.

v2.3.3 released December 5th, 2016
A minor speed improvement.

v2.3.4 released December 9th, 2016
Fixed bug in output of sequence profiles and updated documentation.

v2.4.0 released February 8th, 2017
Added support for Linux on Power8 systems (ppc64le) and Windows on x86_64. Improved de-
tection of pipes when reading FASTA and FASTQ files. Corrected option for specifying

output from fastq_eestats command in help text.

v2.4.1 released March 1st, 2017
Fixed an overflow bug in fastq_stats and fastq_eestats affecting analysis of very large FASTQ files. Fixed maximum memory usage reporting on Windows.

v2.4.2 released March 10th, 2017
Default value for fastq_minovlen increased to 16 in accordance with help text and for compatibility with usearch. Minor changes for improved accuracy of paired-end read merging.

v2.4.3 released April 6th, 2017
Fixed bug with progress bar for shuffling. Fixed missing N-lines in UC files with use-arch_global, search_exact and allpairs_global when the output_no_hits option was not specified.

v2.4.4 released August 28th, 2017
Fixed a few minor bugs, improved error messages and updated documentation.

v2.5.0 released October 5th, 2017
Support for UDB database files. New commands: fastq_stripright, fastq_eestats2, makeudb_usearch, udb2fasta, udbinfo, and udbstats. New general option: no_progress. New options minsize and maxsize to fastx_filter. Minor bug fixes, error message improve-ments and documentation updates.

v2.5.1 released October 25th, 2017
Fixed bug with bad default value of 1 instead of 32 for minseqlength when using the makeudb_usearch command.

v2.5.2 released October 30th, 2017
Fixed bug with where '-' as an argument to the fastq_eestats2 option was treated liter-ally instead of equivalent to stdin.

v2.6.0 released November 10th, 2017
Rewritten paired-end reads merger with improved accuracy. Decreased default value for fastq_minovlen option from 16 to 10. The default value for the fastq_maxdiffs option is increased from 5 to 10. There are now other more important restrictions that will avoid merging reads that cannot be reliably aligned.

v2.6.1 released December 8th, 2017
Improved parallelisation of paired end reads merging.

v2.6.2 released December 18th, 2017
Fixed option xsize that was partially inactive for commands uchime_denovo, uchime_ref, and fastx_filter.

v2.7.0 released February 13th, 2018
Added commands cluster_unoise, uchime2_denovo and uchime3_denovo contributed by Davide Albanese based on Robert Edgar's papers. Refactored fasta and fastq print functions as well as code for extraction of abundance and other attributes from the headers.

v2.7.1 released February 16th, 2018
Fix several bugs on Windows related to large files, use of "-" as a file name to mean stdin or stdout, alignment errors, missed kmers and corrupted UDB files. Added documentation of UDB-related commands.

v2.7.2 released April 20th, 2018
Added the sintax command for taxonomic classification. Fixed a bug with incorrect FASTA headers of consensus sequences after clustering.

v2.8.0 released April 24th, 2018
Added the fastq_maxdiffpct option to the fastq_mergepairs command.

v2.8.1 released June 22nd, 2018
Fixes for compilation warnings with GCC 8.

v2.8.2 released August 21st, 2018
Fix for wrong placement of semicolons in header lines in some cases when using the sizeout or xsize options. Reduced memory requirements for full-length dereplication in cases with many duplicate sequences. Improved wording of fastq_mergepairs report. Updated manual regarding use of sizein and sizeout with dereplication. Changed a compiler option.

v2.8.3 released August 31st, 2018
Fix for segmentation fault for --derep_fulllength with --uc.

v2.8.4 released September 3rd, 2018
Further reduce memory requirements for dereplication when not using the uc option. Fix output during subsampling when quiet or log options are in effect.

v2.8.5 released September 26th, 2018
Fixed a bug in fastq_eestats2 that caused the values for large lengths to be much too high when the input sequences had varying lengths.

v2.8.6 released October 9th, 2018
Fixed a bug introduced in version 2.8.2 that caused derep_fulllength to include the full FASTA header in its output instead of stopping at the first space (unless the notrunclabels option is in effect).

v2.9.0 released October 10th, 2018
Added the fastq_join command.

v2.9.1 released October 29th, 2018

Changed compiler options that select the target cpu and tuning to allow the software to run on any 64-bit x86 system, while tuning for more modern variants. Avoid illegal instruction error on some architectures. Update documentation of rereplicate command.

v2.10.0 released December 6th, 2018
Added the sff_convert command to convert SFF files to FASTQ. Added some additional option argument checks. Fixed segmentation fault bug after some fatal errors when a log file was specified.

v2.10.1 released December 7th, 2018
Improved sff_convert command. It will now read several variants of the SFF format. It is also able to read from a pipe. Warnings are given if there are minor problems. Errors messages have been improved. Minor speed and memory usage improvements.

v2.10.2 released December 10th, 2018
Fixed bug in sintax with reversed order of domain and kingdom.

v2.10.3 released December 19th, 2018
Ported to Linux on ARMv8 (aarch64). Fixed compilation warning with gcc version 8.1.0 and 8.2.0.

v2.10.4 released January 4th, 2019
Fixed serious bug in x86_64 SIMD alignment code introduced in version 2.10.3. Added link to BioConda in README. Fixed bug in fastq_stats with sequence length 1. Fixed use of equals symbol in UC files for identical sequences with cluster_fast.

v2.11.0 released February 13th, 2019
Added ability to trim and filter paired-end reads using the reverse option with the fastx_filter and fastq_filter commands. Added --xee option to remove ee attributes from FASTA headers. Minor invisible improvement to the progress indicator.

v2.11.1 released February 28th, 2019
Minor change to the handling of the weak_id and id options when using cluster_unoise.

v2.12.0 released March 19th, 2019
Take sequence abundance into account when computing consensus sequences or profiles after clustering. Warn when rereplicating sequences without abundance info. Guess offset 33 in more cases with fastq_chars. Stricter checking of option arguments and option combinations.

v2.13.0 released April 11th, 2019
Added the --fastx_getseq, --fastx_getseqs and --fastx_getsubseq commands to extract sequences from a FASTA or FASTQ file based on their labels. Improved handling of

ambigu-
ous nucleotide symbols. Corrected behaviour of --uchime_ref command with and options
--self and --selfid. Strict detection of illegal options for each command.

v2.13.1 released April 26th, 2019
Minor changes to the allowed options for each command. All commands now allow the log,
quiet and threads options. If more than 1 thread is specified for commands that are not
multi-threaded, a warning will be issued. Minor changes to the manual.

v2.13.2 released April 30th, 2019
Fixed bug related to improper handling of newlines on Windows. Allowed option strand
plus to uchime_ref for compatibility.

v2.13.3 released April 30th, 2019
Fixed bug in FASTQ parsing introduced in version 2.13.2.

v2.13.4 released May 10th, 2019
Added information about support for gzip- and bzip2-compressed input files to the out-
put of the version command. Adapted source code for compilation on FreeBSD and NetBSD
systems.

v2.13.5 released July 2nd, 2019
Added cut command to fragment sequences at restriction sites. Silenced output from the
fastq_stats command if quiet option was given. Updated manual.

v2.13.6 released July 2nd, 2019
Added info about cut command to output of help command.

v2.13.7 released September 2nd, 2019
Fixed bug in consensus sequence introduced in version 2.13.0.

v2.14.0 released September 11th, 2019
Added relabel_self option. Made fasta_width, sizein, sizeout and relabelling options
valid for certain commands.

v2.14.1 released September 18th, 2019
Fixed bug with sequences written to file specified with fastaout_rev for commands
fastx_filter and fastq_filter.

v2.14.2 released January 28th, 2020
Fixed some issues with the cut, fastx_revcomp, fastq_convert, fastq_mergepairs, and
makeudb_usearch commands. Updated manual.

v2.15.0 released June 19th, 2020
Update manual and documentation. Turn on notrunclabels option for sintax command by de-
fault. Change maxhits 0 to mean unlimited hits, like the default. Allow non-ascii char-
acters in headers, with a warning. Sort centroids and uc too when clusterout_sort spec-

ified. Add cluster id to centroids output when clusterout_id specified. Improve error
messages when parsing FASTQ files. Add missing fastq_qminout option and fix label_suf-
fix option for fastq_mergepairs. Add derep_id command that dereplicates based on both
label and sequence. Remove compilation warnings.

v2.15.1 released October 28th, 2020
Fix for dereplication when including reverse complement sequences and headers. Make
some extra checks when loading compression libraries and add more diagnostic output
about them to the output of the version command. Report an error when fastx_filter is
used with FASTA input and options that require FASTQ input. Update manual.

v2.15.2 released January 26th, 2021
No real functional changes, but some code and compilation changes. Compiles success-
fully on macOS running on Apple Silicon (ARMv8). Binaries available. Code updated for
C++11. Minor adaptations for Windows compatibility, including the use of the C++ stan-
dard library for regular expressions. Minor changes for compatibility with Power8.
Switch to C++ header files.

v2.16.0 released March 22nd, 2021
This version adds the orient command. It also handles empty input files properly. Docu-
mentation has been updated.

v2.17.0 released March 29nd, 2021
The fastq_mergepairs command has been changed. It now allows merging of sequences with
overlaps as short as 5 bp if the --fastq_minovlen option has been adjusted down from
the default 10. In addition, much fewer pairs of reads should now be rejected with the
reason 'multiple potential alignments' as the algorithm for detecting those have been
changed.

v2.17.1 released June 14th, 2021
Modernized code. Minor changes to help info.

v2.18.0 released August 27th, 2021
Added the fasta2fastq command. Fixed search bug on ppc64le. Fixed bug with removal of
size and ee info in uc files. Fixed compilation errors in some cases. Made some general
code improvements. Updated manual.

v2.19.0 released December 21st, 2021
Added the lcaout and lca_cutoff options to enable the output of last common ancestor
(LCA) information about hits when searching. The randseed option was added as a valid
option to the sintax command. Code improvements.

v2.20.0 released January 10th, 2022
Added the fastx_uniques command and the fastq_qout_max option for dereplication of FASTQ files. Some code cleaning.

v2.20.1 released January 11th, 2022
Fixes a bug in fastq_mergepair that caused an occational hang at the end when using multiple threads.

v2.21.0 released January 12th, 2022
This version adds the sample, qsegout and tsegout options. It enables the use of UDB databases with uchime_ref.

v2.21.1 released January 18th, 2022
Fix a problem with dereplication of empty input files. Update Altivec code on ppc64le for improved compiler compatibility (vector->__vector).

v2.21.2 released September 12th, 2022
Fix problems with the lcaout option when using maxaccepts above 1 and either lca_cutoff below 1 or with top_hits_only enabled. Update documentation. Update code to avoid compiler warnings.

v2.22.0 released September 19th, 2022
Add the derep_smallmem command for dereplication using little memory.

v2.22.1 released September 19th, 2022
Fix compiler warning.

v2.23.0 released July 7th, 2023
Update documentation. Add citation file. Modernize and improve code. Fix several minor bugs. Fix compilation with GCC 13. Print stats after fastq_mergepairs to log file instead of stderr. Handle sizein option correctly with dbmatched option for usearch_global. Allow maxseqlength option for makeudb_usearch. Fix memory allocation problem with chimera detection. Add lengthout and xlength options. Increase precision for eeout option. Add warning about sintax algorithm, random seed and multiple threads. Refactor chimera detection code. Add undocumented experimental long_chimeras_denovo command. Fix segfault with clustering. Add more references.

v2.24.0 released October 26th, 2023
Update documentation. Improve code. Allow up to 20 parents for the undocumented and experimental chimeras_denovo command. Fix compilation warnings for sha1.c. Compile for release (not debug) by default.

v2.25.0 released November 10th, 2023
Allow a given percentage of mismatches between chimeras and parents for the

experimen-
tal chimeras_denovo command.

v2.26.0 released November 24th, 2023
Enable the maxseqlength and minseqlength options for the chimera detection commands.
When the usearch_global or search_exact commands are used, OTU tables will include sam-
ples and OTUs with no matches.

v2.26.1 released November 25th, 2023
No real changes, but the previous version was released without proper updates to the
source code.

v2.27.0 released January 19th, 2024
The usearch_global and search_exact commands now support FASTQ files as well as FASTA
files as input. This version of vsearch includes clarifications and updates to the man-
ual. Some code has been refactored. Generic Dockerfiles for major Linux distributions
have been included. Some warnings from compilers and other tools have been eliminated.
The release for Windows will also include DLL's for the two compression libraries.

v2.27.1 released April 6th, 2024
This version fixes the weak_id option and makes searches report weak hits in some
cases. It also updates the names of the compression libraries to libz.so.1 and
libbz2.so.1 on Linux to make them work on common Linux distributions without installing
additional packages. README.md has been updated with information about compression li-
braries on Windows.

v2.28.0 released April 26th, 2024
The sintax command has been improved in several ways in this version of vsearch. Please
note that several details of this algorithm is not clearly described in the preprint,
and the implementation in vsearch differs from that in usearch. The former vsearch ver-
sion did not always choose the most common taxonomic entity over the 100 bootstraps
among the database sequences with the highest amount of word similarity to the query.
Instead, if several sequences had an equal similarity with the query, the sequence en-
countered in the earliest bootstrap was chosen. The confidence level was calculated
based on this sequence compared to the selected sequences from the other 99 bootstraps.
This could lead to a suboptimal choice with a low confidence. In the new version, the
most common of the sequences with the highest amount of word similarity across the 100
bootstraps will be selected, and ties will be broken randomly. Another problem

with the
old implementation was that if several sequences had the same amount of word similar-
ity, the shortest one in the reference database would be chosen, and if they were
equally long, the earliest in the database file would be chosen. A new option called
sintax_random has now been introduced. This option will randomly select one of the se-
quences with the highest number of shared words with the query, without considering
their length or position. This avoids a bias towards shorter reference sequences. This
option is strongly recommended and will probably soon be the default. Furthermore, a
ninth taxonomic rank, strain (letter t), is now recognized. The speed of the sintax
command has also been significantly improved at least in some cases. Run vsearch with
the randseed option and 1 thread to ensure reproducibility of the random choices in the
algorithm.

v2.28.1 released April 26th, 2024
Fix a segmentation fault that could occur with the blast6out and output_no_hits op-
tions.

v2.29.0 released September 26th, 2024
This version fixes seven bugs (see changelog below), adds initial support for RISC-V
architectures, and improves code quality and code testing (1,210 new tests):

- add: experimental support for RISCV64 and other 64-bit little-endian architectures,
thanks to Michael R. Crusoe and his fellow Debian developers (issue #566),

- add: official support for clang-19 and gcc 14,

- add: beta support for clang-20,

- remove: unused --output option for command --fastq_stats (issue #572),

- fix: bug in --sintax when selecting the best lineage (only low confidence values be-
low 0.5 were affected) (issue #573),

- fix: out-of-bounds error in --fastq_stats when processing empty reads (issue #571),

- fix: bug in --cut, patterns with multiple cutting sites were not detected (commit
4c4f9fa70f14b28d50185dbf322cf5727087e86a),

- fix: memory error (segmentation fault) when using --derep_id and --strand (issue
#565),

- fix: --fastq_join now obeys to --quiet and --log options (commit

87f968b09f17c17ebf8db00aebe86e89b13a3948),

- fix: --fastq_join quality padding is now also set to Q40 when quality offset is 64 (commit be0bf9b48d782286c4ce38f0bf1a4c82bd230250),

- fix: (partial) --fastq_join's handling of abundance annotations (commit f2bbcb421dc2f4dfa6603b9f31ec3e4598c1b591),

- improve: additional safeguards to validate input values and to make sure that they are within acceptable limits. Changes concern options --abskew (commit a530dd8990f8a05cb25fc0b6a5da5a14d28fbedd) and --fastq_maxdiffs (commit 4b254db7f120bfd49e86185ef3cd9070c236f940),

- improve: code quality (1.3k+ commits, 6k+ clang-tidy warnings eliminated),

- improve: documentation and help messages (issue #568),

- improve: complete refactoring and modernization of a subset of commands (--sort-bylength, --sortbysize, --shuffle, --rereplicate, --cut, --fastq_join, --fasta2fastq, --fastq_chars),

- improve: code-coverage of our test-suite for the above-mentioned commands (1,210 new tests, 4,753 in total)

v2.29.1 released October 24th, 2024

Fix a segmentation fault that could occur during alignment in version 2.29.0, for example with --uchime_ref. Some improvements to code and documentation.

v2.29.2 released December 20th, 2024

Fix a segmentation fault during clustering when the set of clusters is empty. Initial documentation in markdown format available on GitHub Pages.

v2.29.3 released February 3rd, 2025

This version is released in order to mitigate a bug that occurs when compiling the `align_simd.cc` file on x86_64 systems with the GNU C++ compiler version 9 or later with the `-O3` optimization option. It results in incorrect code that may cause bad alignments in some circumstances. We are investigating this issue further, but for now we recommend compiling with the `-O2` flag. The README.md file and the Dockerfiles have been updated to reflect this. The binaries released with this version will include this fix.

v2.29.4 released February 14th, 2025

Adjust the window size used for chimera detection down from 64 to 32. The window size

was by accident increased from 32 to 64 in version 2.23.0, leading to somewhat fewer chimeras being predicted. In addition, a compiler pragma has been included in align_simd.cc to further protect the compiler from generating wrong code.

v2.30.0 released February 27th, 2025

Add options `--n_mismatch`, `--fastq_minqual`, and `--fastq_truncee_rate`. The `--n_mismatch` option will count N's as mismatches in alignments, which may be useful to get sensible alignments for sequences with lots of N's. By default N's are counted as matches. Both the scoring and the counting of matches are affected. The new `--fastq_minqual` option for the `fastq_filter` and `fastx_filter` commands will discard sequences with any bases with a quality scores below the given value. The new `--fastq_truncee_rate` option for the same commands will truncate sequences at the first position where the number of expected errors per base is above the given value.

v2.30.1 released October 3rd, 2025

This version incorporates many code improvements, more extensive testing, better documentation and some minor bug fixes. List of changes:

- fix: use-after-free introduced in commit de6c1d8 (Jun 13, 2024),
- fix: (harmless) out-of-bounds memory issue in --derep_prefix (commit 8a0a508b),
- fix: (harmless) memory leak in --fastx_getseqs --label_field (commit a9c42713),
- fix: (harmless) memory leak when using option --userfields (--allpairs_global commit 03b95bcf; --cluster_* commit 2fde5472; --search_exact commit 45cd56d6; --usearch_global commit d83bfee9),
- fix: (harmless) valgrind error, use of uninitialized values (commit 8bab2444), also eliminates a pesky compilation warning,
- change: passing a negative value to --fastq_truncee_rate is now an error (commit a120f371),
- change: passing a negative value to --fastq_minqual is now an error (commit ff5b0c99),
- change: passing a non-ASCII symbol to --join_padgap or --join_padgapq is now an error (commit a708f5b3),
- change: when using --gapopen "*" to forbid gap opening, the penalty is now set to

INT_MAX (rather than 1,000). This might change alignment results for users who relied on the old behavior (thanks to Denis Filloux, issue #602, commit 96e9cf9e),

- change: when using command --chimeras_denovo, --tabbedout or --alnout can be the only output files specified (commit d51f0a456300a0ea69c035ba3de23c5ecf3da348)
- change: when using command --chimeras_denovo, option --lengthout is now accepted (commit 3f55cc6b)
- change: when using command --chimeras_denovo, option --xlength is now accepted (commit 0fd346cd)
- add: compilation option GLIBCXX_DEBUG when compiling for debugging (commit 5cf4a6c1, option activates more runtime checks),
- add: official support for clang 20,
- add: initial support for clang 21, initial support for GCC 15,
- add: experimental support for clang 22,
- improve: more accurate line number when reporting illegal characters in fastq headers (commit 539084e9),
- improve: more accurate line number when reporting non-ASCII characters in fastq headers (commit 98a851ed),
- improve: remove checks for unneeded libraries during compilation (commit 249bb5d5276b1be6c9add60a538a16e1b9d0ebfb),
- improve: code quality (8,536 clang-tidy warnings eliminated),
- improve: documentation and help messages (issues #604),
- improve: complete refactoring and modernization of the command --fastq_stats,
- improve: command --fastq_stats is now up to twice faster (tested on x86-64),
- improve: extensive test-suites for --fastq_stats and --sff_convert,
- improve: code coverage of our test-suite