VSEARCH and Swarm: New tools for clustering and other analyses of microbiome sequencing data

BIO9905MERG1 Course, UiO, 28 March 2017

Torbjørn Rognes
Dept. of Informatics, UiO & Dept. of Microbiology, OUS
torognes@ifi.uio.no

UiO : University of Oslo

Oslo University Hospital

Motivation: Problems with USEARCH

Frustration in the metagenomics community with the **USEARCH** tool (Edgar, 2010) often used for many purposes in metagenomics projects, including the QIIME pipeline.

Main problems:

- Inner workings only rudimentary described: we don't really know how it works
- Source code is not publicly available: we cannot improve on it or adapt it
- Only a 32-bit version freely available for academic use: it cannot be used with large datasets (64-bit version requires payment)

Aims

Our aim was to develop a new tool that is:

- Open source code software
- Able to handle very large databases (>4GB)
- Free of charge
- A drop-in replacement for USEARCH
- At least as accurate as USEARCH
- At least as fast as USEARCH

Features of VSEARCH v 2.4.2

- Chimera detection (*de novo* and reference)
- Clustering (after sorting by abundance or length, or using initial order)
- Dereplication of sequences (full length and prefix) and rereplication
- FASTQ encoding detection and conversion
- Masking of low-complexity regions with Dust
- Paired-end reads merging
- Pairwise global sequence alignment (all vs all)
- Reverse complement sequences
- Searching (global alignment and exact matches)
- Sequence quality statistics and filtering
- Shuffling and sorting (abundance and length)
- Subsampling of sequences

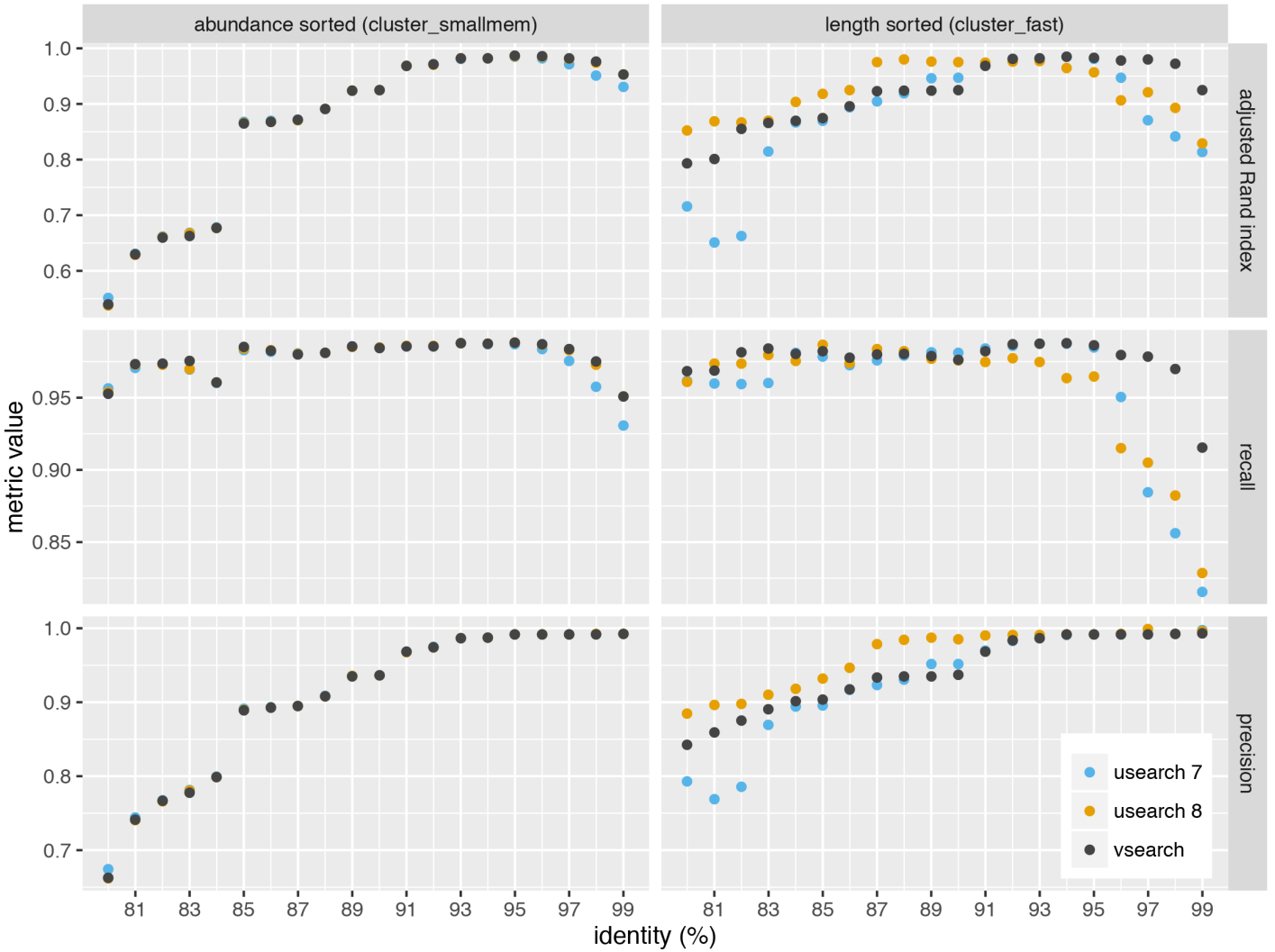
More features and options

- Almost drop-in replacement for USEARCH v 7
- Over 180 command-line options
- A wide range of different output file formats (fasta, fastq, blast-like tab-separated, user-defined tab-separated, human readable alignments, otu tables in 3 formats, uclust-like, fastapairs, chimera alignment, ...)
- Can automatically read gzip or bzip2 compressed input files

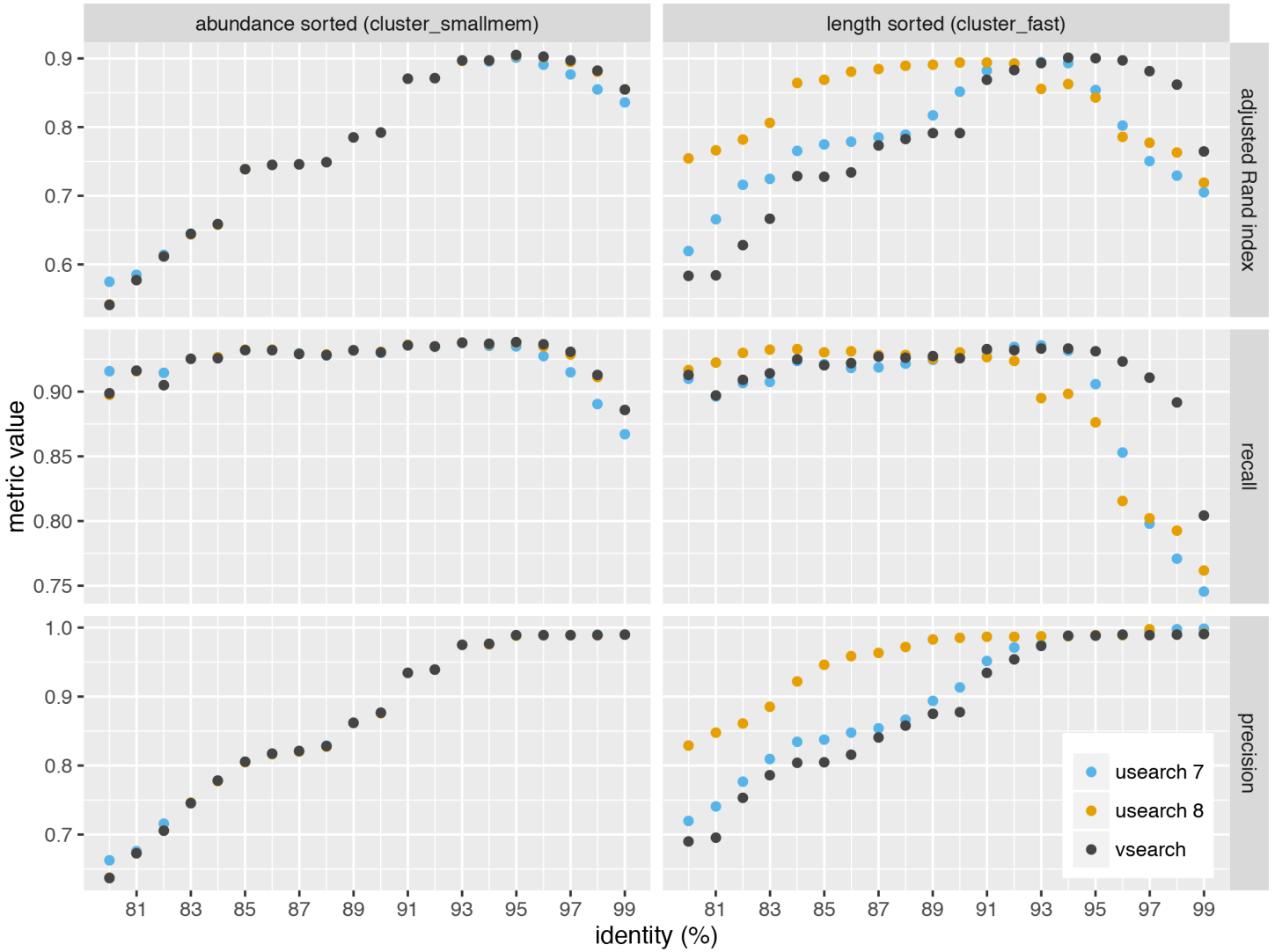
Clustering accuracy

- Tested the ability of methods to cluster together organisms that are already classified as belonging to the same clade in the tree of life
- 2 datasets: even and uneven mock communities with 57 different species, 16S rRNA V4, MiSeq 250bp PE
- 100 different sequence orders (by shuffling)
- 3 methods: USEARCH v 7, USEARCH v 8, VSEARCH
- 2 sort options: length or abundance
- 20 clustering thresholds (divergence of 1-20%)
- 3 metrics: adjusted rand index, recall and precision
- Median values plotted

Clustering accuracy - uneven dataset

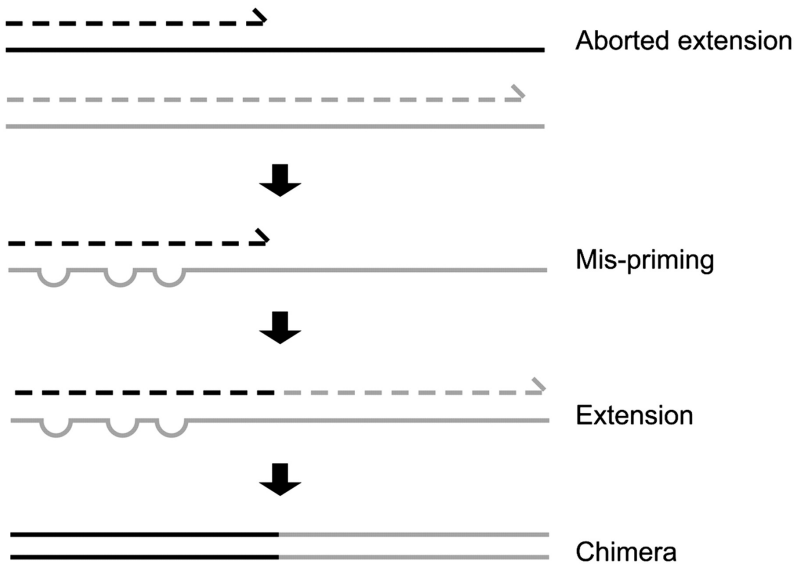


Clustering accuracy - even dataset



Chimeras

- Chimeric DNA sequences may form during PCR amplification
- Chimeras contain segments from 2 or more parent sequences
- Will inflate the apparent diversity of organisms in the sample unless removed
- VSEARCH implements the UCHIME algorithm (Edgar *et al.*, 2011)



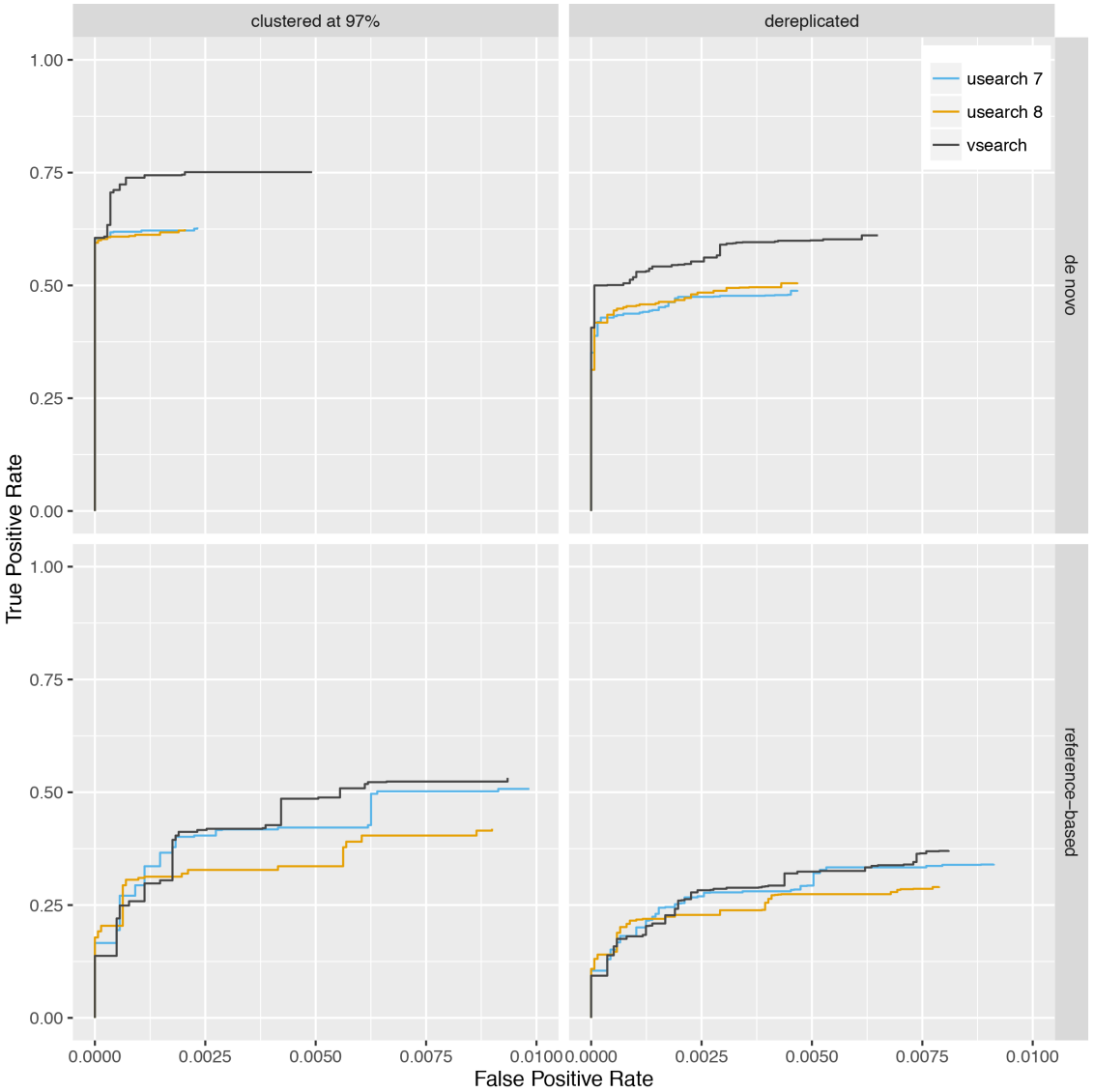
Chimera detection example

```
Query   (  250 nt) ch31_m2_90_95/sp8:0-149/sp62:149-249/N=2/top=sp8:92.4%
ParentA (  250 nt) sp8/name=Clostridiummethylpentosum_0_1_2
ParentB (  250 nt) sp62/name=Clostridiumsporogenes_0_3_2

A     1 TGCTGCCTCCCGTAGGAGTCTGGGCCGTGTctcagtcCCAATGTGGCCGTT-CAACCTCTCAGTCCGGCTA-CTGATCGt 78
Q     1 TGCTGCCTCCCGTAGGAGTCTGGGCCGTGTTCAGTCGCCAATGTGGCCGTTCCAACCTCTCAGTCCGGCTAGCTGATCG- 79
B     1 TGCTGCCTCCCGTAGGAGTCTGGaCCGTGTctcagttCCAATGTGGCCGaT-CAcCCTCTCAGgtCGGCTAcgcatcgt- 78
Diffs                          A      NNNNNN?            A    A         AA        AAAAAAA
Votes                          +      0000000            +    +         ++         +++++
Model   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

A    79 CGACTTGGTGAGCCATTACCTCACCAACTATcTAATCAGA-CGCGAGCCCATCTTaCAGCGATATAATCTTTGAT-AAcA 156
Q    80 CGACTTGGTGAGCCATTACCTCACCAACTAT-TAATCAGACCGCGAGCCCATCTT-CAGCGATATAATCTTTGATAAAAA 157
B    79 tGcCTTGGTaAGCCgTTACCTtACCAACTAg-ctAatgcgCCGCGgGtCCATCTc-aAagcAataAATCTTTGAT-AAAA 155
Diffs   A A      A    A      A         A AA AAAAA     A A      A A AAA AAA             B
Votes     +      +    +      +           + ++++      + +          +++ +++             +
Model   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAxxxxxxxxxxxxxBB

A   157 AAAcCATGCGATTCcgTTATgTTATGCGGTATTAgcgTTCgTTTCc--AAacGtTATtCCCctcTgtAAGGCAGGTTgCt 234
Q   158 AAATCATGCGATTCTCTTATATTATGCGGTATTAATCTTCCTTTCG--AA--GCTATCCCCACTTTGAAGGCAGGTTACC 233
B   156 AAATCATGCGATTCTCTTATATTATGCGGTATTAATCTTCCTTTCGgaAg--GCTATCCCCcacTTtgAGGCAGGTTACC 233
Diffs      B          BB    B             BBB   B    B   A   B   B   N?N BNa         B B
Votes      +          ++    +             +++   +            +   +   000 +0!         + +
Model   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

A   235 CACGTGTTACTCACCC- 250
Q   234 CACGTGTTACTCACCCG 250
B   234 CACGTGTTACTCACCCG 250
Diffs
Votes
Model   BBBBBBBBBBBBBBBB

Ids.  QA 88.9%, QB 82.7%, AB 80.5%, QModel 94.7%, Div. +6.5%
Diffs Left 34: N 0, A 7, Y 27 (79.4%); Right 19: N 1, A 4, Y 14 (73.7%), Score 0.8952
```

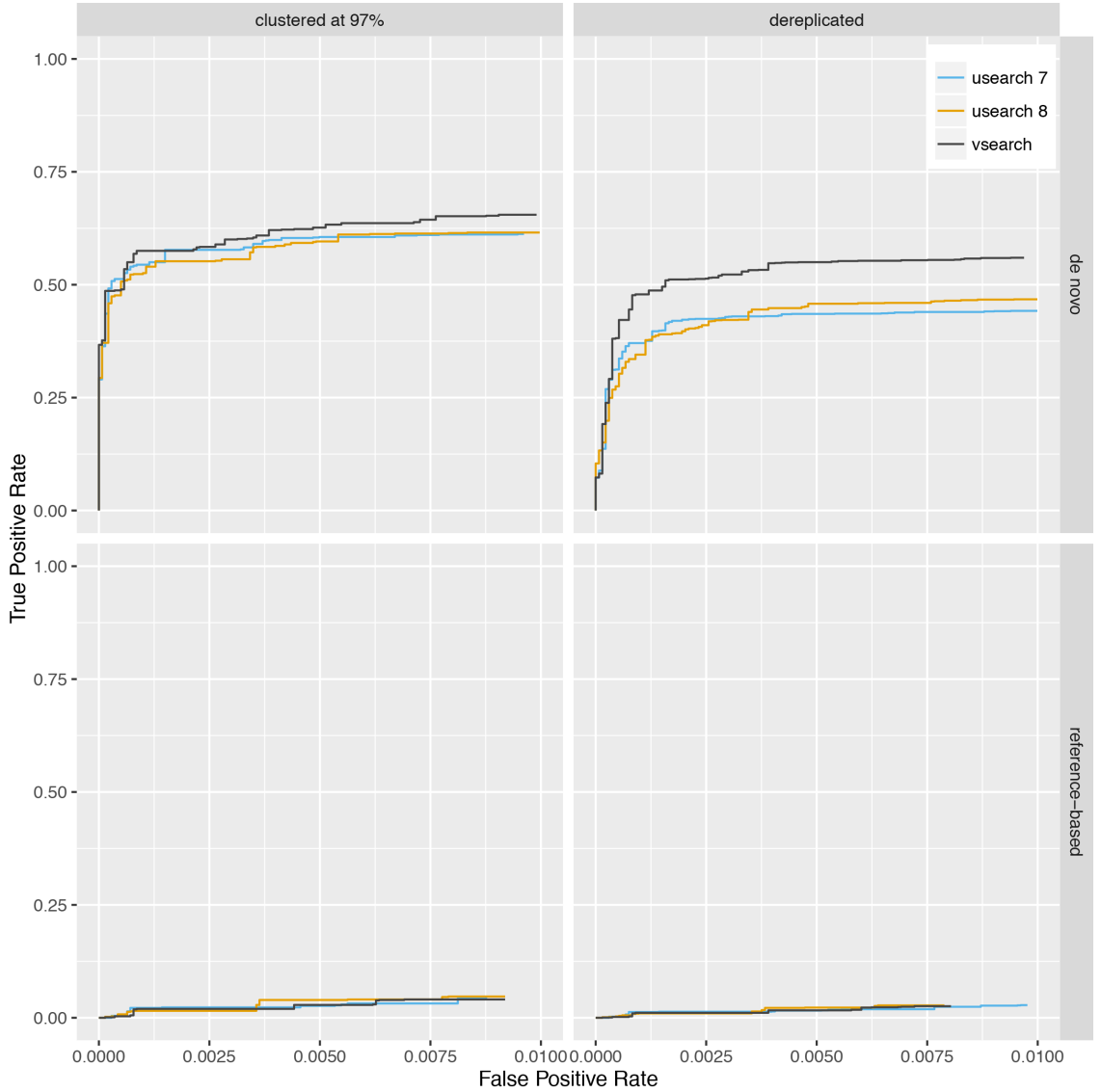
Chimera detection accuracy

- Data from Greengenes and SILVA databases (Chimera-checked 16S rRNA db)
- PCR simulation using Simera tool
- Chimeras created by combining two different sequences
- Added Illumina noise
- First performed simple dereplication or clustering at 97%
- Performed chimera detection using USEARCH v7, USEARCH v 8 and VSEARCH
- Using either the *de novo* approach (uchime_denovo) or with a reference database of clean sequences (uchime_ref) (the RDP classifier database)
- True and false positives and negatives identified, sorted by score
- ROC-like graphs plotted

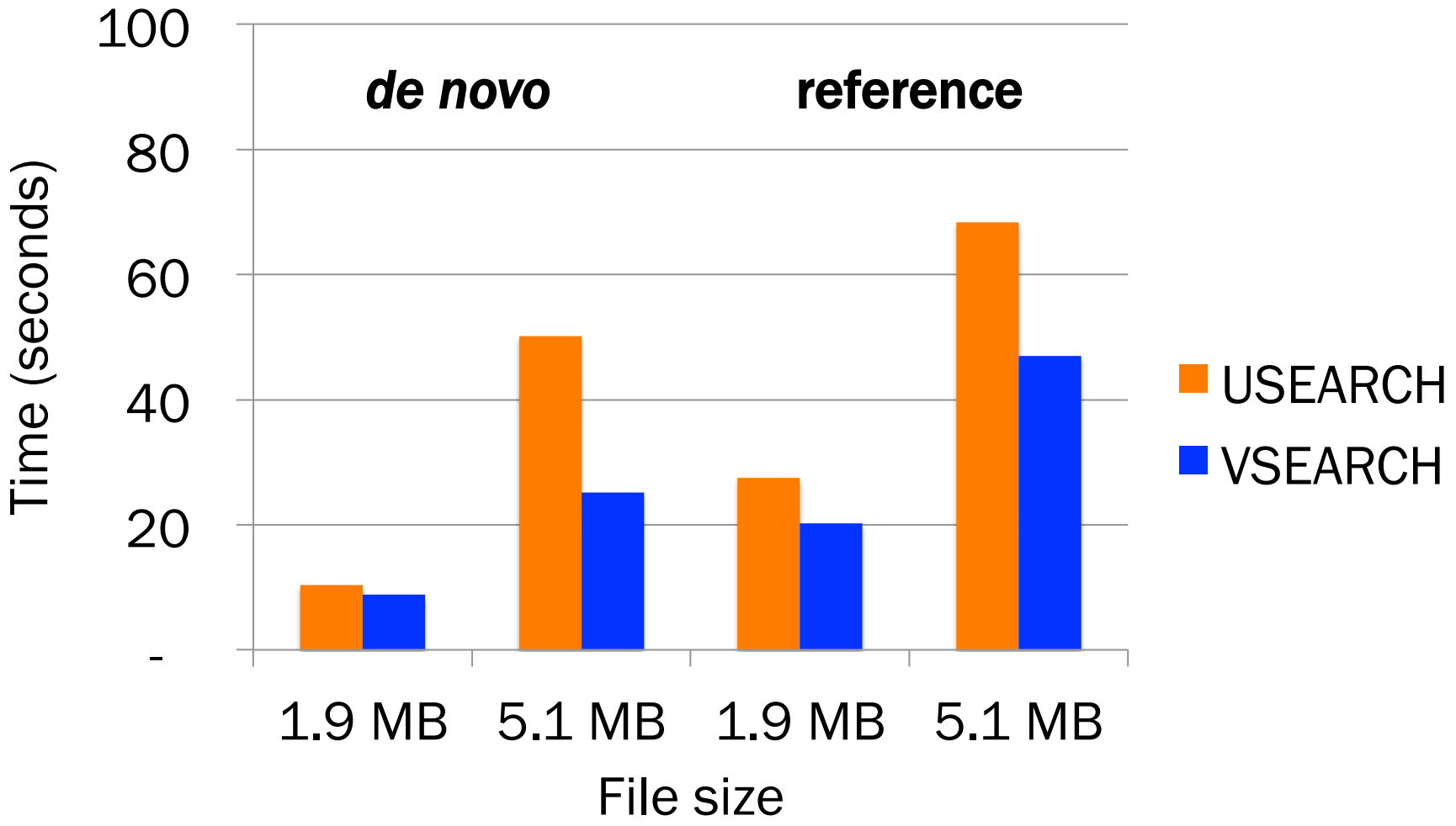
Chimera detection - Greengenes



Chimera detection - SILVA



Chimera detection speed



General search algorithm

For each query sequence:

- Sort target (database) sequences by decreasing number of *k*-mers shared with the query sequence
- Consider target sequences in order and align the query to each candidate target sequence
- Stop when *A* targets (default 1) have been accepted, i.e. they satisfy the accept criteria (e.g. id > 97%)
- Stop when *R* targets (default 32) have been rejected, i.e. they do not satisfy the accept criteria.

Heuristic search algorithm

- The sequence database is indexed
- Which database sequences contain at least one occurrence of a given k-mer?
- 4^k possible k-mers
- Quickly identify which database sequences share the most k-mers with the query
- By default k=8 (*wordlength*)
- By default at least 12 shared k-mers are required (*minwordmatches*), but never more than exist in the query
- Repeated k-mers counts only once

Example with *k*=2

Query: **AACGTCCTGCATCGATC**

k-mers:	AA AC CG GT TC CC CT TG GC CA AT TC CG GA AT TC
Sorted:	AA AC AT AT CA CC CG CG CT GA GC GT TC TC TC TG
Remove duplicates:	AA AC AT -- CA CC CG -- CT GA GC GT TC -- -- TG

Target: **AACGGCCTGCATAGATC**

k-mers:	AA AC CG GG GC CC CT TG GC CA AT TA AG GA AT TC
Sorted:	AA AC AG AT AT CA CC CG CT GA GC GC GG TA TC TG
Remove duplicates:	AA AC AG AT -- CA CC CG CT GA GC -- GG TA TC TG
Shared k-mers:	AA AC AT CA CC CG CT GA GC TC TG

11 shared *k*-mers

Alignment differences

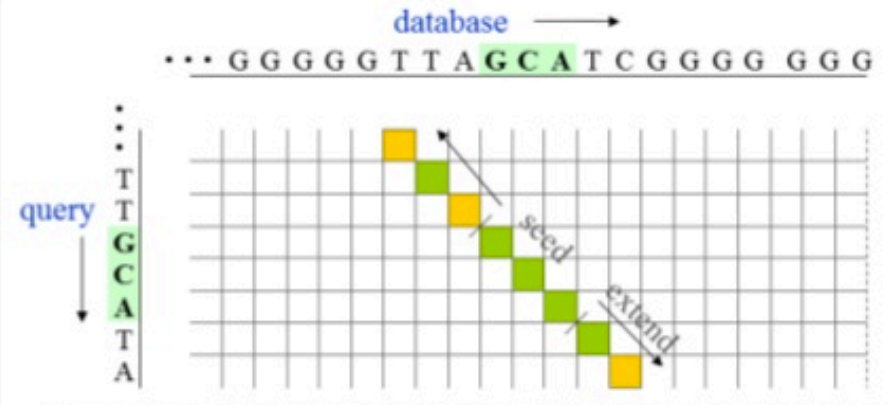
VSEARCH:

- optimal global alignment algorithm with backtracking (Needleman-Wunsch + Gotoh), 8-way SIMD parallelised

	_	A	G	A	C	T	A	G	T	T	A	C
_	0	-5	-10	-15	-20	-25	-30	-35	-40	-45	-50	-55
C	-5	-3	-8	-13	-6	-11	-16	-21	-26	-31	-36	-41
G	-10	-6	4	-1	-6	-9	-12	-9	-14	-19	-24	-29
A	-15	0	-1	14	9	4	1	-4	-9	-14	-9	-14
G	-20	-5	7	9	9	6	3	8	3	-2	-7	-12
A	-25	-10	2	17	12	7	16	11	6	1	8	3
C	-30	-15	-3	12	26	21	16	11	11	6	3	17
G	-35	-20	-8	7	21	23	20	25	20	15	10	12
T	-40	-25	-13	2	16	29	24	20	33	28	23	18

USEARCH:

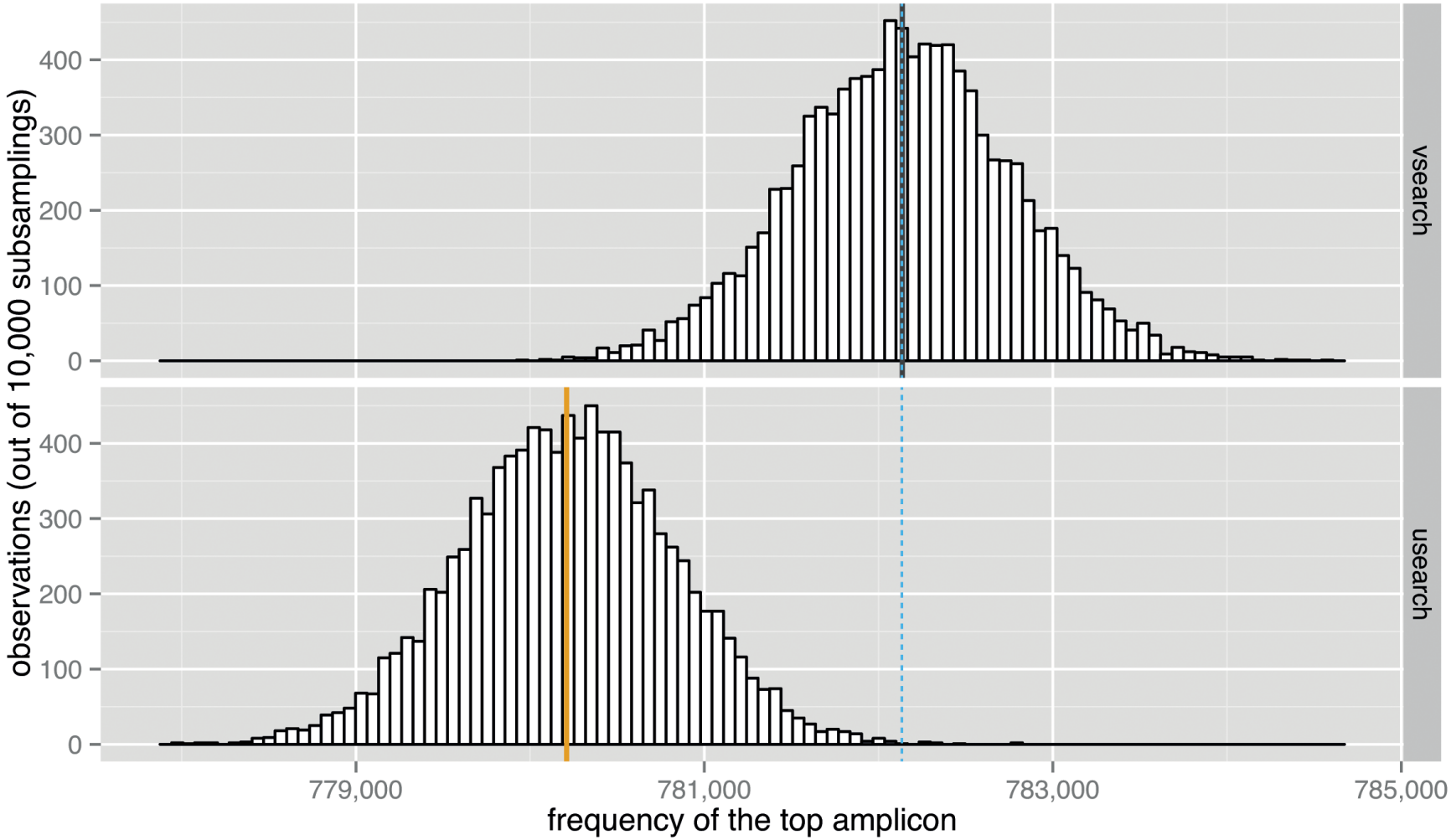
- Seed-and-extend heuristic gapped alignment, as in BLAST



Paired-end reads merging

Bacterium	Program	Pairs	Merged	Correct	%Merged	%Cor/Mer	%Cor/All	Time (s)
Staphylococcus aureus	USEARCH 7	647,052	273,438	270,849	42.26	99.05	41.86	11.65
	USEARCH 8	647,052	203,729	202,003	31.49	99.15	31.22	4.69
	VSEARCH	647,052	214,988	213,103	33.23	99.12	32.93	2.15
Methylococcus capsulatus strain Bath	USEARCH 7	673,845	643,903	642,720	95.56	99.82	95.38	14.43
	USEARCH 8	673,845	554,099	553,747	82.23	99.94	82.18	6.27
	VSEARCH	673,845	581,752	581,346	86.33	99.93	86.27	3.61

Subsampling bias



Low-complexity masking with Dust

- Dust is an old algorithm for removing low-complexity regions from DNA sequences
- Tatusov & Lipman, ca 1995 (unpublished)
- Code available
- Used in BLAST and other programs
- Examples: aaaaaaa, acacacacaca, acgacgacgac
- Dust optimized and parallelised in VSEARCH
- Dust is much faster in VSEARCH than in USEARCH
- USEARCH also includes a default alternative masking algorithm called Fastnucleo that is faster but which masks numerous short regions

Examples of low-complexity filtering

Original sequence

AGCTCCAGGAACGTGTGTTAATGTTGTTGCAGTTAAAAAGCTCGTAGTTGGATGTGTGCGACGTTGCACGAATCGGCCCG
GCAGGATGCTGTGTGCTGTTCTCGCGTCGCATTCCGCGCCGCCGCGCCGCGCGCGCGCGTTAGCTCGCGTGCGCGTACAC
CAAAGGCGTGCGGCGCCGTTTACTTTGAGAAAATGAGAGTGCTCCAAGCAGCCGAACTCTGTTCGGGCGTTGTATCATGC
AGCATGGGATAATGGAACAGGATCTCGCTCCGCGTTTCGTTGGTTTTTGGTGGCGGGATAATGATTAACAGAAACGGGCG
GGGGTACTGGTATTCACATGTCAGAGGTGAAATTCTTGGATTCTGTGAGGACCAACCTATGCGAAAGCGTCTGCCTAGGA
CGTTTTCA

VSEARCH Dust

AGCTCCAGGAACGTGTGTTAATGTTGTTGCAGTTAAAAAGCTCGTAGTTGGATGTGTGCGACGTTGCACGAATCGGCCCG
GCAGGATGCTGTGTGCTGTTNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNTACAC
CAAAGGCGTGCGGCGCCGTTTACTTTGAGAAAATGAGAGTGCTCCAAGCAGCCGAACTCTGTTCGGGCGTTGTATCATGC
AGCATGGGATAATGGAACAGGATCTCGCTCCGCGTTTCGTTGGTTTTTGGTGGCGGGATAATGATTAACAGAAACGGGCG
GGGGTACTGGTATTCACATGTCAGAGGTGAAATTCTTGGATTCTGTGAGGACCAACCTATGCGAAAGCGTCTGCCTAGGA
CGTTTTCA

USEARCH Fastnucleo

AGCTCCAGGAACGNNNNNTAATGTTGTTGCAGTTAANNNGCTCGTAGTTGGATNNNNNCGACGTTGCACGAATCGGCCCG
GCAGGATGCTNNNNNCTGTTCTCGCGTCGCATTCCGCGCCGCCGCGCCNNNNNNNNNNNTTAGCTCGCGTGCGCGTACAC
CAAAGGCGTGCGGCGCCGTTTACTTTGAGAAAATGAGAGTGCTCCAAGCAGCCGAACTCTGTTCGGGCGTTGTATCATGC
AGCATGGGATAATGGAACAGGATCTCGCTCCGCGTTTCGTTGGTTNNNGGTGGCGGGATAATGATTAACAGAAACGGGCG
GNNNTACTGGTATTCACATGTCAGAGGTGAAATTCTTGGATTCTGTGAGGACCAACCTATGCGAAAGCGTCTGCCTAGGA
CGTTTTCA

Selected command line examples

```
vsearch --help
vsearch --allpairs_global FILENAME --id 0.5 --alnout FILENAME
vsearch --cluster_size FILENAME --id 0.97 --centroids FILENAME
vsearch --derep_fulllength FILENAME --output FILENAME
vsearch --fastq_chars FILENAME
vsearch --fastq_eestats FILENAME --output FILENAME
vsearch --fastq_mergepairs FILENAME --reverse FILENAME --fastqout FILENAME
vsearch --fastx_filter FILENAME --fastaout FILENAME --fastq_maxee 0.5
vsearch --fastx_mask FILENAME --fastaout FILENAME
vsearch --fastx_subsample FILENAME --fastaout FILENAME --sample_pct 1.0
vsearch --search_exact FILENAME --db FILENAME --alnout FILENAME
vsearch --shuffle FILENAME --output FILENAME
vsearch --sortbysize FILENAME --output FILENAME
vsearch --uchime_denovo FILENAME --nonchimeras FILENAME
vsearch --uchime_ref FILENAME --db FILENAME --nonchimeras FILENAME
vsearch --usearch_global FILENAME --db FILENAME --id 0.97 --alnout FILENAME
```

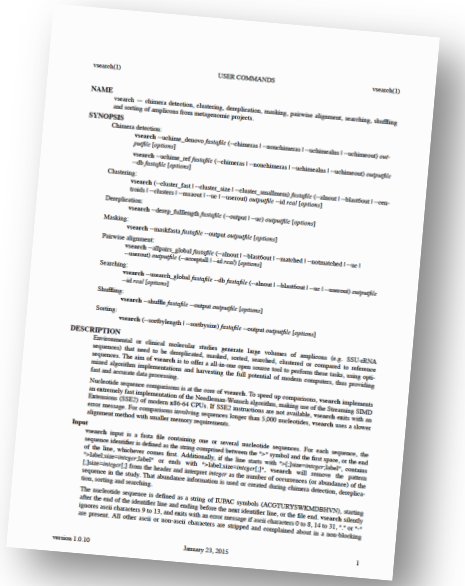
Some disadvantages of VSEARCH

Compared to USEARCH:

- Slower on long sequences due to full optimal alignment
- no support for amino acid sequences
- no UPARSE pipeline support
- no UTAX support
- few USEARCH 8 and 9 commands supported

Documentation

- Documentation / user manual in man file and 38-page PDF
- Publication:
  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](https://doi.org/10.7717/peerj.2584)
- Reproducible results. All code and data for generating all results are available:
  https://github.com/torognes/vsearch-eval


vsearch(1) USER COMMANDS vsearch(1)

NAME

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

SYNOPSIS

Chimera detection:

vsearch --uchime_denovo fastqfile (--chimeras | --nonchimeras | --uchimealns | --uchimeout) output-fastqfile [options]

vsearch --uchime_ref fastqfile (--chimeras | --nonchimeras | --uchimealns | --uchimeout) output-fastqfile --db fastqfile [options]

Clustering:

vsearch (--cluster_fast | --cluster_size | --cluster_smallmem) fastqfile (--id real | --blast6out | --centroids | --msaout | --sc | --uc | --userout) output-file --id real [options]

Dereplication:

vsearch --derep_fulllength fastqfile (--output | --uc) output-file [options]

Masking:

vsearch --maskfastq fastqfile --output output-file [options]

Pairwise alignment:

vsearch --allpairs_global fastqfile (--blast6out | --blast6out | --matched | --notmatched | --uc | --useroout) output-file (--acceptall | --id real) [options]

Searching:

vsearch --usearch_global fastqfile --db fastqfile (--alnout | --blast6out | --uc | --useroout) output-file --id real [options]

Shuffling:

vsearch --shuffle fastqfile --output output-file [options]

Sorting:

vsearch (--sortbylength | --sortbysize) fastqfile --output output-file [options]

DESCRIPTION

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

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

Input

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

The nucleotide sequence is defined as a string of IUBAC 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 reapplied about in a non-blocking

version 1.0.10 January 23, 2015 1


PeerJ

VSEARCH: a versatile open source tool for metagenomics

Torbjørn Rognes[1,2], Tomáš Flouri[3,4], Ben Nichols[5], Christopher Quince[1,6] and Frédéric Mahé[7,8]

[1] Department of Informatics, University of Oslo, Oslo, Norway
[2] Department of Microbiology, Oslo University Hospital, Oslo, Norway
[3] Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
[4] Institute for Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[5] School of Engineering, University of Glasgow, Glasgow, United Kingdom
[6] Warwick Medical School, University of Warwick, Coventry, United Kingdom
[7] Department of Ecology, University of Kaiserslautern, Kaiserslautern, Germany
[8] UMR LSTM, CIRAD, Montpellier, France

ABSTRACT

Background. VSEARCH is an open source and free of charge multithreaded 64-bit tool for processing and preparing metagenomics, genomics and population genomics nucleotide sequence data. It is designed as an alternative to the widely used USEARCH tool (Edgar, 2010) for which the source code is not publicly available, algorithm details are only rudimentarily described, and only a memory-confined 32-bit version is freely available for academic use.

Methods. When searching nucleotide sequences, VSEARCH uses a fast heuristic based on words shared by the query and target sequences in order to quickly identify similar sequences, a similar strategy is probably used in USEARCH. VSEARCH then performs optimal global sequence alignment of the query against potential target sequences, using full dynamic programming instead of the seed-and-extend heuristic used by USEARCH. Pairwise alignments are computed in parallel using vectorisation and multiple threads.

Results. VSEARCH includes most commands for analysing nucleotide sequences available in USEARCH version 7 and several of those available in USEARCH version 8, including searching (exact or based on global alignment), clustering by similarity (using length pre-sorting, abundance pre-sorting or a user-defined order), chimera detection (reference-based or *de novo*), dereplication (full length or prefix), pairwise alignment, reverse complementation, sorting, and subsampling. VSEARCH also includes commands for FASTQ file processing, i.e., format detection, filtering, read quality statistics, and merging of paired reads. Furthermore, VSEARCH extends functionality with several new commands and improvements, including shuffling, reverse complementation of low-complexity sequences with the well-known DUST algorithm, a choice among different similarity definitions, and FASTQ file format conversion. VSEARCH is here shown to be more accurate than USEARCH when performing searching, clustering, chimera detection and subsampling, while on a par with USEARCH for paired-ends read merging. VSEARCH is slower than USEARCH when performing clustering and chimera detection, but significantly faster when performing paired-end reads merging and dereplication. VSEARCH is available at https://github.com/torognes/vsearch under either the BSD 2-clause license or the GNU General Public License version 3.0.

Submitted 5 September 2016
Accepted 17 September 2016
Published 18 October 2016

Corresponding author
Torbjørn Rognes, torognes@ifi.uio.no

Academic editor
Tomas Hrbek

Additional Information and Declarations can be found on page 18

DOI 10.7717/peerj.2584

Copyright
2016 Rognes et al.

Distributed under
Creative Commons CC-BY 4.0

OPEN ACCESS

How to cite this article Rognes et al. (2016), VSEARCH: a versatile open source tool for metagenomics. PeerJ 4:e2584; DOI 10.7717/peerj.2584

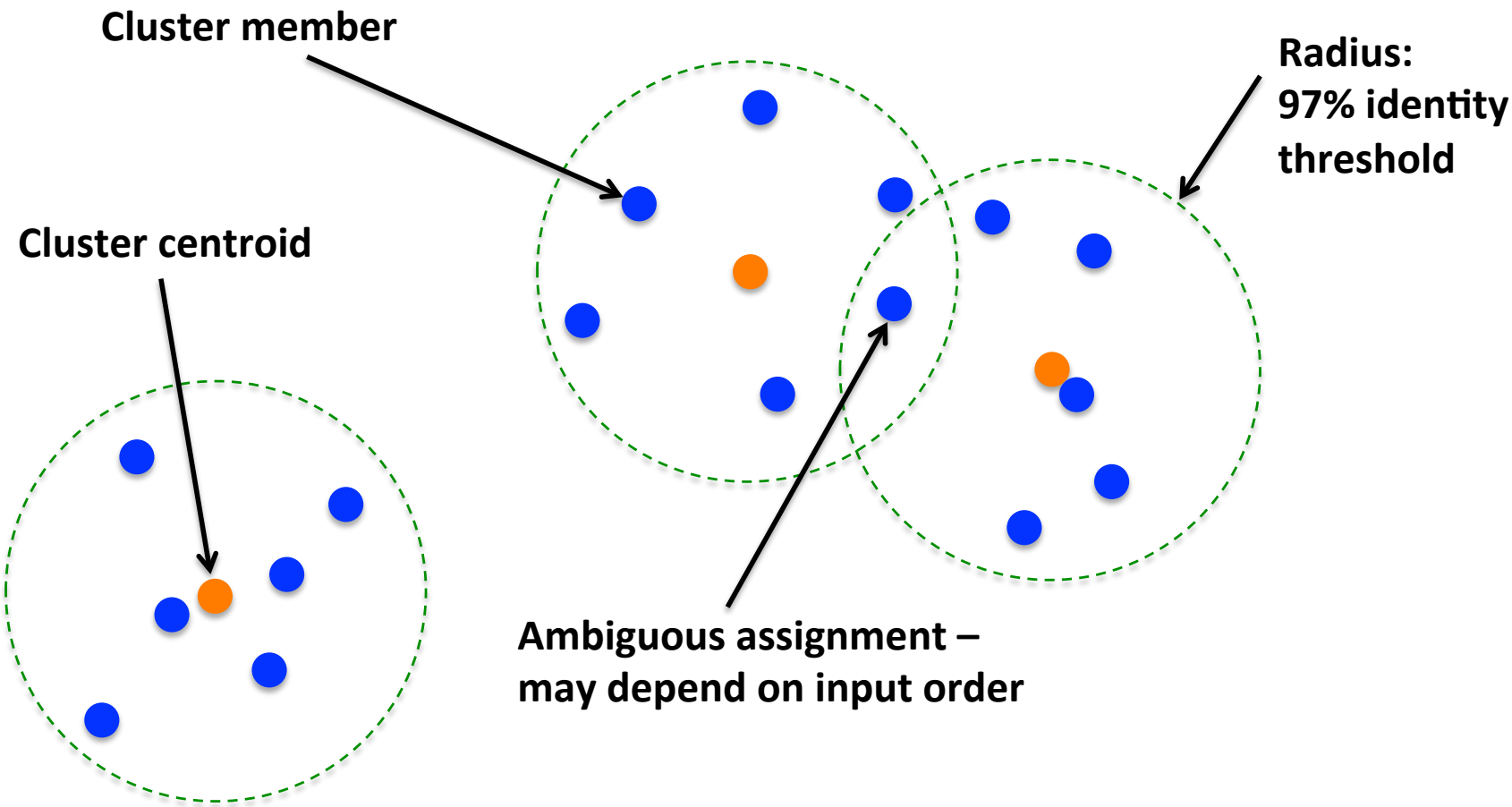
Availability

- VSEARCH source code is available on GitHub: https://github.com/torognes/vsearch
- Dual open-source license:
  - GNU Affero General Public License version 3
  - BSD 2-clause license
- Binary distribution (64 bit) available for
  - Linux x86 and ppc64le
  - macOS 10.7 and later
  - Windows 7 and later
- May be used in mothur for clustering and chimera detection
- To be integrated in QIIME, but it reportedly partially works already as a drop-in replacement for USEARCH.

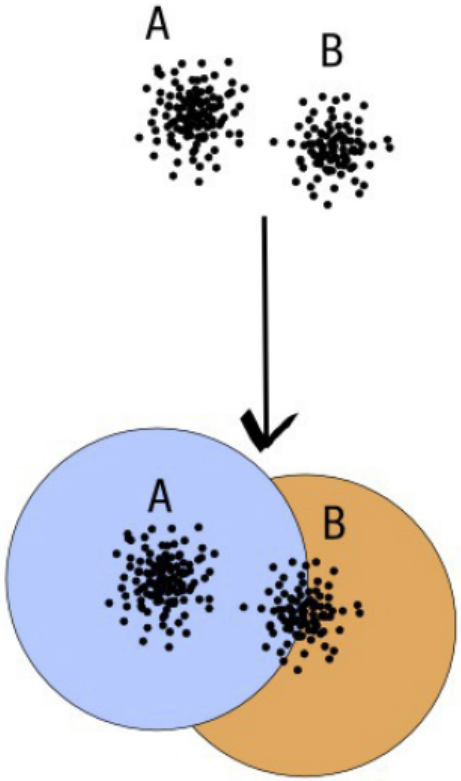
Swarm: a new clustering method

- Method for clustering metagenomic amplicon sequences into Operational Taxonomic Units (OTUs)
- OTUs are groups of sequences that could approximately represent a species or a genus
- Single linkage hierarchical clustering approach
- Avoids two problems with traditional methods:
  - input order dependence
  - global clustering threshold
- High resolution
- Linear time and space complexity

Heuristic centroid-based clustering

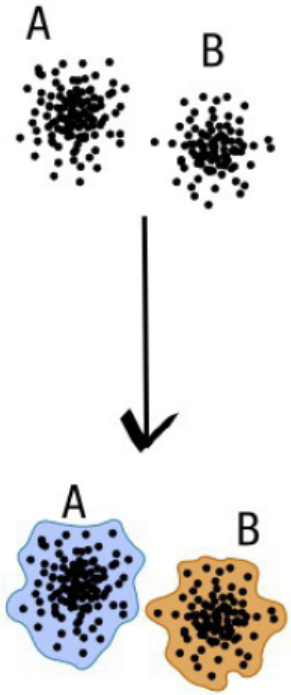


Clustering thresholds



compromise threshold
unadapted threshold

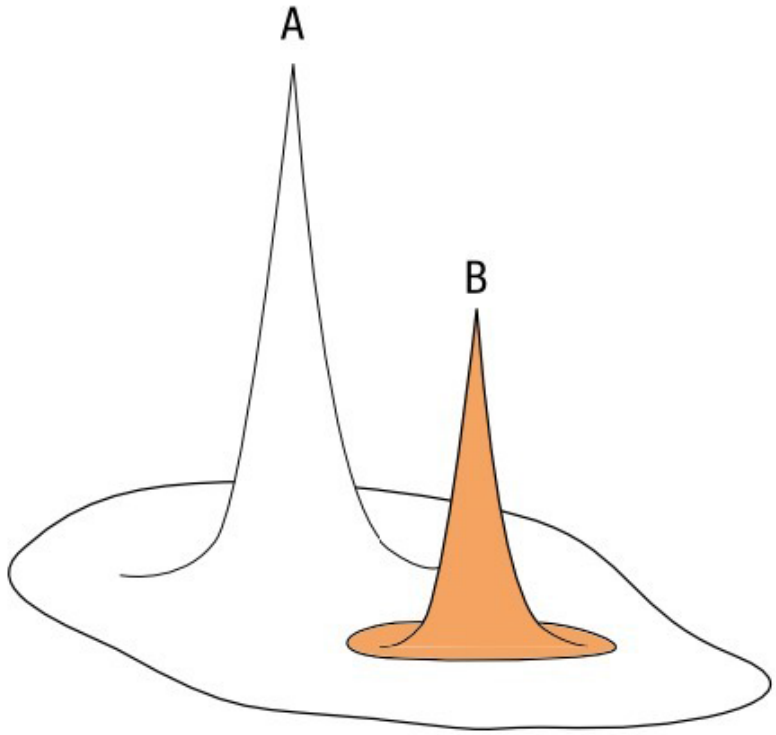
defined by maximum
distance from centroid



natural limits of OTUs

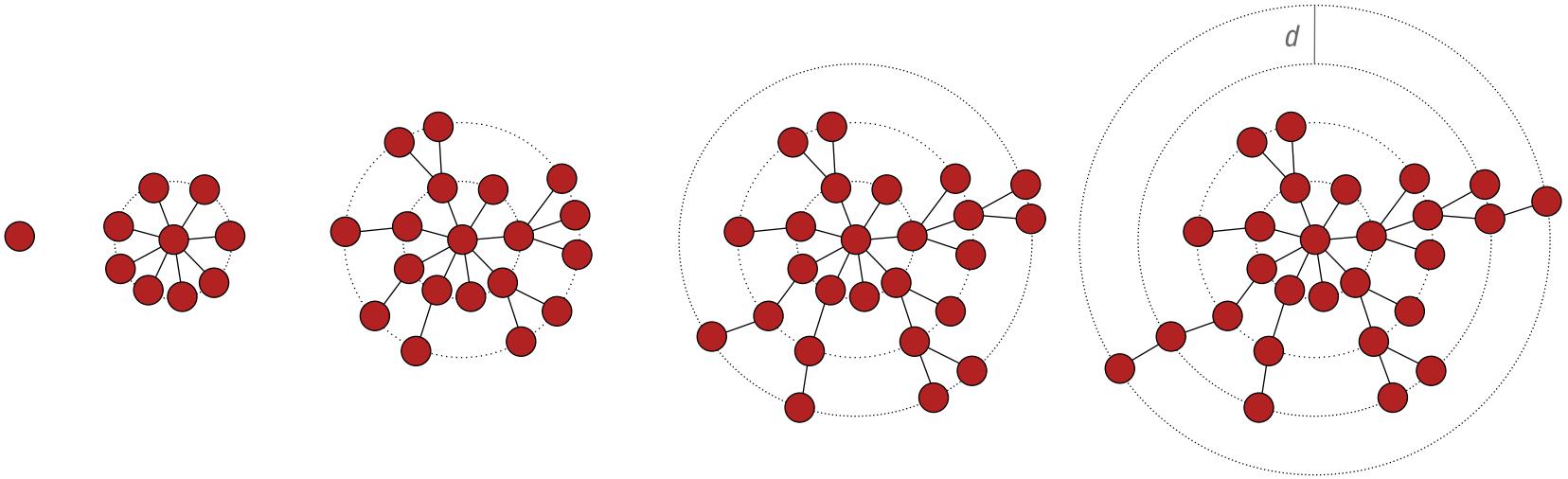
defined by minimum
distance separating clusters

OTUs - mountains in sequence space



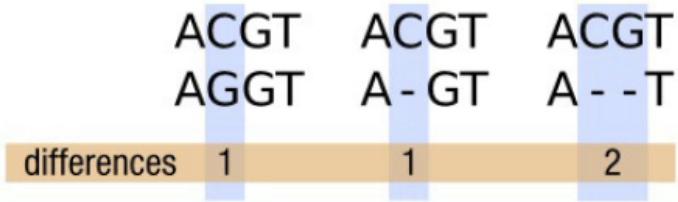
The height of the peaks is given by the abundance of each amplicon, while the position is given by the sequence.

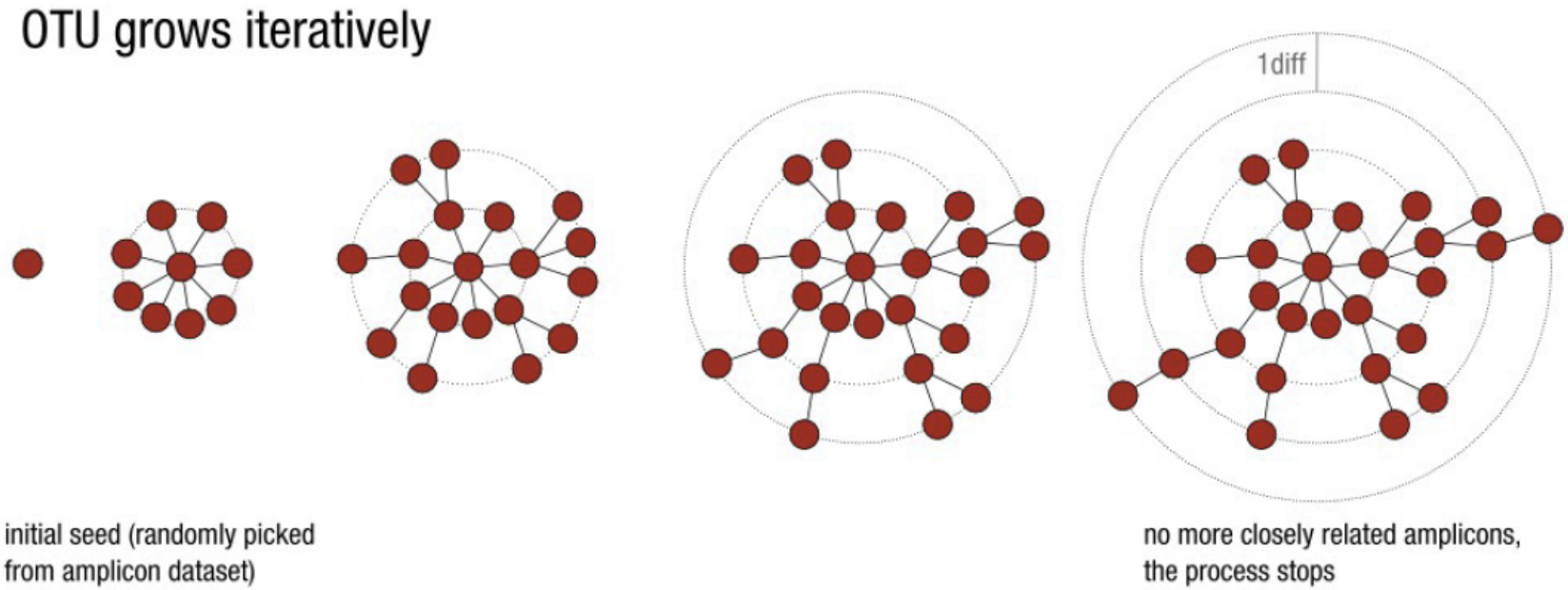
Linking amplicons



- Amplicons (nodes) are linked when distance is less than or equal to *d*.
- Similar to single linkage hierarchical clustering
- Forms a spanning tree
- The adjustable parameter *d* is 1 by default.

Linking amplicons





Algorithm for linking when d>1

- Find all sequences within distance *d* from each other.
- Need to compute distance between all pairs of sequences using global alignment
- Time complexity: $O(N^2)$, where N is the number of sequences, assuming constant sequence length.

Avoiding alignments

We can eliminate most alignments using:

- *k*-mer based filtering of sequences that must be at a distance of more than *d* because they have more than 2*kd* differences in their *k*-mer vectors.
- triangle inequality to eliminate computation of dist(a,c) if we already know dist(a,b) and dist(b,c).

Algorithm for linking when d=1

- Generates all “microvariants” of each sequence
- Microvariants are at distance 1 from the sequence and contain a single mutation, either a substitution, a deletion or an insertion of 1 bp resulting in a different sequence.
- There exists between 6L + 5 and 7L + 4 microvariants of a nucleotide sequence of length L, depending on the contents of the sequence

Microvariants example

Original sequence (L=2): AT

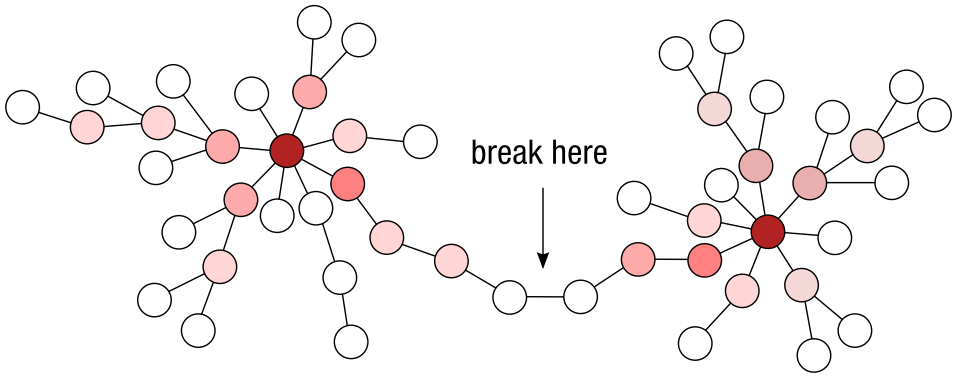
Microvariants (18 different):

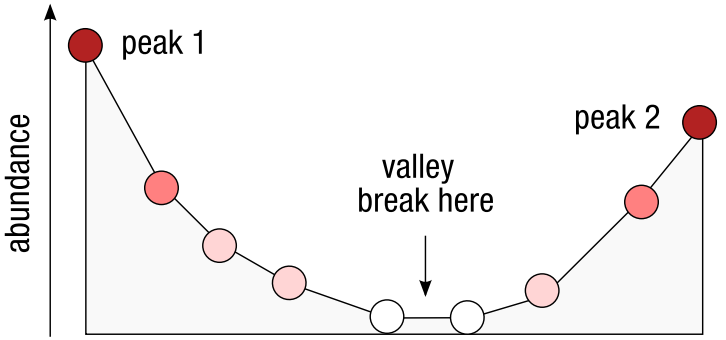
CT, GT, TT	(substitution at position 1)
AA, AC, AG	(substitution at position 2)
AAT, CAT, GAT, TAT	(insertion before position 1)
~~AAT~~, ACT, AGT, ATT	(insertion between 1 and 2)
ATA, ATC, ATG, ~~ATT~~	(insertion after position 2)
T	(deletion at position 1)
A	(deletion at position 2)

Hashing microvariants

- A hash is computed for each sequence in the set and they are all stored in a hash table.
- All microvariants of each sequence is generated, their hash is computed and a lookup in the hash table is performed to see of the microvariant is equal to an existing sequence.
- Time complexity: O(*N*), where *N* is the number of sequences, assuming constant sequence length.

Breaking long chains of amplicons

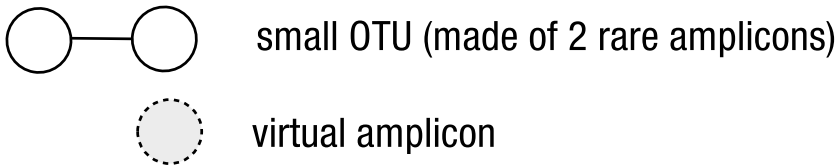


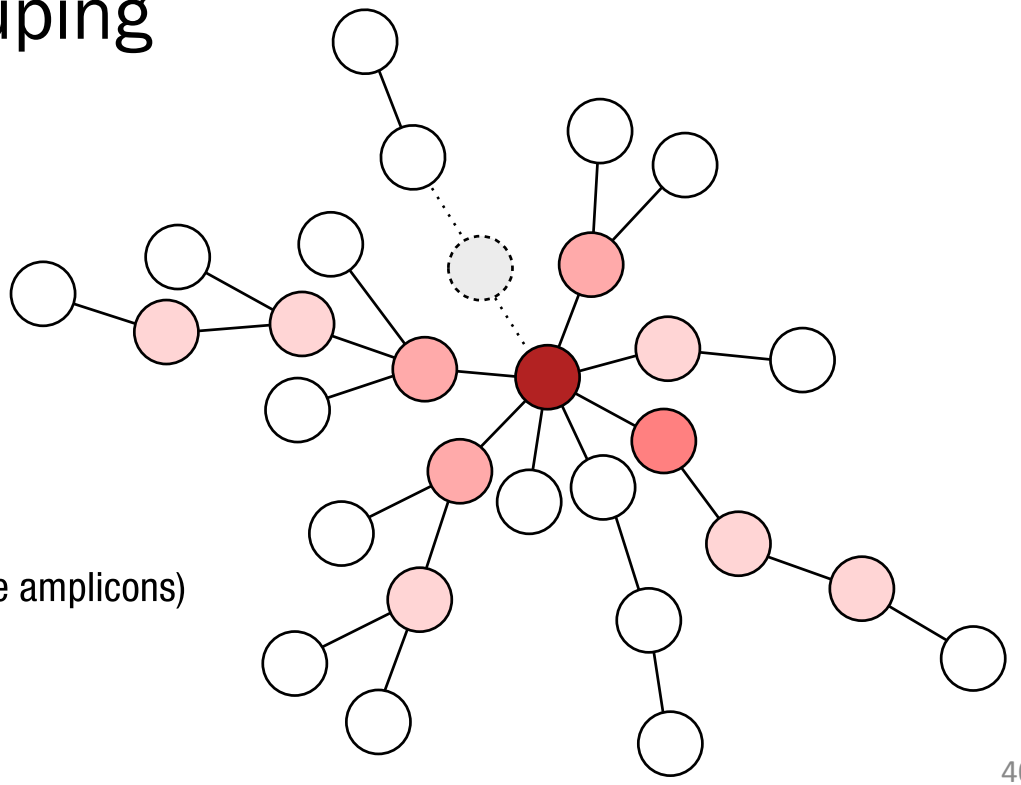


- Chains of linked amplicons can be broken to avoid problems with over-grouping.
- Breakpoint in valleys between abundance peaks.
- Always start at highest abundance and only link amplicons with lower or equal abundance.

“Fastidious” mode

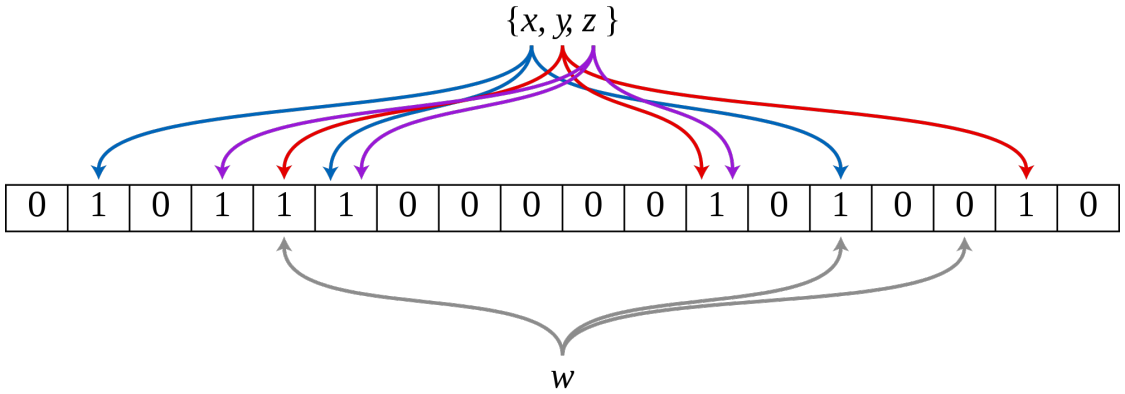
- Heuristic approximation to clustering with d=2
- Still linear complexity (but 3x slower than d=1)
- Connects small OTUs to large OTUs if d=2.
- Reduces under-grouping



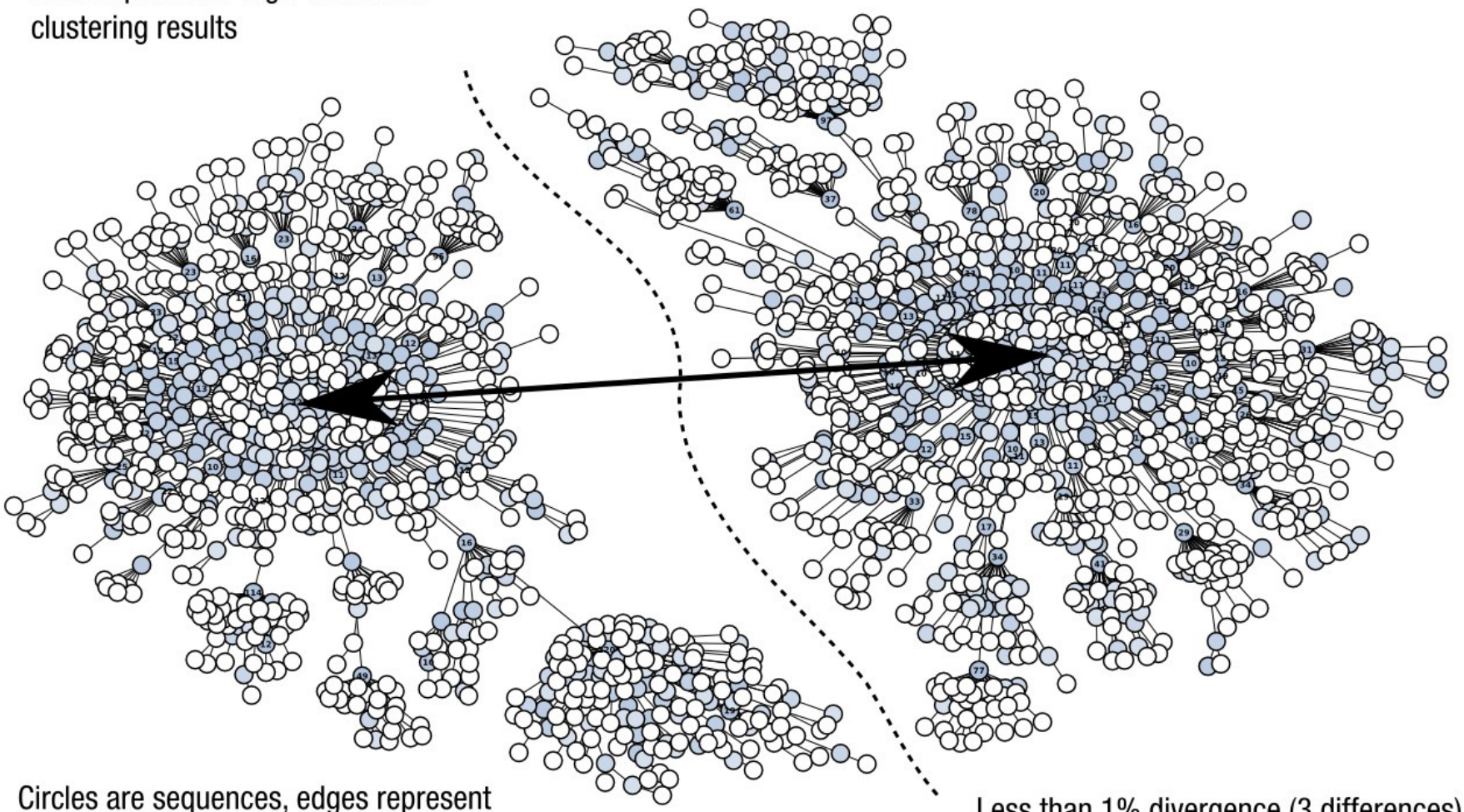


“Fastidious” algorithm

- Stores all non-existing microvariants (“virtual amplicons”) outside real amplicons of large OTUs in a Bloom filter, a very space-efficient probablistic data structure.
- Detects “virtual amplicons” by checking the rim of the small OTUs against the Bloom filter.

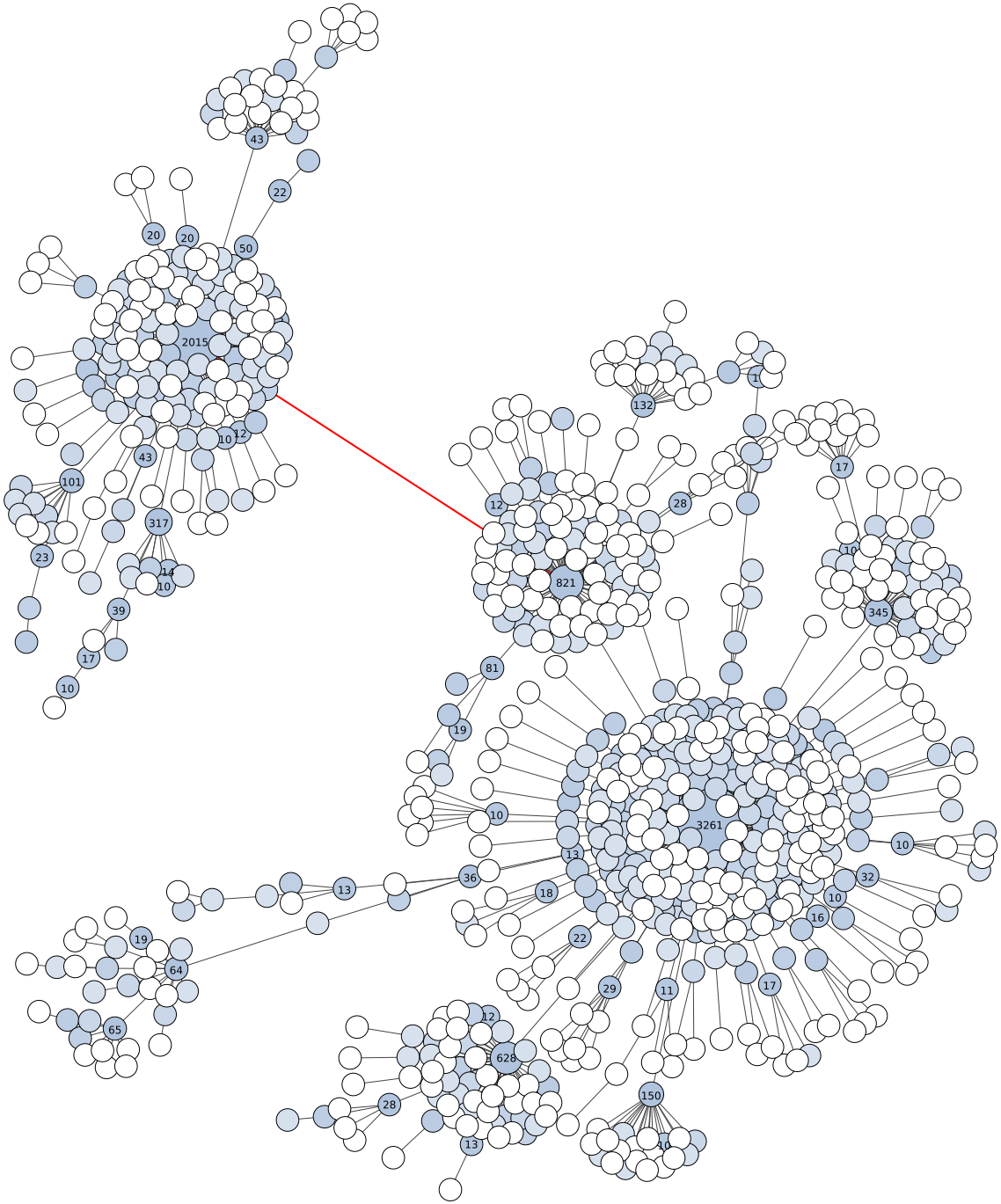


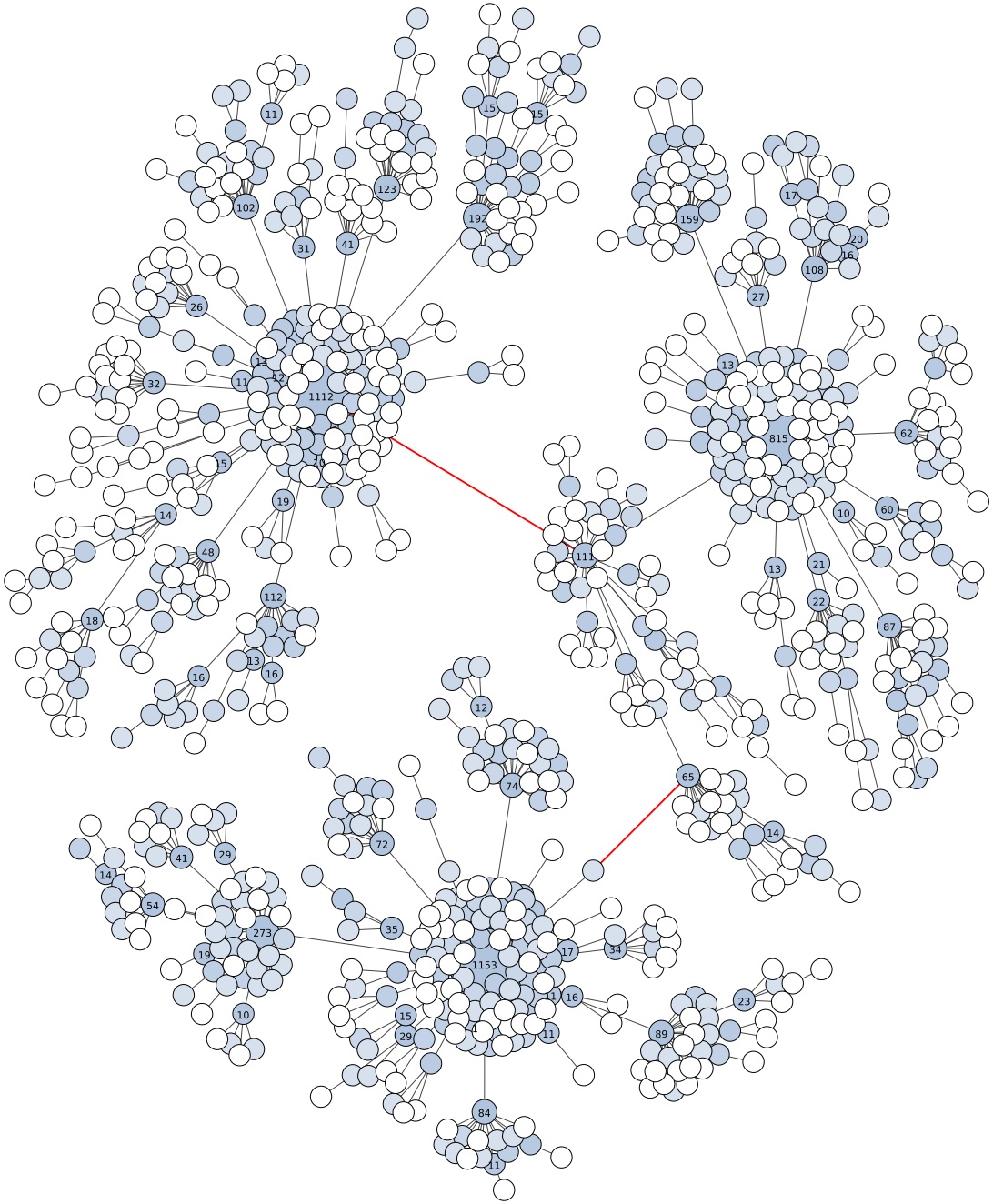
Swarm produces high-resolution clustering results



Circles are sequences, edges represent one difference (substitution or indel)

Less than 1% divergence (3 differences) between the two peaks of abundance


43
22
20
20
50
2015
10
12
43
101
317
23
14
10
39
17
10
1
132
17
12
28
10
821
345
81
19
10
3261
10
13
36
18
32
10
16
13
19
64
22
29
11
17
12
65
628
28
150
13
10

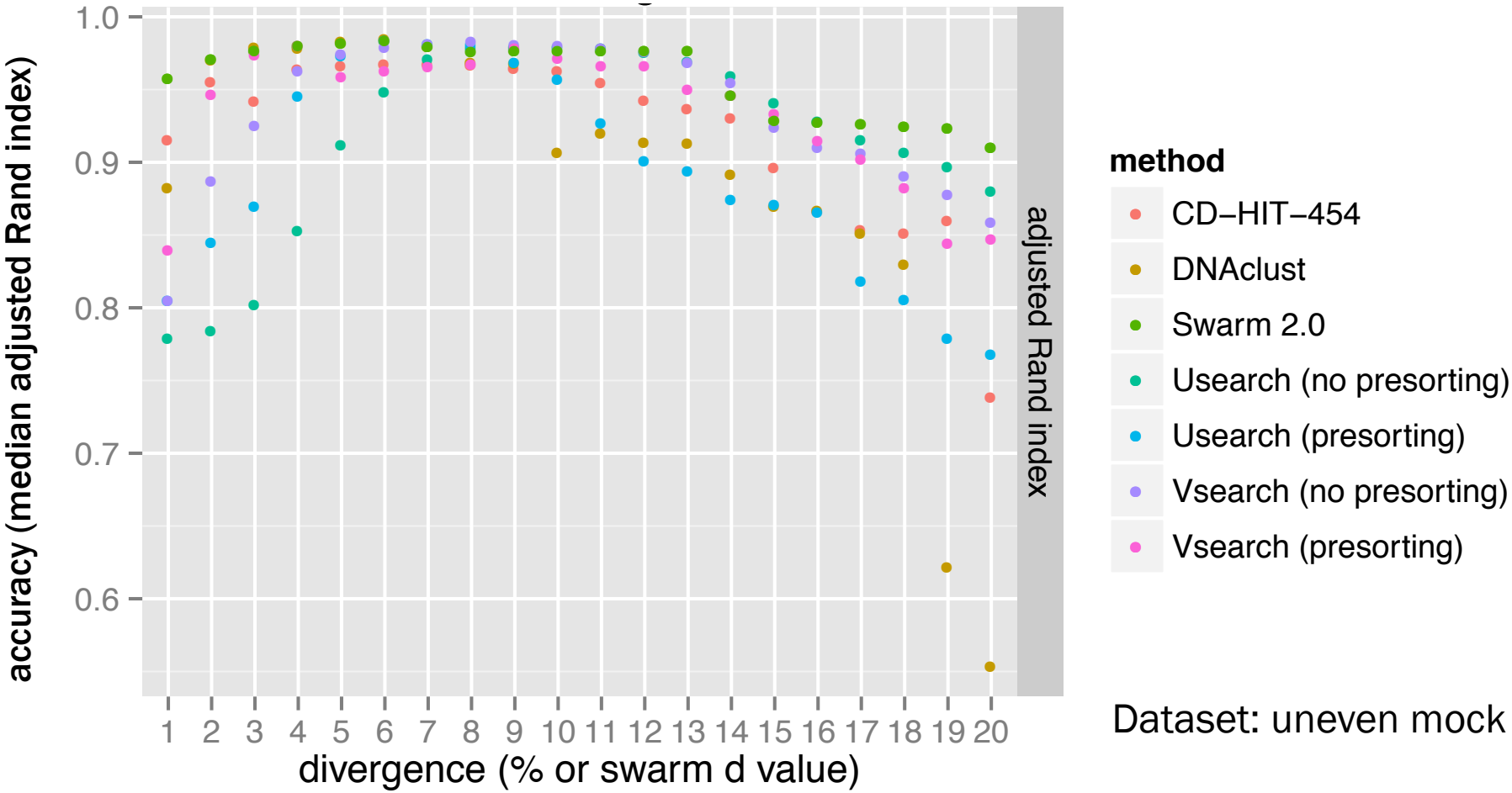

11
15
15
123
102
193
159
17
20
31
41
108
16
27
26
13
12
32
11
1112
62
815
15
14
19
10
60
48
111
13
21
112
22
87
18
13
16
16
12
74
65
14
72
41
29
14
54
273
35
1153
17
34
19
10
11
16
23
15
29
11
89
84
11

Clustering accuracy

- Tested the ability of methods to cluster together organisms that are already classified as belonging to the same clade in the tree of life
- 2 datasets: even and uneven mock communities with 57 different species, 16S rRNA V4, MiSeq 250bp PE
- 100 different sequence orders (by shuffling)
- 5 methods: CD-HIT, DNACLUST, SWARM, USEARCH (both with and without presorting by length)
- 20 clustering thresholds (divergence of 1-20%)
- 3 metrics: adjusted rand index, recall and precision
- Median values plotted

Clustering accuracy (uneven)

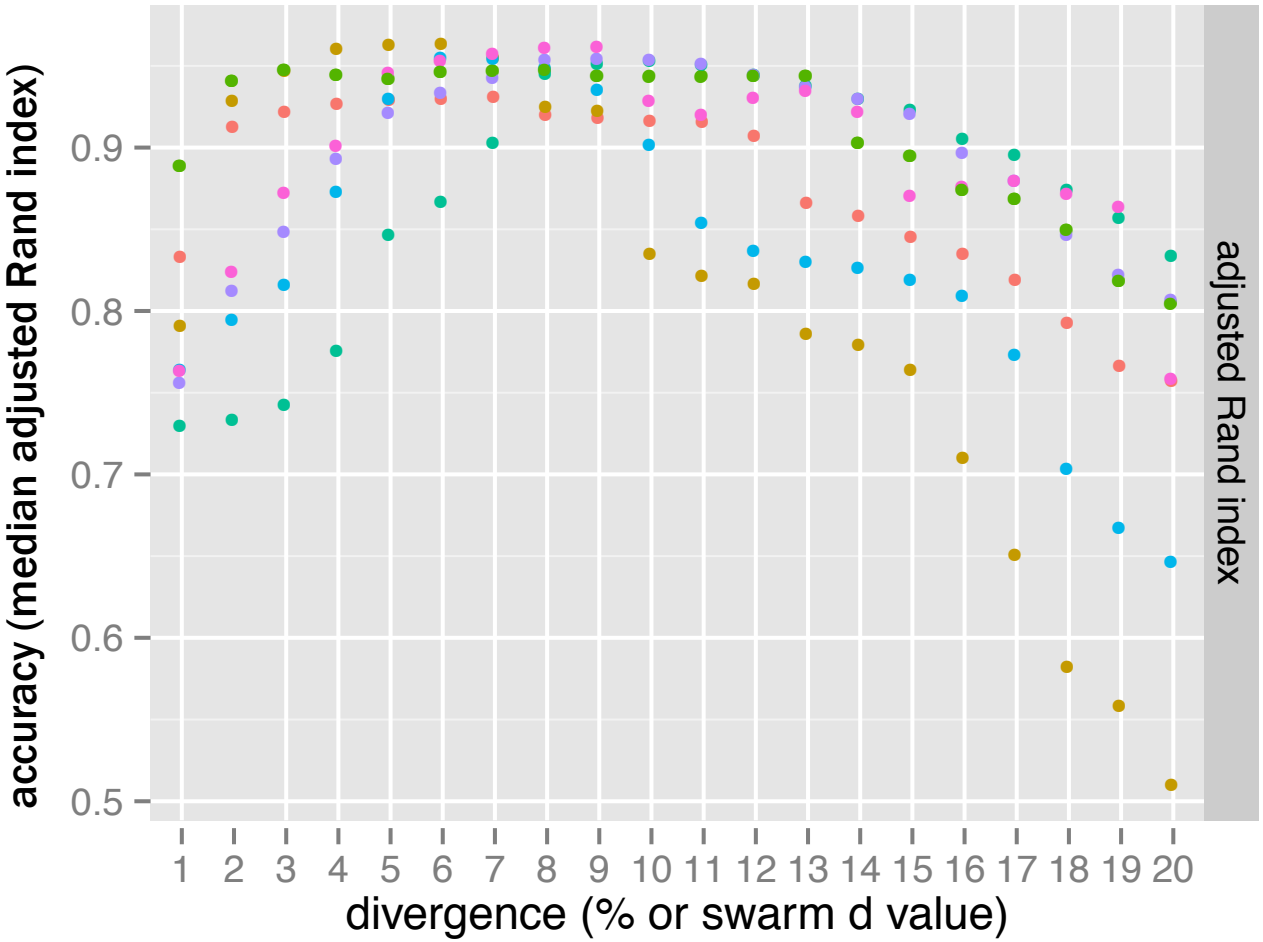
Ability of methods to correctly cluster together sequences of the same clade



Dataset: uneven mock

Clustering accuracy (even)

Ability of methods to correctly cluster together sequences of the same clade



method

- CD–HIT–454
- DNAclust
- Swarm 2.0
- Usearch (no presorting)
- Usearch (presorting)
- Vsearch (no presorting)
- Vsearch (presorting)

Dataset: even mock

SWARM 2 Performance

Dataset	Fastidious	Elapsed time (hh:mm:ss)	Max resident memory (GB)	OTUs	OTU reduction
small	N	01:45:07	41	49,138,023	
small	Y	04:59:05	114	29,485,078	40,0%
large	N	03:41:16	86	107,620,192	
large	Y	11:28:26	239	65,520,237	39,1%

- Fastidious mode seems to increase time and memory by a factor of about 3 while reducing the number of OTUs by about 40%, compared to a regular run with d=1.
- Clustering an even smaller dataset with UCLUST was estimated to take about 5 months (Rideout et al 2014). The same is probably true for swarm with d>1.

Small EMP dataset: 154,896,650 identical amplicons from 1,277,640,415 reads
Large EMP dataset: 314,871,149 identical amplicons from 2,254,207,945 reads
Using 8 cores and threads on an Abel cluster node.

Publications

Swarm 1:

- Mahé F, Rognes T, Quince C, de Vargas C, Dunthorn M. (2014) **Swarm: robust and fast clustering method for amplicon-based studies.** PeerJ 2:e593. doi: 10.7717/peerj.593

Swarm 2:

- Mahé F, Rognes T, Quince C, de Vargas C, Dunthorn M. (2015) **Swarm v2: highly-scalable and high-resolution amplicon clustering.** PeerJ 3:e1420. doi: 10.7717/peerj.1420

Software availability

- SWARM available on GitHub: https://github.com/torognes/swarm
- Free
- Open source (GPL)
- Extensive documentation
- Precompiled binaries available
  - Linux x86 (64 bit)
  - Mac OS X (64 bit)
- Available for *de novo* OTU picking in QIIME 1.9

Main collaborators

Frédéric Mahé
CIRAD, Montpellier, France

Tomáš Flouri
Heidelberg Institute for
Theoretical Studies, Germany

Christopher Quince
Warwick & Glasgow Univ., UK


Thank you!