ReClustOR User's Guide

Re-Clustering tool using an Open-Reference method that improves OTU definition

ZENODO repository: http://doi.org/10.5281/zenodo.2597403
Version 1.0; September 2019

Sébastien TERRAT, Christophe DJEMIEL, Corentin JOURNAY
UMR 1347 Agroécologie, AgroSup Dijon, INRA,
Univ. Bourgogne Franche-Comté, F-21000 Dijon, France
Contact: sebastien.terrat@inra.fr

CONTENTS

INTRODUCTION

ReClustOR is a new post-clustering method (for RE-CLUSTering method using an Open-Reference approach) to improve OTU consistency. This new strategy combines two previously-described clustering methods. Firstly, a classical clustering method (*e.g.* Swarm, or VSEARCH) is used to define OTU centroids and create a reference database. Secondly, a closed- or open-reference method (depending on the user's choice) is computed for all reads which are not considered as OTU centroids.

This software is composed of two different PERL modules (v5.26.1 or higher), **`ReClustOR_db.pl`** and **`ReClustOR.pl`**. The first program **`ReClustOR_db.pl`** takes as input several files (clustering file, taxonomy file, dereplication details file and FASTA sequence file) to create a specific database. Then, you can use the **`ReClustOR.pl`** to compare to the defined database (or a new one) the given input sequences.

Main interests of ReClustOR are that it overcomes problems associated with classical clustering methods (stability and reliability of OTUs) and consequently increases the quality and the congruence of the reconstructed OTUs. Moreover, the OTUs database defined with ReClustOR can be used as reference(s) with gradual enrichment of it by merging new studies and samples. Finally, this post-clustering step improved the congruence of obtained results (alpha-diversity, beta-diversity, composition) whatever the initial clustering method used.

INSTALLATION

Quick installation instructions

ReClustOR is distributed as PERL source code (v5.26.1 or higher). It comprises two independent modules: the first one is dedicated to OTUs reference database development (**`ReClustOR_db.pl`**), and the second one is used for clustering against a given database (**`ReClustOR.pl`**). These programs were designed to be used on UNIX platforms. They have been developed on Intel GNU/Linux systems, and intermittently tested on a variety of other UNIX platforms. They have also been tested on Apple OS/X. However, they were not currently tested on either Microsoft Windows, but they should work there, if the system answers to required dependencies (see Dependencies paragraph below).

Download the source (**`ReClustOR.zip`**) from: *http://doi.org/10.5281/zenodo.2597403* (available Zenodo repository).
Unpack the software:

```
> gzip -d ReClustOR.zip
```

Then, in the newly created top-level directory, both programs will be available. No compilation is needed. However, these programs are based on third-party tools, check the Dependencies paragraph below.

Dependencies

The ReClustOR programs (**`ReClustOR_db.pl`** or **`ReClustOR.pl`**) are based on two third-party programs:

- the Infernal alignment tool (1.1.1 or higher) and,
- the Inline PERL module (0.80 or higher).

By default, the program will check if these two mandatory tools are already installed or not. If it is not the case, these two programs are required, and must be installed by you.

INFERNAL alignment tool

Infernal ('INFERence of RNA ALignment') is a software package that allows you to make consensus RNA secondary structure profiles, and use them to search nucleic acid sequence databases for homologous RNAs, or to create new structure-based multiple sequence alignments. You can use the defined profile to align a set of unaligned sequences to the profile, producing a structural alignment, using the program **`cmalign`**. This allows you to build hand-curated representative alignments of RNA sequence families, then use a profile to automatically align any number of sequences to that profile. <u>This seed alignment/full alignment strategy combines the strength of stable, carefully human curated alignments with the power of automated updating of complete alignments as sequence databases grow. This is the strategy used to maintain the RFAM database of RNA multiple alignments and profiles.</u>

Here, Infernal is used for searching DNA sequence databases for RNA structure and sequence similarities. It is an implementation of a special case of profile stochastic context-free grammars called covariance models (CMs). A CM is like a sequence profile, but it scores a combination of sequence consensus and RNA secondary structure consensus, so in many cases, it is more capable of identifying RNA homologs that conserve their secondary structure more than their primary sequence.

Details and Download: http://eddylab.org/infernal/

Inline module

The Inline module allows you to put source code from other programming languages directly 'inline' in a Perl script or module. The code is automatically compiled as needed, and then loaded for immediate access from Perl.

Inline saves you from the hassle of having to write and compile your own glue code using facilities like XS or SWIG. Simply type the code where you want it and run your Perl as normal. All the hairy details are handled for you. The compilation and installation of your code chunks all happen transparently; all you will notice is the delay of compilation on the first run.

The Inline code only gets compiled the first time you run it (or whenever it is modified) so you only take the performance hit once. Code that is Inlined into distributed modules (like on the CPAN) will get compiled when the module is installed, so the end user will never notice the compilation time.

Details and Download: http://search.cpan.org/~ingy/Inline-0.80/lib/Inline.pod

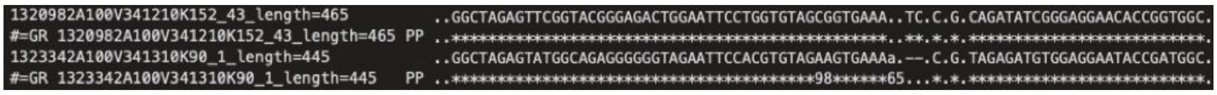
SOME DETAILS BEFORE STARTING

ReClustOR software was based on several elements (Multiple Sequence Alignment, specific clustering program, etc.) essential to understand before using it. To simply describe these elements, their importance, and why we chose to implement ReClustOR and how these elements are combined in ReClustOR, we chose to write this specific part of the User's Guide.

Multiple Sequence Alignment

All given sequences (those used as references in defined databases, or those compared to databases) will be firstly aligned using the Infernal third-party program (see this section for more details). Indeed, all sequences are globally aligned against dedicated structures for 16S rRNA bacteria and/or archaea, 18S rRNA fungi, or 23S plastidial algae genes depending of their origin, using the Infernal alignment program ('INFERence of RNA ALignment', v1.1.1). These dedicated structures are integrated in the ReClustOR program, in a specific folder (**`model/`**)

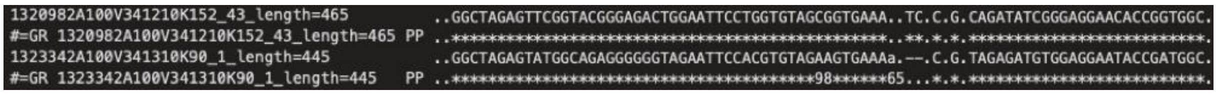
The use of a model to structurally align any number of new sequences to your consensus structure is very efficient, and allow to maintain representative seed alignments that are stable and small enough to be human-curated, while still being able to automatically incorporate and align all homologues rapidly. Moreover, the Infernal alignment against a structural model allows divergent but valid sequences to persist, whereas stricter methods may incorrectly discard such reads (Lynch & Neufeld, 2015) (see example of reads aligned with Infernal below).



```
1320982A100V341210K152_43_length=465         ..GGCTAGAGTTCGGTACGGGAGACTGGAATTCCTGGTGTAGCGGTGAAA..TC.C.G.CAGATATCGGGAGGAACACCGGTGGC.
#=GR 1320982A100V341210K152_43_length=465 PP ..************************************************..**.*.*.**************************.
1323342A100V341310K90_1_length=445           ..GGCTAGAGTATGGCAGAGGGGGGTAGAATTCCACGTGTAGAAGTGAAAa.--.C.G.TAGAGATGTGGAGGAATACCGATGGC.
#=GR 1323342A100V341310K90_1_length=445   PP ..***************************************98******65...*.*.**************************.
```

In the aligned sequences, a "." character indicates an inserted column relative to consensus; the "." Character is an alignment pad. The "-" character is a deletion relative to consensus.

This alignment provides also a more intuitive handling of sequencing errors, such as homopolymer errors, easily detected, due to the secondary-structure aware aligner. Such homopolymer errors can be therefore easily managed and ignored during the clustering step Furthermore, as other Multiple Sequence Alignment algorithms, Infernal software is more efficient in terms of substitution detection (Rosenberg, 2005; Nguyen et al., 2016).
(see example of reads aligned with Infernal below, with highlighted homopolymer potential difference).



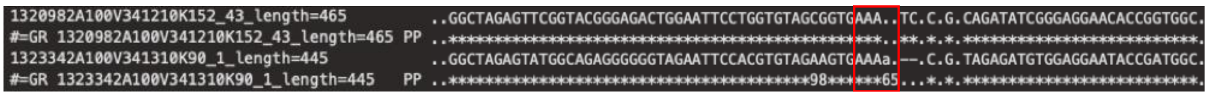
```
1320982A100V341210K152_43_length=465         ..GGCTAGAGTTCGGTACGGGAGACTGGAATTCCTGGTGTAGCGGTGAAA..TC.C.G.CAGATATCGGGAGGAACACCGGTGGC.
#=GR 1320982A100V341210K152_43_length=465 PP ..************************************************..**.*.*.**************************.
1323342A100V341310K90_1_length=445           ..GGCTAGAGTATGGCAGAGGGGGGTAGAATTCCACGTGTAGAAGTGAAAa.--.C.G.TAGAGATGTGGAGGAATACCGATGGC.
#=GR 1323342A100V341310K90_1_length=445   PP ..***************************************98******65...*.*.**************************.
```

In the aligned sequences, a small "a" character in the third line of the global alignment indicates a potential error (an insertion here) relative to the defined consensus. Such difference can be considered as a homopolymer difference, and it will be considered and potentially ignored during the clustering process.

Homopolymer Errors

Homopolymer stretches can be problematic for several sequencing technologies (*e.g.* 454, IonTorrent, MinION, etc.). These sequencing platforms suffer from the inaccuracy in detecting the length of homopolymers repeats of the same nucleotide. These homopolymer errors often lead to the inaccurate local alignment results, and become a critical barrier against accurate detection of genomic variations, or during clustering steps of homolog sequences. To explain this clearly, check these examples below.

Example 1:

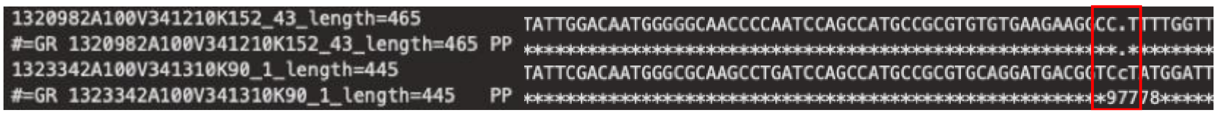


```
1320982A100V341210K152_43_length=465          ..GGCTAGAGTTCGGTACGGGAGACTGGAATTCCTGGTGTAGCGGTGAAA..TC.C.G.CAGATATCGGGAGGAACACCGGTGGC.
#=GR 1320982A100V341210K152_43_length=465 PP ..************************************************..**.*.*.**************************.
1323342A100V341310K90_1_length=445            ..GGCTAGAGTATGGCAGAGGGGGGTAGAATTCCACGTGTAGAAGTGAAAa.--.C.G.TAGAGATGTGGAGGAATACCGATGGC.
#=GR 1323342A100V341310K90_1_length=445    PP ..****************************************98******65...*.*.**************************.
```

Here, two sequences are compared by the clustering program after global alignment with Infernal. As Infernal is based also on the secondary structure of the rRNA, it can easily detect insertion-deletion errors originated from the sequencing step. As one of the two sequences has a high occurrence (here, 43 occurrences, indicated in its ID, see Input File Structure section for more details), the difference is ignored by the clustering program. Indeed, the sequence with 43 occurrences (after strict dereplication) can be considered as a “true” sequence. The difference with the other sequence (with only one occurrence) will be ignored during the clustering process.

However, such differences cannot be detected on singleton sequences (if the two sequences have only one occurrence), as we are unable to confirm if there is a ‘true biological’ difference or not.

Example 2:



```
1320982A100V341210K152_43_length=465          TATTGGACAATGGGGGCAACCCCAATCCAGCCATGCCGCGTGTGTGAAGAAGGCC.TTTTGGTT
#=GR 1320982A100V341210K152_43_length=465 PP *******************************************************.********
1323342A100V341310K90_1_length=445            TATTCGACAATGGGCGCAAGCCTGATCCAGCCATGCCGCGTGCAGGATGACGGTCcTATGGATT
#=GR 1323342A100V341310K90_1_length=445    PP ******************************************************9778******
```

Moreover, homopolymer differences are only evaluated if a stretch of three identical bases are present in one of the compared sequences (or at least two bases and a gap, as shown in the example 2).

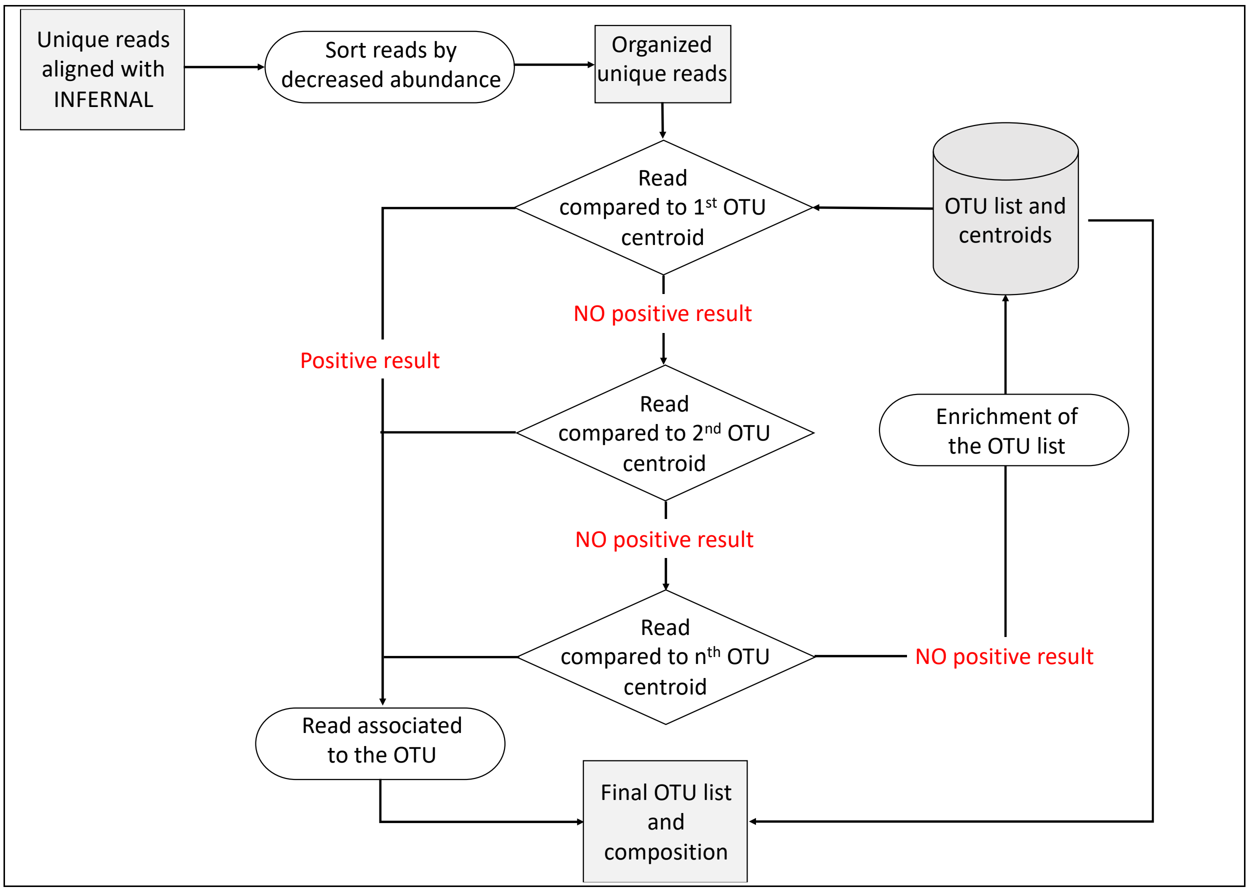
Ignoring homopolymer differences can drastically reduce the number of defined OTUs (between 5 to 20%, depending on the origin of the sequencing, and the quality of the sequencing). Such errors are known, and were already considered in other programs, such as CRUNCHCLUST (https://code.google.com/archive/p/crunchclust/). To our knowledge, this CRUNCHCLUST program was the first to ignore such differences for the clustering step.

Specific Clustering Process in ReClustOR

First of all, both programs (`ReClustOR_db.pl` or `ReClustOR.pl`) needs dereplicated reads (see Input File Structure section for more details). These dereplicated reads will be then aligned using Infernal alignment (Nawrocki & Eddy, 2013), and after a first step of organization of input sequences by decreasing abundance, they will be clustered based on the threshold chosen by the user.

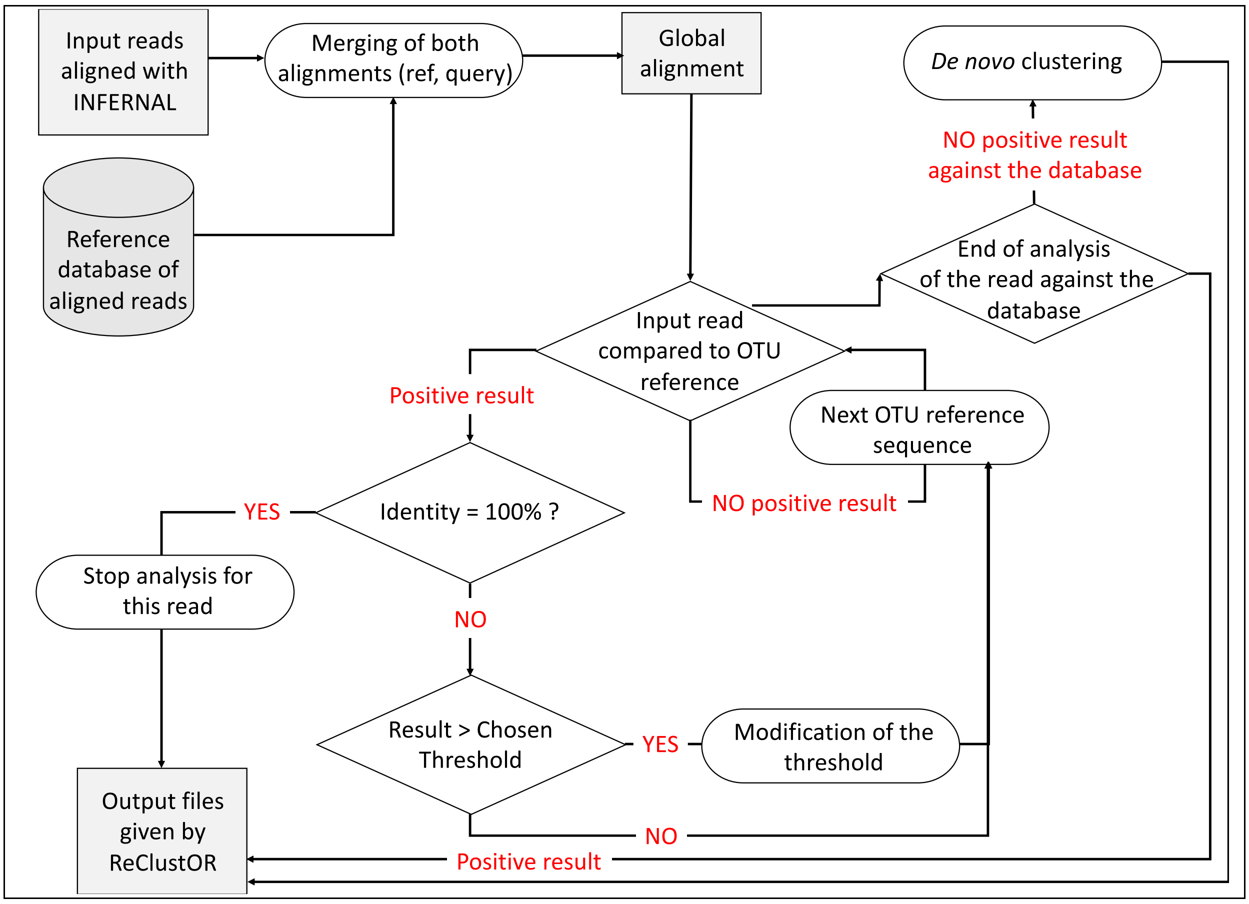
Our clustering program will compare two globally aligned sequences and determine their number of differences. To determine this number of differences, the clustering program relied on a Levenshtein distance. Levenshtein distance is a metric for measuring the amount of differences between two sequences (*ie* an edit distance). As the amplicon sequences tend to start at the same place, Levenshtein distance is the most appropriate and most efficient measure for calculating distances between them.

Moreover, our clustering process is a is a greedy incremental clustering algorithm developed in C (introduced in PERL by the Inline specific module). The first sequence from the dataset becomes the seed of the first cluster (this first seed is also the most abundant sequences, and potentially the truest one). Then, the distance between the seed and each remaining sequence is compared. If the distance of the query to the seed sequence is equal or below a given threshold then it is assigned to that cluster. Otherwise, a new cluster is defined with that sequence as the seed. For more details, please read the scheme below, describing the clustering process:


Unique reads aligned with INFERNAL
Sort reads by decreased abundance
Organized unique reads
Read compared to 1st OTU centroid
OTU list and centroids
NO positive result
Positive result
Read compared to 2nd OTU centroid
Enrichment of the OTU list
NO positive result
Read compared to nth OTU centroid
NO positive result
Read associated to the OTU
Final OTU list and composition

Main process of our clustering program. The OTU list and their corresponding centroids (or seeds) will be stored and enriched during the treatment of organized reads (by decreasing abundance).

Moreover, it is important to note that, against a defined database, the clustering program <u>will not be stopped</u> after a positive result is found (see scheme below).


Input reads aligned with INFERNAL
Merging of both alignments (ref, query)
Global alignment
De novo clustering
Reference database of aligned reads
NO positive result against the database
End of analysis of the read against the database
Input read compared to OTU reference
Positive result
Next OTU reference sequence
YES
Identity = 100% ?
NO positive result
Stop analysis for this read
NO
Result > Chosen Threshold
YES
Modification of the threshold
Output files given by ReClustOR
NO
Positive result

Main process of our clustering program of input sequences against the chosen database. Here, the clustering process is quite different, as if a positive result is found, the clustering threshold is modified to consider this positive result, and to check if a better result will be found with other reference sequences, or not. As indicated, the program will be stopped for a specific sequence if: (i) No results were found against all reference sequences, (ii) 100% of identity was found between a reference sequence and the analyzed sequence, (iii) after all comparisons, the closest reference sequence was found. For sequences with no results against all reference sequences, a step of *de novo* clustering is realized to define OTUs with only these sequences. These specific OTUs are called “OUTs” and not “OTUs”. This is detailed in the Summary_results.txt output file.

INPUT FILE STRUCTURE

Sequence ID structure

All sequence Identifications (ID) must have a specific format to be accepted by the Database Definition Program (**`ReclustOR_db.pl`**). The format is simple. Each sequence has to be defined by an ID, the number of dereplicated reads and its length, all elements separated by '**_**' characters as shown below.

Example:
```
M0098755_6_length=374
```

This sequence had the ID: **`M0098755`** (specific length not defined, but composed only with alphanumeric characters (a-z,A-Z,0-9). This sequence corresponded to 6 reads in the dataset, with a length of 374 bases.

The sequence must be also affiliated to a specific sample ID added to the name (separated by a '**|**' character):

Example:
```
00001|M0098755_6_length=374
```

Here, this sequence **`M0098755_6_length=374`** came from the sample 00001. The ID for a sample must be a number only (0-9). Its length is not defined.

Standard FASTA file details

All sequence Identifications (ID) must have a specific format to be accepted by the Database Definition Program. (see Sequence ID Structure section). The standard FASTA file must contain at least sequences considered as representative sequences OTUs, or all sequences. For the comparison step, all query sequences will be compared to the chosen database.

Clustering file details

The clustering file had a specific format to be loaded by the PERL program. Each line contains four elements, separated by tabulations. The line starts by the OTU number (1 or OTU1), then the filename, then the number of reads in the OTU and the IDs of all reads associated within this OTU, separated by a space.

Example:
```
1    filename  12   00003|M0098755_6_length=374 00001|M0098758_6_length=374
2    filename  5    00002|HB018GD8_5_length=371
```

It is important to highlight that the seed of the OTU (the OTU representative sequence) is the first one in the list of each OTU. So, this sequence must be available in the FASTA file given in input.

Taxonomic file details

Regarding the taxonomic file, it must have also a specific format to be used by the program. For each OTU, five taxonomic levels must be defined (phylum, class, order, family and genus), separated by a tabulation. For each taxonomic level, one name must be given, with or without the number of corresponding sequences affiliated in this OTU in parenthesis. **The OTU number and order must be the same for the taxonomic file and the clustering file to link both files.**

Example:
```
OTU32  Actinobacteria(70) Acidimicrobiia(70) Acidimicrobiales(70)     Iamiaceae(70)     Iamia(70)
```

It is also possible to give several names for a taxonomic level, separated by a space (see example for the genus level below). In the case where several names are available, the number of sequences associated to this taxonomy must be given to select the highest one automatically.

Example:
```
OTU32   Actinobacteria(70)    Acidimicrobiia(70)    Acidimicrobiales(70)  Iamiaceae(70)  Unknown(20) Iamia(50)
```

Finally, all not affiliated groups must be called 'Unknown', to be clearly identified by the Database Definition Program (`ReclustOR_db.pl`). All other names will be considered as known organisms.

Dereplication file details

This file is used to detail the composition of dereplicated sequences encompassing identical sequences from several samples. To define the database properly, the program must know the detailed organization of all samples in all OTUs. The specific format of this file is simple, as it is composed of lines starting with the sequence ID encompassing the sequences from several samples. The detail is given in the second part of the line, with each sample sequence separated by a space. Both parts of the line are separated by a tabulation.

Example:
```
00003|M0098755_6_length=374  00003|M0098755_3_length=374 00001|M0012749_3_length=374
```

Here, this sequence **`M0098755_6_length=374`** encompass three sequences from the sample 00003 and three from the 00001. In the clustering file, only the ID sequence **`00003|M0098755_6_length=374`** will be found.

TUTORIAL

Here's a simple tutorial to understand how ReClustOR programs work. The subdirectory (**`Data_example/`**) in the ReClustOR distribution contains the files used in the tutorial, and highlight various file formats that ReClustOR reads and uses.

Please, create a new directory that you can work in, and copy all the files in **`Data_example/`** there. We assume for the following examples that you've installed the ReClustOR programs in your path.

If not, you'll need to give a complete path name to the programs (*e.g.* something like **`perl /usr/people/terrat/ReClustOR/ReClustOR_db.pl`** instead of just **`perl ReClustOR_db.pl`**).

The Programs in ReClustOR

These are two main modules in the ReClustOR package:

`ReClustOR_db.pl`	define new database of reference sequences
`ReClustOR.pl`	compare given sequences to an existing database of reference sequences.

ReClustOR_db

Usage

```
ReClustOR_db.pl [-option] <filename>
```

Mandatory options

`-clust`	clustering filename needed to create the database
`-taxo`	taxonomy filename needed to create the database
`-derep`	dereplication information filename to create the database
`-seq`	FASTA file containing the sequences analyzed

Example

The given example is based on files available in **Data_example/** directory, and more specifically, those stored in the **Files_For_DB_Definition/** subdirectory.

```
> perl ReClustOR_db.pl -clust Cluster_file.clust -taxo Taxo_file.txt
-derep Derep_details_file.fasta -seq Fasta_file.fasta
```

First of all, the **ReClustOR_db.pl** will check if the various files have correct structures and format.

```
# Dereplication file format checked and passed
# Sequence file Fasta_file.fasta format checked and passed
# Taxonomic file format checked and passed
# Clustering file format checked and passed
```

Then, the **ReClustOR_db.pl** will ask for more information to define efficiently all files for the reference database:

```
Please choose the name of the database (short name whithout spaces
or special characters).
==>
```

As an answer, we can provide the term '**EXAMPLE**'

```
Will you want to keep the single-singletons in the created database
(yes-no)?
==>
```

As an answer, we can provide the term '**yes**' as we want to keep all potential OTUs from our dataset for further analyses.

```
Please indicate the similarity level defined for the clustering step
of your samples? (default: 95)[1-100].
==>
```

As an answer, we can provide the value by default (**95**). If no value is given, the value by default will be applied. This value is only given for information, no clustering is realized in this step.

```
Please indicate the level used to define the sequence taxonomy of
your samples? (default: 80)[50-100].
==>
```

As an answer, we can provide the value by default (**80**). If no value is given, the value by default will be applied. This value is only given for information, no taxonomic assignment is realized in this step.

```
Please indicate the targeted organism (to define the potential model
used in the global alignment) [bacteria/fungi/archaea/algae]
==>
```

Here, as file examples are specific of bacteria organisms, it is advised to specify **`bacteria`** (for 16S structures).

Then, the program will indicate which step of analysis is realized to create the database.

```
LAUNCHING THE EXTRACTION OF CLUSTERING DATA
END OF EXTRACTING CLUSTERING DATA
12 Seconds of treatment
LAUNCHING THE ANALYSIS OF THE CLUSTERING DATA
END OF ANALYSIS OF THE CLUSTERING DATA
13 Seconds of treatment
LAUNCHING THE ANALYSIS OF SAMPLE DISTRIBUTION
END OF ANALYSIS OF SAMPLE DISTRIBUTION
14 Seconds of treatment
LAUNCHING THE DEFINITION OF OTU SEQUENCE FILE
END OF DEFINITION OF OTU SEQUENCE FILE
15 Seconds of treatment
LAUNCHING THE ANALYSIS OF TAXONOMIC DATA
END OF ANALYSIS OF TAXONOMIC DATA
16 Seconds of treatment
LAUNCHING THE DATABASE DEFINITION FILES
END OF DATABASE DEFINITION FILES
17 Seconds of treatment
LAUNCHING THE ALIGNMENT OF DATABASE SEQUENCES
19 Seconds of treatment
END OF ALIGNMENT OF DATABASE SEQUENCES
200 Seconds of treatment
```

The **`ReClustOR_db.pl`** created a specific folder based on the given answers here: **`DB_EXAMPLE_BACTERIA_C95_T80_G/`** to store the output files.

Output files produced by ReClustOR_db

Three output files can be found in the database folder:

dbreads_EXAMPLE_BACTERIA_C95_T80_G	FASTA file containing reference sequences of each OTU
database_EXAMPLE_BACTERIA_C95_T80_G	Complete description (OTUname, number of sequences, composition, taxonomy) of each OTU
dbalign_EXAMPLE_BACTERIA_C95_T80_G	Alignment file (in Stockholm format) of reference sequences of each OTU

The **dbreads_EXAMPLE_BACTERIA_C95_T80_G** contains FASTA sequences, for each OTU (see example below).

```
>OTU000002798
GAAGGGTGCAAGCGTTACTCGGAATTACTGGGCGTAAAGCGTGCGTAGGTGGTTCGTTAAGTCTGATG
TGAAAGCCCTGGGCTCAACCTGGGAATTGCATTGGATACTGGCGAGCTAGAGTGCGGTAGAGGATGGC
GGAATTCCCGGTGTAGCAGTGAAATGCGTAGAGATCGGGAGGAACATCTGTGGCGAAGGCGGCCATCT
GGACCAGCACTGACACTGAGGCACGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAGTCCAC
GCCGTAAACGATGGGTGCTAGGTGTCGCGGgctttgacccCTGCGGTGCCGTAGCTAACGCATTAAGC
ACCCCGCCTGGGGAGTACGGCCGCAAGGCTAAA
>OTU000001372
GAAGGGGGCTAGCGTTGCTCGGAATCACTGGGCGTAAAGGGTGCGTAGGCGGGTTTTTAAGTCAGAGG
TGAAATCCTGGAGCTCAACTCCAGAACTGCCTTTGATACTGAGAATCTTGAGTATGGGAGAGGTGAGT
GGAACTGCGAGTGTAGAGGTGAAATTCGTAGATATTCGCAAGAACACCAGTGGCGAAGGCGGCTCACT
GGCCCATAACTGACGCTGAGGCGCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCTGGTAGTCCAC
GCTGTAAACGATGAGTGCTAGGTATCGGGAgaatttcTTTCGGTTCCGTAGTTAACACGTTAAGCACT
CCGCCTGGGGAGTACGATCGCAAGATTAAA
```

The **database_EXAMPLE_BACTERIA_C95_T80_G** contains all information available for each OTU (see example below).

```
OTU000002798	1
	1.00_Proteobacteria|1.00_Deltaproteobacteria|1.00_Myxococcales|
1.00_Haliangiaceae|1.00_Haliangium	1_000000001
```

Here, each element is separated by a tabulation. The first element is the name of the OTU (**OTU000002798**), then, the number of sequences in this OTU (here, **1**). Then, its complete taxonomy, with for each level the percentage of sequences with this taxonomy, and the taxonomic name. Finally, the last element describes the sharing of this OTU between samples, based on information stored in IDs of each sequence.

Finally, the **dbalign_EXAMPLE_BACTERIA_C95_T80_G** file stored the global alignment of all reference sequences from the defined database (see example below).

```
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2

OTU000002798         -----------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------
-----------------------------....-----------------------------------
-----...-------------------------------......-----------------------
---------------------.......----------
GAAGGGTG.CAAGCGTT...ACTCGGA..ATTACTGGGCGTAAAGCGT.GCGTAGGTGGT.T.C.G.T
TAA.G.TCT.G..A.T.GTGAAAGCC.CT.G.GGCTCAA.............................
...................................CCTG.G.GAATTGCATTG.GA.TACTGG.C.G.
AGCT.A.G.AGTGCGGTA..GAGGATGGC.....G.GAATTCCCGGTGTAGCAGTGAAA.........
........TGCGTAGAGATCGGGAGGAA..C.ATCTG...T...G.GCGAAGGCGGCCATCTG.....
..........GAC.CA.G...................C..ACTGACACTGAG.G.C.ACGAAAGCGTG
GGGAGCAAACAGGATTAGATACCCTGGTAGTCCACGCCGTAAACGATGGGTGCTAGGTG.T.CGCGG.
...............gcttt........................................gacccCT.
GC.GGTGCCG.T.A.GCTAACGCATTAA..GCACCC.CGCCTGGGGAGTACGGCCGCAA.....GGCT
AAA
#=GR OTU000002798 PP
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
....................................................................
..................********.********...*******..*******************.*
**********.*.*.*.****.*.***.*..*.*.*********.**.*.*******...........
.....................................................****.*.********
***.**.******.*.*.****.*.*.*********..*********.....*.**************
*********.................********************..*.*****...*...*.****
*************...............***.**.*...................*..**********
**.*.*.*************************************************************
*********.*.*****................*****..............................
..........*******.**.*******.*.*.*************..******.*************
*********.....*******
```

ReClustOR

Usage

```
ReClustOR.pl [-option] <filename>
```

Mandatory options

-db	Database folder containing the database definition files
-th	Threshold chosen by the user for the comparison with the database
-cl	Chosen clustering type [infern/needle]
-sq	input FASTA file that will be compared to the database

By default, the program will be based on a comparison of input sequences against the database given as a mandatory option (**-db**). The user must also give a specific threshold level for the clustering step to avoid the selection of too many results.

The clustering type is also mandatory. Two choices are available for the user:

- infern	Based on a global alignment of all reads against the database using the Infernal program.
- needle	Using also a global alignment of reads, but with an algorithm closed to the Needleman-Wunsch Algorithm (low, and not optimized).

Additional options

Computing and parallelization analysis

-cp <i> Number of cpus that the program will be able to use (default: 1).

Optional output filename

-o <f> Results will be written in the <f> folder and not in the Results folder

Optional clustering parameters

-c	ignoring uncommon parts of compared reads at both extremities (default: not activated)
-h	ignoring homopolymer errors between reads (default: not activated)
-hc	ignoring both uncommon parts of compared reads at both extremities and homopolymer errors between reads (default: not activated)
-ad	add sequences with no similarities (at the defined threshold) against the database as new OTUs of the database. More precisely, the sequences will be integrated to the chosen database (**-db** option) to enrich it after *de novo* clustering. **BE CAREFUL WHEN USING THIS OPTION, AS THE DATABASE WILL BE MODIFIED, BUT WILL KEEP ITS INITIAL NAME.**

Example

The given example is based on files available in **`Data_example/`** directory, and more specifically, those stored in the **`File_For_Clustering/`** subdirectory.

```
> perl ReClustOR.pl -db DB_EXAMPLE_BACTERIA_C95_T80_G -th 95 -cl infern -sq Fasta_file.fasta
```

First of all, the **`ReClustOR.pl`** will check if the various files have correct structures and format.

```
# Sequence file Fasta_file.fasta format checked and passed
# Sequence file dbreads_EXAMPLE_BACTERIA_C95_T80_G format checked
and passed
1 seconds of treatment
# Temporary and Results/ folders checked and cleaned
```

Then, the program will realize the treatment and the comparison of the sequences given in the FASTA file to the dedicated database.

```
STARTING GLOBAL ALIGNMENT OF THE DATABASE AND THE SEQUENCES
END OF GLOBAL ALIGNMENT AND EXTRACTING DATA
1 Seconds of treatment
END OF ORGANIZING DATA
2 Seconds of treatment
STARTING CLUSTERING DATA AGAINST THE DATABASE
2 Seconds of treatment
END OF CLUSTERING DATA AGAINST THE DATABASE
2 Seconds of treatment
STARTING CLUSTERING DATA USING A DE NOVO APPROACH
2 Seconds of treatment
END OF CLUSTERING DATA USING A DE NOVO APPROACH
2 Seconds of treatment
STARTING RESULTS FORMATTING
2 Seconds of treatment
ENDING RESULTS FORMATTING
2 Seconds of treatment
```

The **`ReClustOR.pl`** created a specific folder (**`Results/`**) to store the output files.

Output files produced by ReClustOR

Four files are produced by the main program (ReClustOR.pl).

`Data_from_db_database.txt`	Complete description of the given dataset against the defined database (OTUs, abundances, taxonomies, etc)
`Matrix_data.txt`	OTU matrix defined using the given dataset of sequences against the database
`Raw_clust_results.clust`	Description of each clustering result for each sequence against the database
`Summary_results.txt`	Summary results after clustering

The **`Data_from_db_database_EXAMPLE_BACTERIA_C95_T80_G.txt`** file summarize the results for the treated dataset of sequences against the database. It will give all OTU names with at least one sequence matching, the number of sequences associated to this OTU in your dataset, the number of sequences associated to this OTU in the defined database, its taxonomy when you constructed the database, and also its representativity when the database was defined (see example below).

```
OTU	Number of reads from your sample	Number of reads for this OTU in the database	Defined taxonomy	Sample representativity
OTU000000007	1	167	1.00_Actinobacteria|0.99_Thermoleophilia|0.99_Solirubrobacterales|0.89_Solirubrobacteraceae|0.89_Solirubrobacter	18_000000001|13_000000002|26_000000003|17_000000004|11_000000005|11_000000006|18_000000007|11_000000008|20_000000009|8_000000010|7_000000011|7_000000012
```

Here, the **OTU000000007** is found in your dataset. More precisely, **1** sequence matched with the reference sequence of this OTU, composed of **167** sequences when the database was defined. This OTU was mainly composed of **Actinobacteria** microorganisms (**100%** of sequences), mainly **Solirubrobacter** (**89%** of sequences). This OTU was found in various samples, with various abundances when the database was defined (**18_000000001|13_000000002|26_000000003|17_000000004|11_000000005|11_000000006|18_000000007|11_000000008|20_000000009|8_000000010|7_000000011|7_000000012**).

The second file is the **`Matrix_data.txt`** file. This file is only a file containing the contingency of all sequences for all defined 'OTUs' (and 'OUTs' if any). It will be useful for further analyses (statistical analyses in R for example). OTUs are organized in columns, and samples in lines, separated by tabulations.

The third file is the **`Raw_clust_results.clust`** file. It compiles the results obtained for each sequence given as input against the defined database. More precisely, each line describes the results: the first element is the ID of the input sequence, the second result the percentage of similarity against the third element, the OTU name of the database. For example, the sequence **`AACJA1110712441192911N028_1_length=372`** is **`99.46%`** identical to the reference sequence of the **`OTU000000774`** from the defined database. For sequences that did not match to OTUs of the database, they will be also printed in this file, but associated to 'OUT' OTUs (see last line of the example below). They will be considered as the OTU seed of the new 'OUT' OTUs is they 100% of similarity with this OTU, or associated to this new OTU if they have an identity score below 100%.

```
AACJA1110310453223441N028_1_length=373      96.25 OTU000000007
AACJA1110712441192911N028_1_length=372      99.46 OTU000000774
AACJA1110716895155501N028_1_length=373      96.25 OTU000000072
AACJA111072141102431N028_1_length=369 100.00      OTU000001377
AACJA111079999103525N028_1_length=372 100.00      OUT000000001
AACJA111074119203525N028_1_length=371 99.55 OUT000000001
```

Finally, the last output file (**`Summary_results.txt`**) summarize all the results for the user, as seen below in the example.

```
Total number of reads:     12
Number of reads associated to OTUs from the database: 10
Number of reads associated to OTUs de novo 2
Total number of OTUs (database and de novo):     11
Number of OTUs associated to OTUs from the database:  10
Number of OTUs defined de novo:  1
```