Amplicon_sorter

A tool for reference-free sorting of ONT sequenced amplicons based on their similarity in sequence and length and for building solid consensus sequences. The limit for separating closely related species within a sample is currently around 95 - 96%.

(Only works on Linux because it uses multiprocessing)

Requirements:

- Python 3
- edlib: Lightweight, super fast C/C++ library for sequence alignment using edit (Levenshtein) distance (https://pypi.org/project/edlib/#description) (`python3 -m pip install edlib`)
- biopyton (`sudo apt-get install python3-biopython`)
- matplotlib (`sudo apt-get install python3-matplotlib`)

Citation:

Vierstraete, A. R., & Braeckman, B. P. (2022). Amplicon_sorter: A tool for reference-free amplicon sorting based on sequence similarity and for building consensus sequences. *Ecology and Evolution*, 12, e8603. https://doi.org/10.1002/ece3.8603

Options:

`-i, --input`: Input file in fastq or fasta format (auto-detect). Make sure the inputfile is named as *.fasta or *.fastq because it replaces the extension in parts of the script.
`-o, --outputfolder`: Save the results in the specified outputfolder. Default = current working directory

`-min, --minlength`: Minimum readlenght to process. Default=300
`-max, --maxlength`: Maximum readlenght to process. Default=No limit
`-maxr, --maxreads`: Maximum number of reads to process. Default=10000
`-ar, --allreads`: Use all reads from the inputfile between length limits. This argument is still limited with `--maxreads` to have a hard limit for large files.
`-np, --nprocesses`: Number of processors to use. Default=1
`-sfq, --save_fastq`: Save the results also in fastq files (fastq files will not contain the consensus sequence)
`-ra, --random`: Takes random reads from the inputfile. The script does NOT compare al sequences with each other, it compares batches of 1.000 with each other. You can use this option and sample reads several times and compare them with other reads in other batches. So, it is possible to have an inputfile with 10.000 reads and sample random 20.000 reads from that inputfile. The script will run 20 batches of 1.000 reads. This way, the chance to find more reads with high similarity is increasing when there are a lot of different amplicons in the sample. No need to do that with samples with 1 or 2 amplicons.

Less important options:

`-a, --all`: Compare all selected reads with each other. Only advised for a small number of reads (< 10000) because it is computation intensive. (In contrast with the default settings where it compares batches of 1.000 with each other)
`-ldc, --length_diff_consensus`: Length difference (in %) allowed between consensuses to COMBINE groups based on the consensus sequence (value between 0 and 200). Default=8.0. This can be interesting if you have amplicons of different length, the shorter ones are nested sequence of the longer ones and you want to combine those in one group.

`-sg, --similar_genes`: 'Similarity to sort genes in groups (value between 50 and 100). Default=80.0

`-ssg, --similar_species_groups`: Similarity to CREATE species groups (value between 50 and 100). Default=Estimate

`-ss, --similar_species`: Similarity to ADD sequences to a species group (value between 50 and 100). Default=85.0

`-sc, --similar_consensus`: Similarity to COMBINE groups based on the consensus sequence (value between 50 and 100). Default=96.0

`-ho, --histogram_only`: Only makes a read length histogram. Can be interesting to see what the minlength and maxlength setting should be or how many reads there are between selected read length treshold.

`-so, --species_only`: Only creates species groups and sort to species level. This can only be done if the whole script has run once without this option. This is to save time if you play with the `--similar_species` and/or `--similar_species_groups` parameters. It is using the same `--maxreads`, `--minlength`, `--maxlength` data that is produced in the first part of the script, so those 3 parameters are ignored here.

How it works (in short):

1. The script reads the inputfile and creates a read length histogram of all the reads in the file.
2. It starts processing a selection of the reads (based on minlength, maxlenght, maxreads). It saves the result in .group files that contain reads of the same gene (e.g. group_1 with 18S reads, group_2 with COI reads, group_3 with ITS reads).
3. It processes the gene group files to sort out the genes to species or genus level and saves this to different species files. (e.g. file_1_1.fasta is 18S from species1, file1_2.fasta is 18S from species2, …)
4. Each output file contains at the end the consensus file of the sequences in the file.
5. Files are produced per group with all consensus sequences for that group. A file is produced with all consensus sequences of all groups.
6. Files are produced with 'unique' sequences (script does not find a group where it belongs to according to the settings)
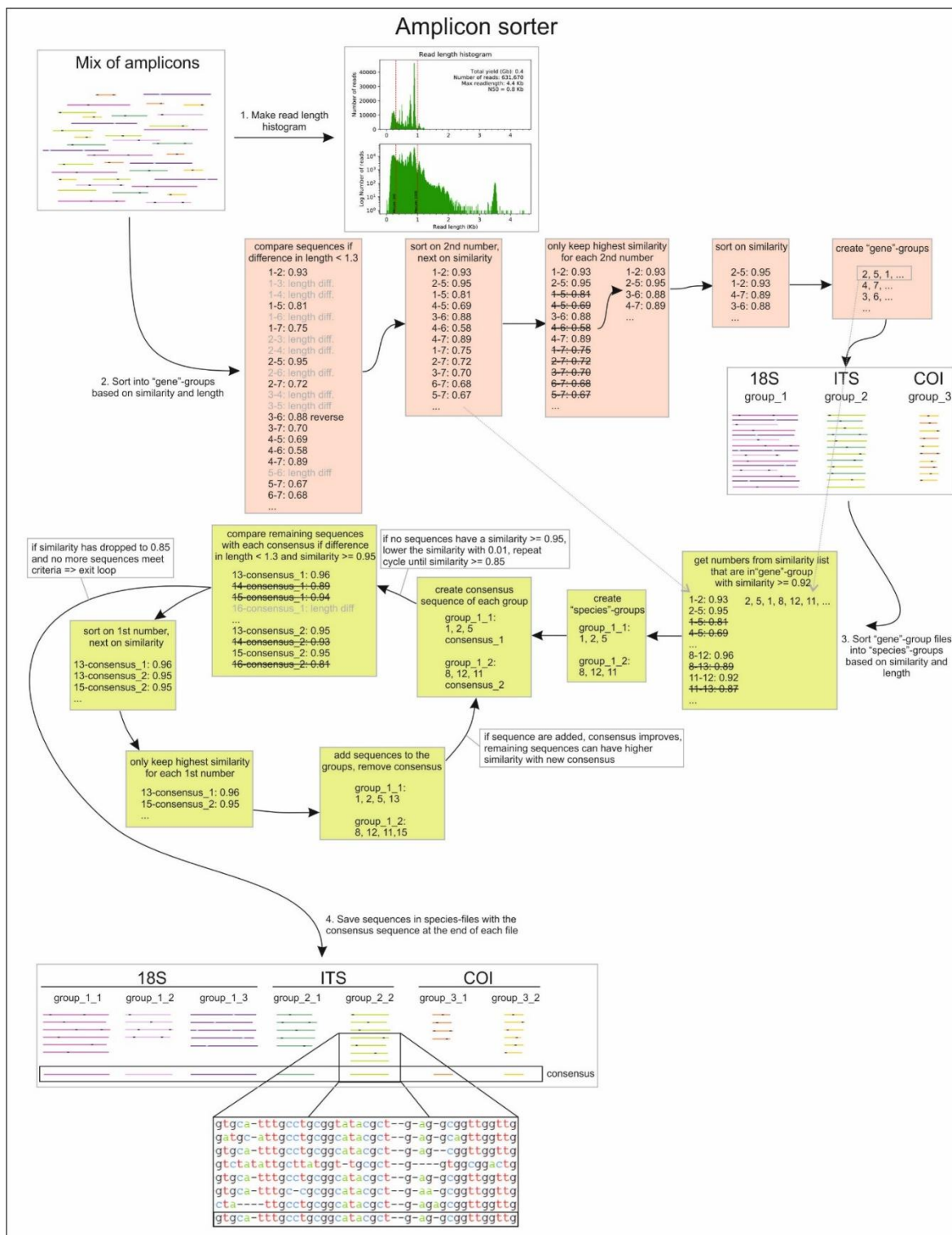
Workflow:

Filter your inputfile for reads >= Q12 with NanoFilt (https://github.com/wdecoster/nanofilt) or other quality filtering software. Use that Q12 inputfile for Amplicon_sorter. (Lower quality reads can be used but will result in longer processing time and a lower percentage of reads that will assigned to a species.

Copy the Amplicon_sorter.py script in the same folder as your inputfile.

Command examples:

- Produce a read length histogram of your inputfile:
  ```
  python3 amplicon_sorter.py -i infile.fastq -o outputfolder -min 650 -max
  750 -ho
  ```
  : produce the readlength histogram of infile.fastq in folder outputfolder. This gives you the information on the number of reads between 650 and 750 bp.

```
avierstr@we11ca01: ~/maestri
File  Edit  View  Search  Terminal  Help
(base) avierstr@we11ca01:~/maestri$ python3 amplicon_sorter.py -i allq12.fastq
-o outputfolder -min 650 -max 750 -ho
Reading allq12.fastq
allq12.fastq contains 39643 reads.
Saved "allq12_total_outputfig.pdf" as a Read Length Histogram.
--> There are 39012 sequences between 650 and 750bp
(base) avierstr@we11ca01:~/maestri$
```

Read length histogram

Total yield (Gb): 0.03
Number of reads: 39,643
650 < bp < 750: 39,012
Max readlength: 0.9 Kb
N50 = 0.7 Kb
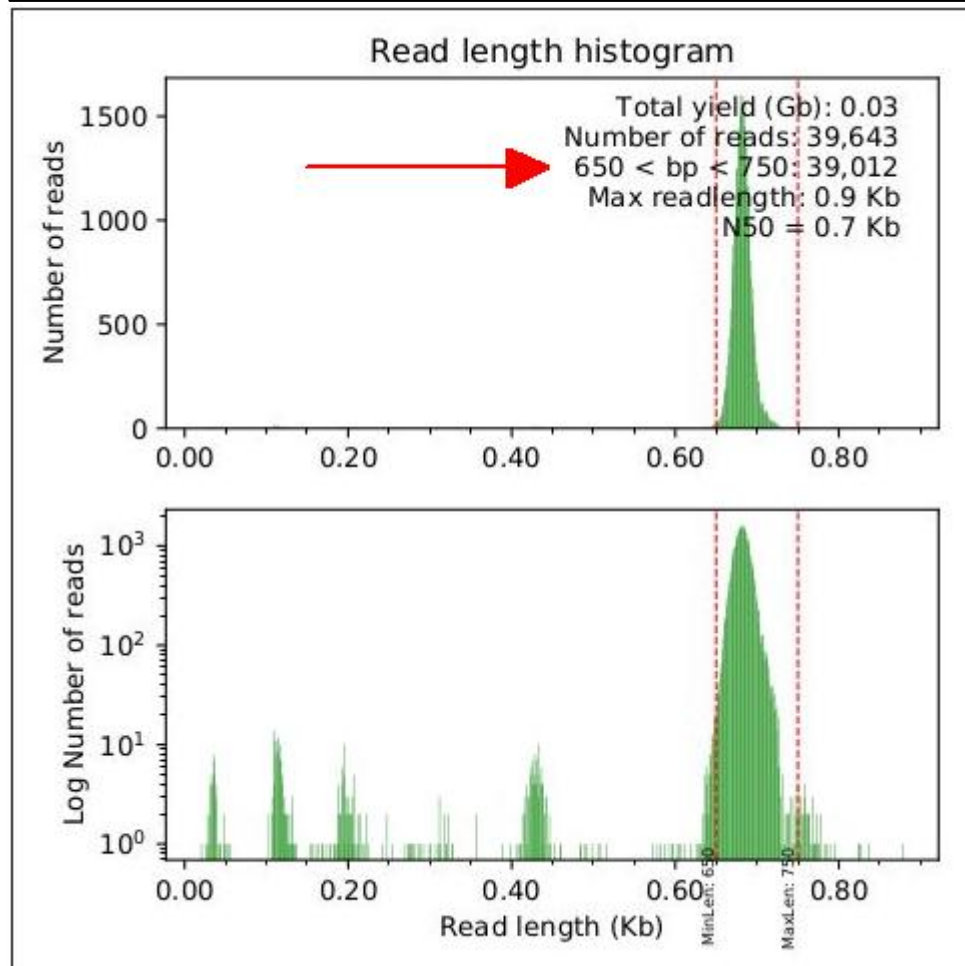
- Sample with one species amplicon of 750 bp:
  ```
  python3 amplicon_sorter.py -i infile.fastq -o outputfolder -np 8 -min 700
  -max 800 -maxr 1000
  ```
  process infile.fastq with default settings, save in folder outputfolder, run on 8 cores, minimum length of reads = 700, max length of reads = 800, use 1000 reads. This will sample the first 1000 reads between 700 and 800 bp of the inputfile. If you add the `-ra` (random) option to the command line, it will sample 1000 random reads between 700 and 800 bp.

- Sample with 2 species: an amplicon of 700 bp and one of 1200 bp:
  ```
  python3 amplicon_sorter.py -i infile.fastq -o outputfolder -np 8 -min 650
  -max 1250 -maxr 2000
  ```

- Metagenetic sample with several amplicons between 600 and 3000 bp, unknown number of species, 30000 reads in the inputfile:
  ```
  python3 amplicon_sorter.py -i infile.fastq -o outputfolder -np 8 -min 550
  -max 3050 -maxr 30000
  ```

- Metagenetic sample with several amplicons between 600 and 3000 bp, unknown number of species, 30000 reads in the inputfile, one low abundant species (< 2% reads):
  ```
  python3 amplicon_sorter.py -i infile.fastq -o outputfolder -np 8 -min 550
  -max 3050 -ra -maxr 600000
  ```
 By random sampling 20x the maximum number of reads, it is possible to find low abundant species.

Parameters depending on the basecaller and flow cell settings:

Guppy v5.xx has a High Accuracy (HAC) and Super Accuracy (SupHAC) option to do the basecalling and sequencing is possible on a 9.4.1 and R10 type of flow cell.

If you are working with species that are more than 95 – 96% similar, it is important to change or finetune some settings of Amplicon_sorter:

- `--similar_species_groups`: this is used to create species groups. The script is looking for the highest similarities between species and uses those to create species groups. When a better basecaller or flow cell is used, the higher this value can be. Below the histogram of similarities between reads from the HAC (left) and SupHAC (right) basecaller.
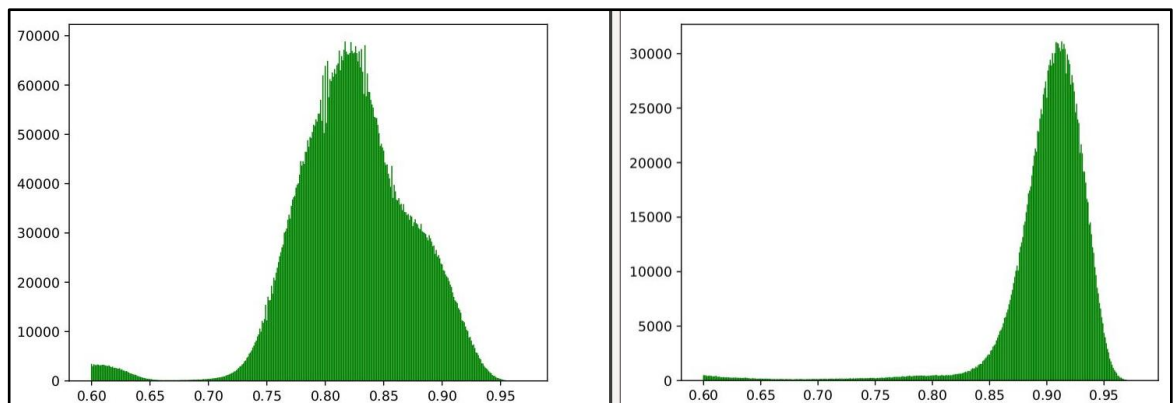

  ~~By default, the setting to create groups is 93% which is only a small portion of the reads with the HAC data, but a big part of the reads with the SupHAC data. If you increase that value for HAC data, you will find less species in the sample. If you decrease the value, it will result in more groups of the same species. For SupHAC and/or R10 data, it is better to increase the value to 94% (or more?).~~ From Amplicon_sorter version 2021-11-16 and later, when no value is entered (default), the script estimates this value based on the dataset.

- `--similar_consensus`: this parameter is used to merge species groups if the consensus is more than 96% (default for HAC) similar. When you increase this value, you will get more groups with the same species that are not merged. When decreasing this value, it is possible that closely related species are merged in one group. For the SupHAC and/or R10 data, this value can be increased to 98%.
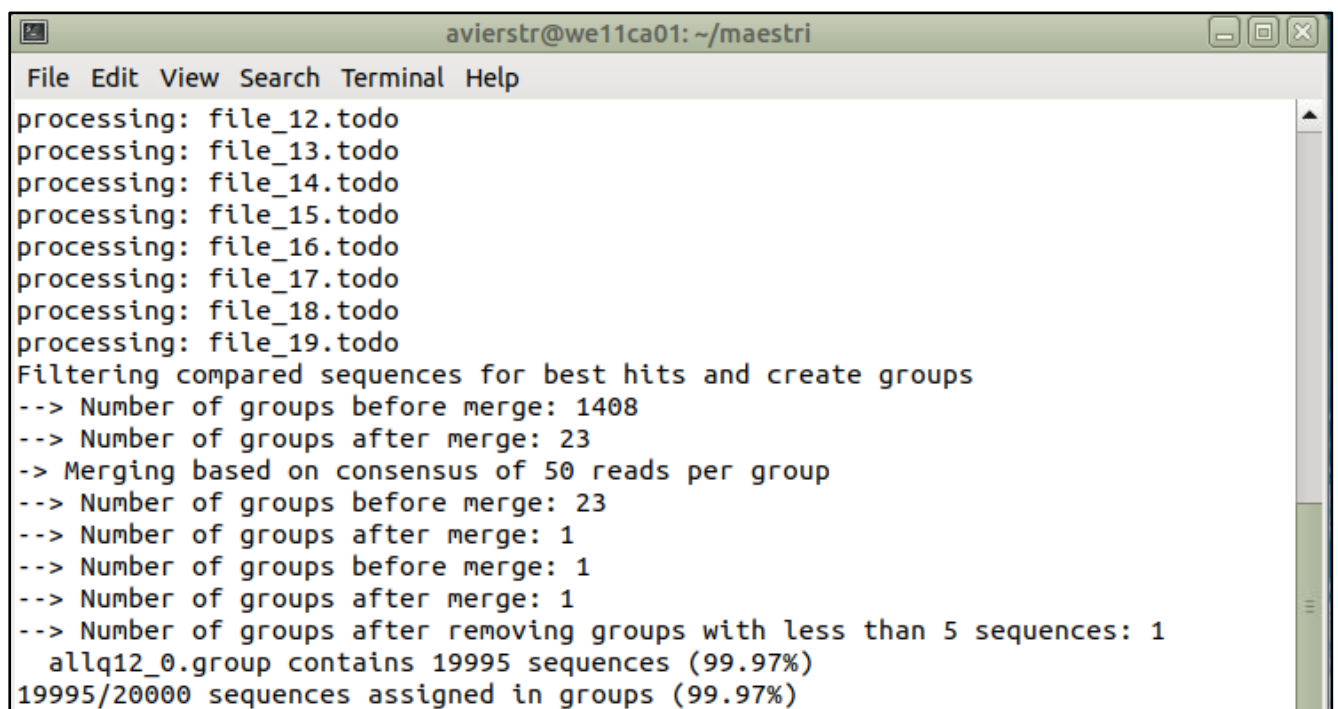
Processing a file:

1. <u>Comparing the reads:</u>
   Amplicon_sorter creates temporary files (file_xx.todo) with the sequences it must compare. It limits the number of files on disk to maximum 20 to save disk space. When files are processed, it removes the files from disk, and it is possible it adds new files to the queue depending on the number of comparisons it must do.
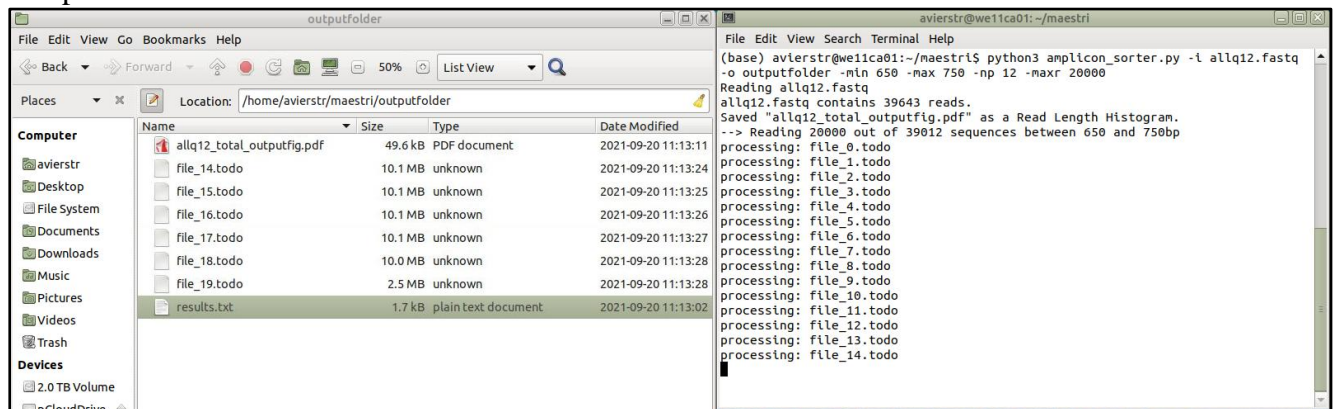
2. <u>Filtering out the results for best scores:</u>
   The script creates gene groups and merges groups based on the same sequences in the groups, and by comparing the consensus sequence of each group with each other. It removes groups that have less than 5 sequences. In this example there is only one group because the input file only contains COI sequences.


```
avierstr@we11ca01: ~/maestri
File  Edit  View  Search  Terminal  Help
processing: file_12.todo
processing: file_13.todo
processing: file_14.todo
processing: file_15.todo
processing: file_16.todo
processing: file_17.todo
processing: file_18.todo
processing: file_19.todo
Filtering compared sequences for best hits and create groups
--> Number of groups before merge: 1408
--> Number of groups after merge: 23
-> Merging based on consensus of 50 reads per group
--> Number of groups before merge: 23
--> Number of groups after merge: 1
--> Number of groups before merge: 1
--> Number of groups after merge: 1
--> Number of groups after removing groups with less than 5 sequences: 1
   allq12_0.group contains 19995 sequences (99.97%)
19995/20000 sequences assigned in groups (99.97%)
```

3. <u>Processing gene group files to species level:</u>
   Amplicon_sorter reads each gene group file and create species groups based on similarity. It merges groups based on sequences occurring in several species groups. It removes groups with less than 3

sequences. A consensus sequence is made for each group and the remaining sequences in the gene group are compared with those consensus sequences. This iterative process starts with a threshold of at least 95% similarity. When sequences are added to a group, a new and more accurate consensus is built to which the remaining sequences are compared. When no more sequences can be added (or a limit of 3 cycles for the same similarity), the similarity threshold is dropped by 1% and a new cycle starts. Every few cycles, the consensuses from all groups are compared and if the similarity is greater than or equal to 96%, the two species groups are merged. This iterative process ends at a similarity threshold of 85% and the sequences from each species group are saved in a file.

```
avierstr@we11ca01: ~/maestri

File  Edit  View  Search  Terminal  Help
reading allq12_0.group containing 19995 sequences.
--> Number of groups before merge: 120        Merging groups
--> Number of groups after merge: 91
--> Number of groups after removing groups with less than 3 sequences: 42
----> Making consensus for each group
allq12_0.group--> Comparing the rest of the sequences with the consensus sequenc
es
allq12_0.group----> similarity = 0.95     First cycle at 95%
allq12_0.group----> 799767 comparisons to calculate
...processing: file_0.todo
...processing: file_1.todo
allq12_0.group----> 1898 sequences added to the groups
allq12_0.group----> Making consensus for each group
allq12_0.group----> 772624 comparisons to calculate
...processing: file_0.todo
...processing: file_1.todo
allq12_0.group----> 164 sequences added to the groups
allq12_0.group----> Making consensus for each group
--> Number of groups before merge: 42
--> Number of groups after merge: 5
--> Number of groups before merge: 5        Merging groups
--> Number of groups after merge: 4
--> Number of groups before merge: 4
--> Number of groups after merge: 4
allq12_0.group----> 72695 comparisons to calculate
...processing: file_0.todo
allq12_0.group----> 16 sequences added to the groups
allq12_0.group----> Making consensus for each group
allq12_0.group----> similarity = 0.94     Cycle at 94%
allq12_0.group----> 1187 sequences added to the groups
allq12_0.group----> Making consensus for each group
--> Number of groups before merge: 4
--> Number of groups after merge: 4
allq12_0.group----> 67852 comparisons to calculate
...processing: file_0.todo
allq12_0.group----> 144 sequences added to the groups
allq12_0.group----> Making consensus for each group
--> Number of groups before merge: 4
--> Number of groups after merge: 4
allq12_0.group----> 67502 comparisons to calculate
...processing: file_0.todo
allq12_0.group----> 19 sequences added to the groups
allq12_0.group----> Making consensus for each group
```

...

```
allq12_0.group----> Making consensus for each group
allq12_0.group----> similarity = 0.85        end at 85%
allq12_0.group----> 24 sequences added to the groups
allq12_0.group----> Making consensus for each group
--> Number of groups before merge: 3
--> Number of groups after merge: 3
--> Number of groups before merge: 3
--> Number of groups after merge: 3
Writing sequences with high similarity in separate files  Writing files
amplicon_sorter version: 2021-09-19
---------------------------------------------------------
- date and time = September 20, 2021, 01:05PM
- input file = allq12.fastq
- output folder = /home/avierstr/maestri/outputfolder
- minlength = 650
- maxlength = 750
- maxreads = 20000
- n_processes = 12
- similar_genes = 80.0                    Settings used
- similar_species_groups = 93.0
- similar_species = 85.0
- similar_consensus = 96.0
- save_fastq = False
- random = False
- all_reads = False
- histogram_only = False
- species_only = True
---------------------------------------------------------
--> allq12_0_0.fasta contains 6541 sequences (32.71%)
--> allq12_0_1.fasta contains 11241 sequences (56.22%)   3 species files saved
--> allq12_0_2.fasta contains 1404 sequences (7.02%)
19186/19995 sequences assigned in groups for allq12_0.group (95.95%)
(base) avierstr@we11ca01:~/maestri$ []
```

4. <u>Files in the output folder:</u>

Name	Size	Type
allq12_0_0.fasta	4.5 MB	plain text document
allq12_0_1.fasta	7.8 MB	plain text document
allq12_0_2.fasta	970.8 kB	plain text document
allq12_0_consensussequences.fasta	2.2 kB	plain text document
allq12_0_unique.fasta	540.9 kB	plain text document
allq12_consensussequences.fasta	2.2 kB	plain text document
allq12_total_outputfig.pdf	49.6 kB	PDF document
results.txt	801 bytes	plain text document

- For each "species" and "gene" found, there is an outputfile: allq12_0_0.fasta contains the reads and consensus for species 1, allq12_0_1.fasta contains the reads and consensus for species 2,… In this example, there was only one gene in the inputfile. (With two genes, there would also have been allq12_1_0.fasta, allq12_1_1.fasta,… files.)
- allq12_0_consensussequences.fasta contains only the consensus sequences for the three species found for the first gene. (With a second gene, there would be a file allq12_1_consensussequences.fasta)
- allq12_0_unique.fasta contains the reads that could not be grouped with other.

- allq12_consensussequences.fasta contains all consensus sequences from all species and genes in one file. The consensus name refers to the fasta file with the reads and the number between brackets is the number of reads in the file.

- Allq12_total_outputfig.pdf is the read length histogram of the input file.
- Results.txt contains the settings used in the run and the % of reads assigned in groups.

Interpreting results:

When working with metagenetic samples and closely related species, it is possible that Amplicon_sorter is generating more than one species file from the same species. The one with the most reads in the file is normally the best one. When Blast is returning similarity values less than 97%, the result is mostly unreliable because Amplicon_sorter grouped some reads with lower similarity/quality of a certain species.

In the example below, Blast finds 5 times the same species Gomphus pulchellus. For line 12 and 17 it has only 4 and 6 reads for the consensus, so those can be ignored.

Line 11 has 314 reads for the consensus for which the individual reads have a similarity with a Gomphus sp. ranging between 80 and 89% (these are reads with a quality > Q12 but low similarity, let call this "sloppy basecalled reads"). Amplicon_sorter probably found a few of those "sloppy basecalled reads" with high similarity with each other, formed a group and added other "sloppy basecalled reads" to it, so the consensus is inaccurate.

Line 15 and 24 have a consensus out of 1152 and 1491 reads. The individual reads have a similarity ranging from 85 to 94% for line 15 and 89 to 99% for line 24 with G. pulchellus (which is much higher than the "sloppy basecalled reads" from line 11). So, the most reliable consensus is the one from line 24.

	species	gene	% iden	length	sequence name (# seq)
1	species	gene	% iden	length	sequence name (# seq)
2	unknown		-	1130	consensus_pieter_q12_trim_4_0(1330)
3	unknown		-	1670	consensus_pieter_q12_trim_5_0(8208)
4	unknown		-	960	consensus_pieter_q12_trim_8_0(1011)
5	Pt3_Pleuroxus_truncates_COI	COI	100.000	686	consensus_pieter_q12_trim_1_0(675)
6	Gomphus_schneiderii_11521_COI	COI	100.000	708	consensus_pieter_q12_trim_0_8(3415)
7	Hb2_COI	COI	100.000	735	consensus_pieter_q12_trim_0_0(5883)
8	Cordulegaster_bidentata_11568_COI	COI	100.000	735	consensus_pieter_q12_trim_0_2(2792)
9	Cordulegaster_amasina_mzymtae_11279_COI	COI	100.000	735	consensus_pieter_q12_trim_0_9(1665)
10	Pt3_Pleuroxus_truncates_COI	COI	83.562	734	consensus_pieter_q12_trim_0_14(208)
11	Gomphus_pulchellus_11849_COI	COI	84.709	740	consensus_pieter_q12_trim_0_18(314)
12	Gomphus_pulchellus_11831_COI	COI	87.678	727	consensus_pieter_q12_trim_0_16(4)
13	Gomphus_lucasii_9429_COI	COI	89.833	739	consensus_pieter_q12_trim_0_15(453)
14	Gomphus_lucasii_9429_COI	COI	91.896	717	consensus_pieter_q12_trim_0_19(307)
15	Gomphus_pulchellus_11831_COI	COI	92.542	732	consensus_pieter_q12_trim_0_6(1152)
16	Gomphus_maroccanus_9246_COI	COI	95.427	727	consensus_pieter_q12_trim_0_13(1252)
17	Gomphus_pulchellus_11849_COI	COI	97.415	739	consensus_pieter_q12_trim_0_17(6)
18	Cordulegaster_buchholzi_COI	COI	98.239	735	consensus_pieter_q12_trim_0_7(1681)
19	Gomphus_lucasii_10720_COI	COI	99.316	704	consensus_pieter_q12_trim_0_10(2533)
20	Cordulegaster_insignis_11567_COI	COI	99.649	734	consensus_pieter_q12_trim_0_4(3039)
21	Gomphus_vulgatissimus_11522_COI	COI	99.696	734	consensus_pieter_q12_trim_0_12(3160)
22	Ag8_COI	COI	99.725	733	consensus_pieter_q12_trim_0_3(3194)
23	Ss9_COI	COI	99.728	735	consensus_pieter_q12_trim_0_1(6546)
24	Gomphus_pulchellus_11849_COI	COI	99.848	734	consensus_pieter_q12_trim_0_11(1491)
25	Gomphus_kinzelbachi_Iran_10588_COI	COI	99.848	735	consensus_pieter_q12_trim_0_5(1936)

When comparing the consensus from line 15 and line 24 (see below), there are 7% differences which is the reason why Amplicon_sorter is not merging those two groups (`--similar_consensus`: is 96% by default)

Todo:

- Try to improve speed for comparison of sequences.
- Try to fine tune sorting to species level. If there are species in the sample that are more than 95-96% similar, the script has difficulties to separate them. Species with a higher similarity are often grouped together and give a lower consensus sequence because it is the "average" of 2 or more closely related species.
- Check for bug: there is sometimes an error in the percentage of reads assigned per group (sometimes > 100%). This has no effect on the sorting or consensus made, only on the information how many reads are assigned.

Release notes:

2023/03/12:

- Added option to use all reads `-ar, --allreads`. This option is still limited by the `-maxr', '--maxreads` to have a hard limit.
- Added option `-ldc', '--length_diff_consensus`: Length difference (in %) allowed between consensuses to COMBINE groups based on the consensus sequence. This can be interesting if you have amplicons of different length, the shorter ones are nested sequences of the longer ones and you want to combine those in one group.
- catch "hang of amplicon_sorter" if only one read is present in the inputfile. Now if less than 5 reads are present in the inputfile, the program exits.

2022/03/28:

- a little speed improvement (1,2 x) by using another consensus calling method (edlib). Levenshtein plugin no longer needed.
- added a "finetune cycle" to remove reads that are too different from the consensus. This results sometimes in a better consensus, can sometimes separate species with a high similarity, removes groups with a low number of reads.

2021/12/24:

(version of the publication of March 2022 (https://doi.org/10.1002/ece3.8603)

- limit number of comparisons (for large data files).
- checks to cleanup temporary files.
- limit number of comparisons to merge files.
- memory improvement to process large data files. Still needs more improvement.

2021/12/19:

- minor changes to increase speed a little bit.
- filtering of the reads to species groups changed from list to dict format: speed increase and more memory efficient.
- parallelizing comparing consensus and making consensus to get speed increase for larger datasets
- merging groups changed from list to dict format: speed increase and more memory efficient.

2021/12/01:

- major speed improvement by using the edlib library to do the comparison between sequences (see requirements for installation) (first tests between 5 - 10 x faster).
- minor speed increase in merging groups (4 x).

- bugfix when non existing path was given as output folder. Now a multilevel path can be given as input and output.
- creation of read length histogram `-ho, --histogram_only` is now optional, no longer default.

2021/11/16:

- fixed rarely occurring bug
- program checks if a newer version of Amplicon_sorter is available.
- fixed bug when entering a "path/infile.fastq" as input. The program crashed halfway with an error. Now it is possible to use a path as input.
- changed the default for --similar_species_groups to "estimate": the programs estimates the value from the dataset instead of the default value of 0.93.

2021/09/21:

- fixed bug that is important for sorting closely related species. (Was wrongfully removed in previous version)

2021/09/11:

- speed and memory improvement when sorting the compared reads for best matches
- added option `--similar_consensus`
- added option `--all`
- save all parameters in results.txt to know afterwards with which parameters a run is performed.

2021/08/08:

- speed improvement by changing allowed difference in length from 1.1 to 1.05 (less comparisons to be done) and 1.08 for comparison of consensus sequences
- change default of `--similar_species_groups` from 0.92 to 0.93

2021/07/13:

- speed improvement by changing allowed difference in length from 1.3 to 1.1 (less comparisons to be done)
- little changes that improve speed a bit

2021/05/28:

- improvement on combining groups
- small speed improvement when creating consensus to compare groups

2021/05/19:

- small speed improvement when comparing remaining sequences with consensus in groups

2021/05/13:

- chunk number of comparisons in groups of 500K to save memory and process parallel
- minor bug fixes and small speed improvements
- improvements in number of groups created

2021/05/06:

- If no max number of reads is given, use all reads

- write version of script in results.txt
- write number of reads in length selection to readlength histogram
- combine groups if consensus is at least 99% similar
- save time by making consensus of max 500 random reads in group
- major speed improvement when comparing a lot of sequences. When the number of sequences is x 100, calculation time is x 10.000. Subsampling in batches of 1000 sequences decreases calculation time x 10 (it does NOT compare al sequences with each other, it compares batches of 1000 with each other)

2020/5/20:

- Save a file "results.txt" in outputfolder with how many reads are in which file.
- Catching error when there are less reads available than asked in `--maxreads`.

2020/5/6:

- Little cosmetic change in read length histogram.
- Corrected a minor bug with (`-ho --histogram_only`) option.

2020/4/27:

- Little change in step 3 that improves the processing speed.
- Made names of temporary files unique in case you want to process different files in one folder.

2020/4/17:

- Added vertical lines to the read length histogram with the minimum and maximum read length that were given as read length selections.
- Option to create a read length histogram only (`--histogram_only`).
- Option to change the similarity when CREATING species groups (`--similar_species_groups`).
- Option (`--species_only`) to play with the `--similar_species` and `--similar_species_groups` parameters without having to start all over.

2020/4/3:

- Option to save results in a subfolder (`-o --outputfolder`).
- More little improvement to sort out the groups in step 2. (The result is still not what I had in mind: sometimes reads of ITS and 18S are in the same group. The reason for this is that ITS can be very different from species to species. If I set similarity to 55%, it can put some ITS in the 18S group. If I set it higher, than COI is not in one group. This has no influence on the end results in step 3-4).

2020/3/12:

- Option to take a random selection of the reads from the inputfile (`--random`).
- Little improvement to sort out the groups in step 2.

2020/3/5:

- Fasta or fastq files possible as input (autodetect).
- Fastq files as output option (when input is fastq) (`--save_fastq`).