Tutorial: UPARSE OTU generation with strategies to fix memory cap problems in usearch

Umer Zeeshan Ijaz

Say we have the following folder structure: We have a "main" folder, and within it we have several subfolders (use meaningful names as the folder names appear in the final OTU table) each with overlapped paired-end reads (*_R12.fasta obtained from pear/pandaseq/AMPLICONprocessing etc.). They can also be individual paired-end files.

Folder structure:

```
~/main/1/1_R12.fasta
~/main/2/2_R12.fasta
~/main/3/3_R12.fasta
~/main/4/4_R12.fasta
```

Step #1: We need to combine all the overlapped sequences (*_R12.fasta) by generating multiplexed.fasta. Final labels are in usearch format: barcodelabel=<folder_name>;S<ids>. The sequences in each sample are given internal identifiers starting with S1, S2, and so on.

```
for i in $(ls -d ~/main/*/); do awk -v k=$(basename ${i})
'/^>/{$0=">barcodelabel="k";S"(++i)}1' < $i/*_R12.fasta;
done > multiplexed.fasta
```

Step #2: Linearize multiplexed.fasta

```
awk 'NR==1 {print ; next} {printf /^>/ ? "\n"$0"\n" : $1}
END {print}' multiplexed.fasta >
multiplexed_linearized.fasta
```

Step #3: Dereplicate the sequences (32-bit version of usearch will fail as there is a 4GB memory cutoff). Let us dereplicate using a work-around and produce the output in usearch format. We assume there are no undetermined bases (N) in our overlapped sequences (you can specify this in pandaseq through a parameter)

```
grep -v "^>" multiplexed_linearized.fasta | grep -v
[^ACGTacgt] | sort -d | uniq -c | while read abundance
sequence ; do hash=$(printf "${sequence}" | sha1sum);
hash=${hash:0:40};printf ">%s;size=%d;\n%s\n" "${hash}"
"${abundance}" "${sequence}"; done
>  multiplexed_linearized_dereplicated.fasta
```

Step #4: Abundance sort and discard singletons

```
./usearch7.0.1001_i86linux32 -sortbysize
multiplexed_linearized_dereplicated.fasta -output
multiplexed_linearized_dereplicated_sorted.fasta -minsize
2
```

Step #5: OTU clustering

```
./usearch7.0.1001_i86linux32 -cluster_otus
multiplexed_linearized_dereplicated_sorted.fasta -otus
otus1.fa
```

Step #6: Chimera filtering using reference database. The -cluster_otus command in usearch discards reads that have chimeric models built from more abundant reads. However, a few chimeras may be missed, especially if they have parents that are absent from the reads or are present with very low abundance. It is therefore recommend to use a reference-based chimera filtering step using UCHIME if a suitable database is available. Use the -uchime_ref command for this step with the OTU representative sequences as input and the -nonchimeras option to get a chimera-filtered set of OTU sequences. For the 16S gene, Robert Edgar recommends the gold database (http://drive5.com/uchime/gold.fa) and not using a large 16S database like Greengenes.

```
./usearch7.0.1001_i86linux32 -uchime_ref otus1.fa -db
gold.fa -strand plus -nonchimeras otus2.fa
```

Step #7: Label OTU sequences OTU_1, OTU_2,…We will use Robert Edgar's python script (http://drive5.com/python/python_scripts.tar.gz)

```
python fasta_number.py otus2.fa OTU_ > otus.fa
```

Step #8: Map reads (including singletons) back to OTUs

```
./usearch7.0.1001_i86linux32 -usearch_global
multiplexed_linearized.fasta -db otus.fa -strand plus -id
0.97 -uc map.uc
```

Step #8 (Alternative Strategy): There is a 3GB limit on free version of usearch and if the above command fails, then we have to split the original file into small chunks. To do so, make two folders and download FASTA file splitter from http://kirill-kryukov.com/study/tools/fasta-splitter/

```
mkdir split_files
mkdir uc_files
```

In split_files, break the FASTA files into 100 equal parts:

```
perl ../fasta_splitter.pl -n-parts-total 100
../multiplexed_linearized.fasta
```

You will get the following files:

```
ls split_files
multiplexed_linearized.part-001.fasta
multiplexed_linearized.part-002.fasta
multiplexed_linearized.part-003.fasta
multiplexed_linearized.part-004.fasta
multiplexed_linearized.part-005.fasta
multiplexed_linearized.part-006.fasta
multiplexed_linearized.part-007.fasta
multiplexed_linearized.part-008.fasta
multiplexed_linearized.part-009.fasta
multiplexed_linearized.part-010.fasta
...
```

Now run the following command in split_files folder:

```
for i in $(ls *.fasta); do ../usearch7.0.1001_i86linux32
-usearch_global $i -db ../otus.fa -strand plus -id 0.97 -
uc ../uc_files/$i.map.uc; done

ls ../uc_files
multiplexed_linearized.part-001.fasta.map.uc
multiplexed_linearized.part-002.fasta.map.uc
multiplexed_linearized.part-003.fasta.map.uc
multiplexed_linearized.part-004.fasta.map.uc
multiplexed_linearized.part-005.fasta.map.uc
multiplexed_linearized.part-006.fasta.map.uc
multiplexed_linearized.part-007.fasta.map.uc
multiplexed_linearized.part-008.fasta.map.uc
multiplexed_linearized.part-009.fasta.map.uc
multiplexed_linearized.part-010.fasta.map.uc
....
```

And combine all the map.uc files together

```
cd ..; cat uc_files/* > map.uc
```

Step #9: Generate tab-delimited OTU table using Robert Edgar's python script

```
python uc2otutab.py map.uc > otu_table.txt
```

Step #10: Convert tab-delimited OTU table to csv file

Quote everything:

```
perl -lpe 's/"/""/g; s/^|$/"/g; s/\t/","/g' <
otu_table.txt > otu_table.csv
```

Simpler solution:

```
tr "\\t" "," < otu_table.txt > otu_table.csv
```

Final files:

otu_table.csv ←Frequency file
otu.fa ←OTU sequences