Trim ONT adapters through MOTUs

Oliver Stringham

Software Used

Concat fastqs into one fastq	command line
Quality filter	NanoFilt
Length Filter	seqkit
Primer match and trim	cutadapt
Trim extra ONT adapters	porechop
Cluster sequences into MOTUs	VSEARCH
BLAST	BLAST

Other helper softwares: bbmap, R, command line

https://github.com/rrwick/Porechop

- **Finds and removes ONT adapters**
  - **Ligation kit adapters; Rapid kit adapters; PCR kit adapters; Barcodes; Native barcoding; Rapid barcoding; potential custom adapters too**
  - **https://github.com/rrwick/Porechop/blob/master/porechop/adapters.py**
- **Why use it? 2 reasons**
  - **Should we run on linked, unlinked reads, or both?**
- **Uses a lot of computation (CPU)**
- **Unsupported since 2018 🤷‍♀️**

Porechop con’t

- Since it’s computational expensive:
  - Let’s subset the unlinked reads, removing any linked reads that are in the unlinked fastq

- Porechop algrorithm for finding adapters
  - Means we need to ‘shuffle’ reads first

Porechop con’t

Porechop parameters

--threads	number of cpu cores to use
--discard_middle	if used, porechop removes reads with adapters in the middle of the read
-i	input file

Porechop con’t

```
singularity exec images/porechop.sif \
porechop --threads 8 --discard_middle \
-i input.fastas > \
output.fasta
```

Clustering

- Reads --> MOTUs
- MOTU (Molecular Operational Taxonomic Unit) ~ a group of reads that share a certain level of similarity
- We have to define the level of similiarity e.g. 98%
- Why do this?
- many different ways to cluster, we'll use VSEARCH

Clustering con’t

VSEARCH Clustering algorithm

(or at least my understand of it)

- Sort reads by length, longest being first
- Read 1 becomes cluster 1's “centroid”
- Check if read 2 is >98% similar* to cluster centroid of cluster 1,
  - If yes, it gets grouped into cluster 1,
  - if not becomes the centroid of cluster 2
- Repeat last step, where each read gets compared to all centroids until it finds one it matches and if doesn’t match any it becomes it’s own centroid.
- Once all reads are sorted. VSEARCH aligns all reads within each cluster to make a ‘**consensus**’ sequence for each cluster. (ie chooses most common base at each position)

*using k-mer based matching

VSEARCH Parameters

--cluster_fast	specifies length based clustering
--threads	# of cpu cores to use
--id	proportion similiarity to use (ie 0.98)
--centroids	output location for centroids reads .fasta
--consout	output location for consensus reads .fasta
--uc	output location for the key for what reads are in each cluster (.uc extension)

Run VSEARCH

```
singularity exec images/vsearch.sif \
vsearch \
--cluster_fast \
--threads 4 \
--id 0.98 \
input.fasta \
--centroids output.cen.fasta \
--consout output.con.fasta \
--uc output.clusters.uc
```

Clustering con’t

- Now we’re left with the consensus seq of each cluster
- Last thing to do is to remove clusters with 5 or less reads
  - why?
- Also need to tabulate

BLAST is next