The Microbializer pipeline aims to process bacterial genomes and analyze them. The first part finds orthologous sets across all given species. The flow for finding orthologous sets will be handled by a wrapper that will run different modules one by one. The input for the pipeline is a path to a directory that contains genome fasta files.

1. extract\_orfs.py

**Input**: (1) an input path for a fasta file with contigs/full genome (2) an output file path (with a suffix as follows: i\_genes.fasta. especially relevant for the wrapper).

**Output**: a fasta file where under each header, there’s a single gene.

1. blast\_all\_vs\_all.py

**Input**: (1) 2 input paths for 2 (different) genes files, g1 and g2 (2) an output file path (with a suffix as follows: i\_vs\_j\_blast.tsv. especially relevant for the wrapper).

**Output**: a tsv file where the first column contains g1 genes and the second column includes the corresponding best match.

Precisely, for each gene x in g1, blast x among all the genes of g2. Let y be the gene in g2 that is the most similar to x among all g2 genes. Append a row to the output file with: ‘{x}\t{y}’.

1. filter\_blast.py

**Input**: (1) a path for a i\_vs\_j\_blast.tsv file (2) an output path (with a suffix as follows: i\_vs\_j\_filtered.tsv. especially relevant for the wrapper).

**Output**: the same format of the input file containing only pairs that passed the filtration. For each row in the input file (pair of genes), apply the following filters:

1. at least X% similarity

2. at least X% of the length

3.

write each pair to the output file if it passed all the above filters.

1. find\_reciprocal\_hits.py

**Input**: (1) a path for a i\_vs\_j\_filtered.tsv file and a path for a j\_vs\_i\_filtered.tsv file (2) an output path (with a suffix as follows: i\_vs\_j\_reciprocal\_hits.tsv

**Output**: a tab delimited file containing only reciprocal best-hit pairs and their bit score from blast, i.e., if x’s best hit was y in the first file with bit score z, x\ty\tz will appear in the output file only if y’s best hit in the other file will be x.

1. construct\_putative\_orthologs\_table.py

**Input**: (1) a path for a i\_vs\_j\_reciprocal\_hits.tsv file (2) a path for a putative orthologs file (with a single line).

**Output**: updates the table with the info from the reciprocal hit file.

1. split\_putative\_orthologs\_table.py

**Input**: (1) a path for a putative orthologs table (each row in this table is a putative orthologous set) (2) an output path to a directory.

**Output**: each line is written to a separate file in the output directory.

1. verify\_cluster.py

**Input**: (1) a file path (containing a row from the putative orthologs table) (2) an output path to a directory.

**Output**: copies this file to the output folder if the group is well-clustered.

1. Join\_final\_orthologs\_table.py

**Input**: (1) a path for directory with all the verified OGs (2) an output path to a final OGs table.

**Output**: aggregates all the well-clustered OGs to the final table.

1. extract\_sequences.py

**Input**: (1) a row from the final orthologs table (2) a path for a directory where the genes files are at (3) a path for an output file.

**Output**: write the sequences of the orthologs group to the output file.