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:

1. blast\_all\_vs\_all.py

The input for this module are 2 paths for 2 (different) bacterial genomes, g1 and g2 and an output file path.

The output will be 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. **Split and QA reads**

Given a fastq file (reader output with all sequences) the first step is QA and splitting the data into the different samples. This is done using the script: **/groups/pupko/orenavr2/gershoni/src/FilterReads.py**

**Inputs:**

* <fastq\_file> (all the reads)
* <out\_dir> the base directory where sub directory for the files of each sample will be created
* <barcode\_to\_name (tab delimited)>
* <code\_to\_barcode (tab delimited)>
* <mistakes\_allowed> number of mismatch allowed within the ‘constant’ (fth1\_anealed\_Site) region. Default=1

**Outputs:**

For each analyzed sample defined in the barcodes list, create a subdirectory inside the *outDir* with name equals to the descriptive name defined in the *Barcode\_to\_ExpName*. Each such sample subdirectory will contain the following files:

1. List of filtered reads for the given sample
2. List of sequences (barcode region) passed the QA (Fasta format)
3. Info file

**Pipeline methodology:**

The expected structure of each read is as follow:

* Sample barcode (5 Nt)
* fth1 annealed Site sequence (16 Nt) = aagtaggggatccagg
* Domain barcode (12 Nt) 🡪 This is the region need to be extracted
* fth1 annealed - anti sense (17 Nt) = tctagagccgaccgcga

The script performs the following:

1. Does the read start with any of the defined samples barcode. If Yes, continue to check this sequence. Otherwise skip the sequence and count this sequence as skipped.
2. Count this sequence as belongs to the sample barcode
3. Extract the ‘fth1 annealed Site sequence’ and check if it matches the expected sequence. If less than ‘*MistakeAllowed’*, continue and save the number of mismatches. Otherwise skip the sequence and:
   1. Report it in the sample filtered reads file
   2. Count it as filtered
4. Extract the ‘fth1 annealed - anti sense’ region and check if it is **equal (no mismatches)** tothe expected sequence. If it is valid sequence, continue. Otherwise skip the sequence and count and report as filtered.

At end we create for each sample (FASTA) file also a unique sequences Fasta file. This is done using the script: /groups/pupko/haim/pupkoSVN/trunk/programs/DeepPanning/SeqAnalysis/countUniqSeq\_Fasta.pl called from the main pipeline script described above. This script takes a Fasta file with sequences and output a unique sequences Fasta file. The header of each unique sequence includes the total copy number of that sequence in the non-unique input file. The unique output file is sorted by the total copy number of each sequence. For example if the sequence GGTTGTGGCAAA appeared 222,272 times in the non-unique input file it will appear only once in the unique output file as follow:

>1\_Length\_12\_Repeats\_222272

GGTTGTGGCAAA

The output is saved to a file named \*.UNIQUE.NORMALIZATION\_NONE.fs

**Run example:**

perl /groups/pupko/haim/pupkoSVN/trunk/programs/DeepPanning/SeqAnalysis/FilterReads\_fth1\_DS\_NCBC\_P8.clean\_version.pl --fastq\_file /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30.For\_DS\_Tests\_barcodes\_GGATC\_GGCTA\_GGTCA.fastq --barcodes "GGATC,GGCTA,GGTCA" --outDir /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/ --Barcode\_to\_ExpName /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/DS\_Samples.txt

1. **Split the QA domains barcodes to those related to the designed experiment and to ‘left-overs’.**

**The inputs for that script are:**

* 1. The unique FASTA file with domains sequences resulted from stage 1.
  2. List of domain barcodes of interest [example: /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/DS50\_HIV.txt]
  3. List of samples scanned with the list of domains barcode of interest (b)

**The outputs of that script are:**

1. csv file with the list of domain barcodes of interest as columns (same order as provided in the input file) and the number of reads for each domain in each sample. Each sample is a row. [example: /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30\_DS\_Samples\_With\_DS150\_HIV\_CopyNumber.csv]
2. Like (a) but with frequencies instead of copy number. [example: /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30\_DS\_Samples\_With\_DS150\_HIV\_percent.csv]
3. File with domains ‘left overs’ – for each sample, create a txt file including all the domains barcode passed QA which are not part of the ‘domains of interest’ list [example: /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/8188\_50\_b12\_1\_DS50\_HIV/8188\_Exp30.For\_DS\_Tests\_barcodes\_GGATC\_GGCTA\_GGTCA.50\_b12\_1\_DS50\_HIV.MistakeAllowed.1.NUC.UNIQUE.NORMALIZATION\_NONE.with\_DS50\_HIV.leftovers.txt]
4. For each sample in the list provided, indicate the percent of peptides belongs to ‘the domains of interest’ and ‘left-overs’ [example: /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30\_DS\_HIV50\_real\_vs\_leftovers.csv]

I do it in few steps:

**# prepare 1 line txt files for given set of domains and FASTA file as an input**

foreach name (`ls -ltr /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/| grep "^d" | awk '{print $9}' | grep DS50\_HIV`)

set x=`ls -1 /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/$name/\*.MistakeAllowed.1.NUC.UNIQUE.NORMALIZATION\_NONE.fs`

perl /groups/pupko/haim/Projects/Gershoni/fasta\_to\_tab\_delim\_selected\_domains.pl /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/DS50\_HIV.txt $x

echo $x

end

**# unite all data DS50\_HIV**

foreach name (`ls -ltr /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/| grep "^d" | awk '{print $9}' | grep DS50\_HIV `)

ls -1 /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/$name/\*.MistakeAllowed.1.NUC.UNIQUE.NORMALIZATION\_NONE.with\_DS50\_HIV.txt >> /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/List\_OF\_Exp30\_DS\_Samples\_With\_DS50\_HIV.txt

end

perl pupkoSVN/trunk/programs/DeepPanning/SeqAnalysis/Join\_DS\_samples.pl /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/List\_OF\_Exp30\_DS\_Samples\_With\_DS50\_HIV.txt /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30\_DS\_Samples\_With\_DS150\_HIV

**# compare real reads to left-overs**

foreach name (`awk '{print $2}' /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/DS\_Samples.txt`)

set real=`ls -1 /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/\*$name/\*.MistakeAllowed.1.NUC.UNIQUE.NORMALIZATION\_NONE.with\_DS50\_HIV.txt`

set left=`ls -1 /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/\*$name/\*.MistakeAllowed.1.NUC.UNIQUE.NORMALIZATION\_NONE.with\_DS50\_HIV.leftovers.txt `

perl /groups/pupko/haim/Projects/Gershoni/Compare\_DS\_real\_with\_leftovers.pl $real $left >> /groups/pupko/haim/Projects/Gershoni/test\_DomainScan\_Pipeline/Exp30\_DS\_HIV50\_real\_vs\_leftovers.csv

end