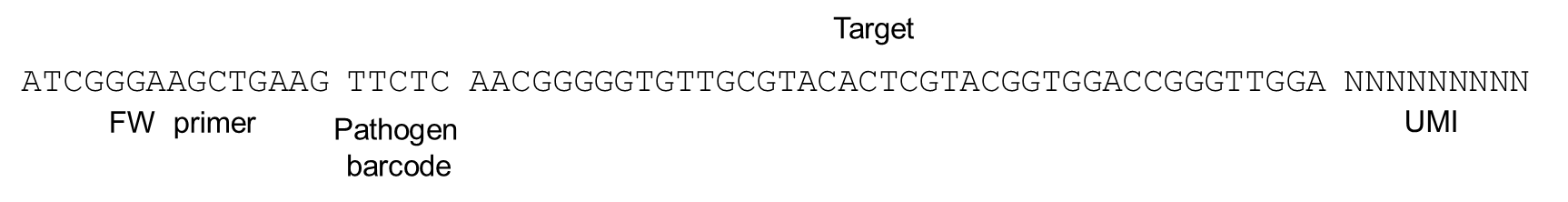
**Read description**

The reads are composed of 4 main parts. The FW primer sequence, followed by a unique barcode for each pathogen. Thereafter these is the target sequence, there are more than one target regions/Pathogen. Finally there is a 9 bp UMI that allows me to count No. of reads/target. Depending on the lenght of the target there are some bases of the RV primer (TGTGATGGCAGCCTA).

My sequencing runs are usually 1X75 single reads, which means my reads can’t be longer than 75 bases.



**Fig 1**. Amplicon/read structure example

**Input files**

I start with \*.gz files, which I decompresed first **manually** to have \*.fastaq files. For each sample I have four \*.gz files, for instance:

Sample-36\_S36\_L001\_R1\_001.fastq

Sample-36\_S36\_L002\_R1\_001.fastq

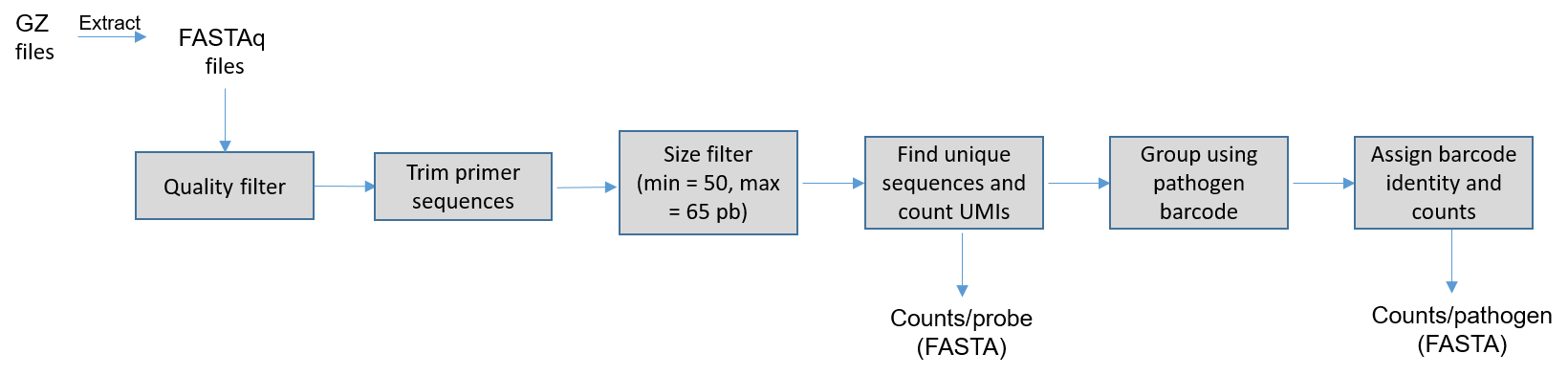
Sample-36\_S36\_L003\_R1\_001.fastq

Sample-36\_S36\_L004\_R1\_001.fastq

**Batch file pipeline:**

I have put together a batch file (windows) that contains a sequence of pearl scripts from (<https://sourceforge.net/projects/ngs-toolbox/>). The main objectives are to be able to count the number of unique sequences per target(probe), given by the UMIs and to be able to report the number of UMIs per Pathogen barcode (Fig 2.).

I usually have 96 samples but I should be able to adjust this for the number of samples each time.



**Fig 2.** Overall view of the pipeline.

The batch file called ”PLP\_findv5.bat” reads as follows. I use Perl in a command line in windows 10 to run it.

perl q\_filter.pl -I inputfilelist.txt

perl FASTQ\_to\_FASTA.pl -i passed\_filter.fastq

perl remove\_TAGs.pl -i passed\_filter.fas -T adapterFW.txt

perl length\_cutoff.pl -i outfile\_ATCGGGAAGCTGAAG.fas -min 28 -max 65

perl discard\_redundant\_sequences.pl -i sequences\_ok.fas

perl remove\_TAGs.pl -i nonredundant.fas -T Target\_List.txt -5

perl discard\_redundant\_sequences.pl -I uniqueseqs.txt - o umis\_count.txt

perl sort\_by\_TAGs.pl -i umis\_count.txt -T Path\_TAGs.fas -1

perl merge\_FASTA.pl -I outfile\_TAG\_sort.txt - o pathcounts.fas

call del\_TAGs\_outfiles.bat

call del\_empty.bat

md Sample\_

In detail, it performs the following operations:

1. perl q\_filter.pl -I inputfilelist.txt # calls the perl script that makes a quality filter of my reads. Uses the file inputfilelist.text Which contains a list of the input files/sample and I **manually** create for each sample. The output is the file passed\_filter.fastq.

(I have tried this python script also to perform this filter, but it gives the same/similar results

https://gist.github.com/tjanez/d23e20c1a777a222fd7d)

1. perl FASTQ\_to\_FASTA.pl -i passed\_filter.fastq # calls the perl script that converts the \*.fastaq files into \*.fasta. Uses the file passed\_filter.fastq generated in step 1. The outfile is simply the file passed\_filter.fas.
2. perl remove\_TAGs.pl -i passed\_filter.fas -T adapterFW.txt # calls the perl script that finds a sequence in the reads and remove it and everything before. I use it to find the FW primer sequence and remove it from all the reads. Input files are passed\_filter.fas and adapterFW.txt that contains the FW primer sequence.

The output are two files: outfile\_ATCGGGAAGCTGAAG.fas and outfile\_NO\_TAG.fas

1. perl length\_cutoff.pl -i outfile\_ATCGGGAAGCTGAAG.fas -min 28 -max 65 # calls the perl script thar filter the reads by length (minimum 28, max 65). It uses the output file outfile\_ATCGGGAAGCTGAAG.fas from the previous script as an input.

The output are three files called sequences\_ok.fas, sequences\_too\_long.fas and sequences\_too\_short. fas

1. perl discard\_redundant\_sequences.pl -i sequences\_ok.fas # this one calls the perl script that find redundant sequences and count them. I use this one to count number of reads/UMIs. Uses the sequences\_ok.fas as an input.

The output is a file called nonredundant.fas where the reads are listed as follows:

>6

AACTGGAATTTCACGTGCTCCGTCGTACTCAGGATCCACTCAAGAGAGACAACATTCTCAT

(...)

Where the number after ”>” corresponds to the number of unique reads for that sequence.

1. perl remove\_TAGs.pl -i nonredundant.fas -T Target\_List.txt -5 # I call again this script but with a different argument, so I can use it to find the sequences of my targets. It also deletes the UMIs from the reads to be able to do the next step.

As input uses the nonredundant.fas file from the previous script and my library of targets Target\_List.txt. I use the argument -5 so when it finds the sequences it deletes the UMIs.

It generates a number of outputfiles for each target in the list. For instance:

outifile\_AACTGGAATTTCACGTGCTCCGTCGTACTCAGGATCCACTCAAGAGAGACAACATTCTCAT.fas

(...)

This output files contain all the reads that matched with my target list. There is a file for each target in my list so it creates empty files since not all targets are present in every sample.

1. perl discard\_redundant\_sequences.pl -I uniqueseqs.txt - o umis\_count.txt # I call again this script but this time to be able to count the number of UMIs/target.

As an input I use the file named uniqueseqs.txt, which is a list of the output files generated in the step before.

The output is a file named umis\_count.txt which is a list of all the UMIs/target found in my sample.

\*\*\* A this point the analysis is done but since I compile and group the results manually, I included the following steps (8 and 9) to organize the reads that are in the file umis\_count.txt so it is easier for me to read it.

1. perl sort\_by\_TAGs.pl -i umis\_count.txt -T Path\_TAGs.fas -1# This script splits a list of sequences into shorter groups based on a barcode/sequence. I use it to sort the reads contained on the file ”umis\_count.txt”. The other input file ”Path\_TAGs.fas” contain the list of pathogen barcodes and it generates one file/barcode named i.e. outfile\_AACTG.fas etc. These can also be empty files for some of the barcodes.
2. perl merge\_FASTA.pl -I outfile\_TAG\_sort.txt - o pathcounts.fas #This script merges all the files generated in the previous step in one file call pathcounts.fas. Which is the one I used for the analysis. It uses as an input the list (outfile\_TAG\_sort.txt) of the files generated in the previous step.
3. call del\_TAGs\_outfiles.bat #This calls a batch file that deletes most of the files I don’t need for analysis. It is written in windows console language
4. call del\_empty.bat #This one calls a script that deletes empty files.
5. md Sample\_ #This one simply creates a folder called ”Sample\_” where I move the resulting files.

Blue underlined: These are input files I prepare before running the pipeline. Therse are common to all samples except for inputfilelist.txt, which I change manually for each sample.

Light blue underlined: These are output files I either keep or use for my analysis.

Green underlined: These are files generated during the process tha are deleted after analysis.

**Compiling results:**

Form the file pathcounts.fas, I **manually** create a heat map grouping the total number of sequences/pathogen barcode. So I simply create a table with excel that looks like this:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pathogen** | Aspergillus fumigatus | Candida albicans | Candida dubliensis | Candida glabrata | Citrobacter koseri | Candida paralipsosis | Enterococcus faecalis |
| Path barcode. | TTCTC | TTGCT | TTAGG | TCTTG | TCCGT | TCGAC | TGCAG |
|  |  |  |  |  |  |  |  |
| Sample\_1 | 585 | 352 | 0 | 0 | 586 | 0 | 332 |
| Sample\_2 | 748 | 435 | 0 | 0 | 783 | 0 | 449 |

**Table 1.** Example of the compiled data.

Each cell contains the sum of UMIs per pathogen barcode. I input the values **manually.** For instance, the pathcounts.fas looks like:

>10

TTCTCGGGGTGTTGCGTACACTCGTACGGTGG

>2

TTCTCGTATGACGCTTTACCTAGTCTACGGGAATTGC

>13

TTCTCTCGGAGGTAATCGACCTAGAGTGTCCTAGC

So. I input **manually** in the cell: =10+2+13, and so on and so forth for the rest of the pathogen barcodes.