What's in my food (WIMF) pipeline v1.0

Olivier Emery, Digital Epidemiology Lab (Ecole Polytechnique Fédérale de Lausanne - EPFL)
Campus Biotech Geneva, Switzerland

August 26th 2019

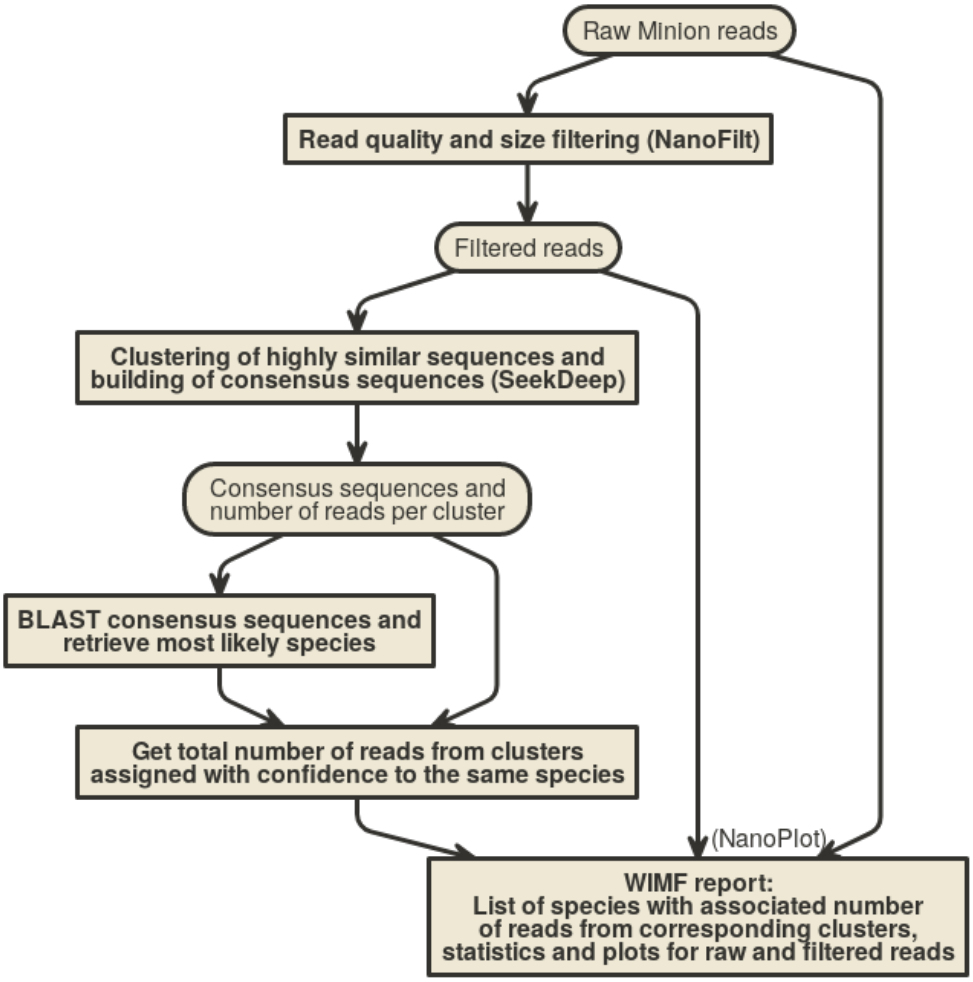
WIMF description

WIMF (for **W**hat's **I**n **M**y **F**ood) is an open source experimental automated bioinformatic pipeline to identify which plants and animals are present in food based on DNA sequences obtained from food samples sequenced with the MinIon instrument (Oxford Nanopore Technologies). It was developped in August 2019 in the context of the Open Food Repo DNA project which is a part of the Open Food Repo initiative. It is a follow up to the Food Repo DNA workshop held on May 25th 2019 in collaboration with the biohacking association Hackuarium and the DNA-based food certification company SwissDeCode in order to analyze the sequencing data obtained. Additional food DNA sequencing data from SwissDeCode were also used to develop this first version of WIMF.

Disclaimer

The WIMF (What's In My Food) pipeline is provided "as is" and "with all faults." It is not affiliated with the WIMP (What's In My Pot) pipeline neither with Oxford Nanopore Technologies. The author makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other potentially harmful components of WIMF neither of the content of the tools used in WIMF or the links provided in this document. There are inherent dangers in the use of any software, and you are solely responsible for determining whether WIMF and its components are compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and the author will not be liable for any damages you may suffer in connection with using, modifying, or distributing WIMF. Importantly, WIMF does not give quantitative results (e.g. a food composed of 50% ingredient A and 50% ingredient B will usually not correspond to 50% reads coming from ingredient A and 50% reads coming from ingredient B for several reasons among which: differential yield of DNA extraction between the ingredients and between various preparation forms, differential amplification of the products from the two ingredients = PCR bias etc...) and should not be used as a proof in determining whether an indicated ingredient is present neither to show that an unexpected ingredient is present.

WIMF workflow summary



WIMF workflow for "typical" samples (i.e. analysis quality filtered at average read quality > Q10). Rounded boxes indicate the different data at each point of the pipeline. The rectangular boxes represent data manipulation processes and arrows the direction of the workflow to and from data and processes. Other checkpoints (i.e. minimal number of raw or filtered reads) or treatment of particularly high quality samples (i.e. which were filtered at >Q12 or even >Q13 before analysis) are not shown here for simplicity.

WIMF input format

WIMF takes as input demultiplexed amplicon sequencing data in **uncompressed FASTQ** format as delivered by the MinIon standard software suite, with one directory per sequencing run containing several directories (the latter with names starting with "BC" - for "Barcode" - containing all reads from one sample). See below how to obtain the appropriate sequencing data. **Tip** Rename your BCXX (XX is 01 or 02 or 03 etc. . .) directory to BCXX_[FOOD_SAMPLE_NAME] with [FOOD_SAMPLE_NAME] replaced by a short name corresponding to the food sequenced with the barcode XX. By doing so you will avoid scrolling through your list of barcodes to find the corresponding food when analyzing your results. When renaming, **always leave `BC` at the beginning of the sequencing run directory name**, this is used by WIMF to recognize directories containing the reads.

Generating DNA sequences for WIMF

The detailed protocol for DNA extraction, amplification and sequencing used to obtain the data suitable for WIMF is available here. Briefly, DNA is extracted from each sample and two specific genomic sequences present in all plants and, respectively, all animals are amplified via polymerase chain reaction (PCR), these two sequences are:

1. a portion of the Ribulose-1,5-bisphosphate carboxylase/oxygenase large subunit ("rubisco") ***rbcL* gene for plants** using the following primer pair:

- **Forward**: TTT CTG TTG GTG CTG ATA TTG CAT GTC ACC ACA A
- **Reverse**: ACT TGC CTG TCG CTC TAT CTT CTA GCA TCG YCC T

2. a portion of the mitochondrial **16S rRNA gene for animals** using the following primer pair:

- **Forward**: TTT CTG TTG GTG CTG ATA TTG CGM CTG TTT MCC
- **Reverse**: ACT TGC CTG TCG CTC TAT CTT CYT CCA YAG GGT).

The resulting amplified sequences (i.e. "amplicons") have an expected size ranging from 250 to 390 base pairs (bp). Samples are barcoded so that PCR products from all samples can be pooled and sequenced simultaneously on a single sequencing run using only one flow cell and later separated (i.e. "demultiplexed") per sample.

Requirements

In order to run WIMF requires the installation of the following freely available softwares and their dependencies (if applicable):

- Python3 (v3.5 or above)
- Git See how to install git here.
- NanoPlot https://github.com/wdecoster/NanoPlot
- PoreChop (github) https://github.com/rrwick/Porechop
- NanoFilt https://github.com/wdecoster/nanofilt
- SeekDeep v2.6 https://github.com/bailey-lab/SeekDeep
- NCBI BLAST+ v2.9.0 NCBI install instructions

Important: Python 3 must be set as default Python interpreter, PoreChop, NanoPlot, NanoFilt, SeekDeep, NCBI BLAST+ utilities need to be accessible from any directory. See the **Troubleshooting guide** on page 10 of this document for help on how to comply with these requirements.

In terms of hardware WIMF requires at least 8Gb of RAM and a sufficient amount of free disk space for the local BLAST nucleotide database (1.4Gb) and for temporary files of each run analysis (2 to 3 times the disk space used for the FASTQ files of the run to be analyzed). WIMF has been developped and tested on Linux (Ubuntu 14.04 LTS 64bit) on a PC with an Intel® Core™ i7-4600U CPU @ 2.10GHz × 4 and 8Gb of RAM. Some programs in WIMF make use of the 4 cores and corresponding parameters may need to be changed from their default values in order to work on a computer with less than 4 cores or to run faster on computers with more than 4 cores.

Installation

After checking that your hardware meets the requirements and installing the required softwares (see Requirements section and Troubleshooting guide), enter the directory where you would like to install WIMF and clone the `WIMF_v1` directory to your computer:

```
git clone https://github.com/salathegroup/Food-Repo-DNA-analysis-WIMF.git
```

Enter the install directory (`WIMF_v1`) and run the `config.sh` script in order to download and install the WIMF v1.0 BLAST database (**Warning** the compressed database has a size of 771Mb, and 1.4Gb once uncompressed, make sure to have 2.2Gb for this step, the compressed database is then removed to free space, a good internet connection is recommended) and to make WIMF accessible from any directory on your computer:

```
cd Food-Repo-DNA-analysis-WIMF/WIMF_v1
./config.sh
```

To test that the installation was successful, enter the following command which should print the WIMF help if WIMF is correctly installed:

```
wimf -h
```

In case this prints an error, you may need to look at the Troubleshooting guide (p.10).

Basic usage

In order to run WIMF on your sequencing run directory (containing the barcoded demultiplexed FASTQ files in directories with names starting with "BC"), execute the following command from the `WIMF_v1` directory (replacing `$PATH_TO_SEQUENCING_RUN_DIRECTORY` with the path of the sequencing run directory, it is recommended to use the double quotes if your directory contains spaces):

```
wimf -i "$PATH_TO_SEQUENCING_RUN_DIRECTORY"
```

Which corresponds to WIMF with default parameters. The results will be stored in a directory called `$PATH_TO_SEQUENCING_RUN_DIRECTORY_`**WIMF** which will be located next to the sequencing run directory, see the "Description of results" section below for details. In case you want to cancel the run press CTL+C.

Important notes In case WIMF is launched for the second time on the same sequencing directory (i.e. with existing WIMF outputs), the previous WIMF outputs will first be deleted before producing outputs from the last command. In case you would like to keep the WIMF outputs of the first run (which can be of interest if you use specific parameters for the different WIMF runs in order to later compare them, see advanced usage section below), just rename the WIMF output directory before launching WIMF for the second time. If you would like to run WIMF on only a subset of the samples, create a directory in which you copy the directories starting with "BC" that you would like to analyze and launch WIMF indicating this directory path instead of `$PATH_TO_SEQUENCING_RUN_DIRECTORY`. In case of interruption of the analysis because of an error, temporary files are kept by default (beware some files may be large in memory), this helps to understand if anything went wrong. This behavior can be changed with the -e option to erase all WIMF intermediary files if a run is interrupted (See Advanced usage section below).

Advanced usage

Changing default parameters

WIMF default settings have been set to values tested to detect plant and animal species using specific molecular markers (i.e. rbcL gene region for plants and 16S gene region for animals). It is possible to change numerous settings to adapt to other molecular markers (i.e. target gene regions specific to genetically different groups which will be PCR-amplified to result in amplicons of known length for each marker). The expected range of amplicon can be provided to WIMF so that reads outside of this range are filtered out using the options `-a` (for the expected length of the smallest amplicon) and `-b` (for the expected length of the smallest amplicon). In the case of different molecular markers, a custom NCBI BLAST database containing annotated sequences corresponding to those markers will need to be generated by the user (See the "Building your own local BLAST nucleotide database" section below to see how to do this) and the NCBI BLAST index (including the path) provided to WIMF using the `-d` option. Other options can be used to suit a particular type of sample or determine how reads are clustered or modify settings to suit another sequencing technology. The goal is to allow users to experiment and profide flexibility in the settings.

For example to set the minimum read length to 200bp and the maximal to 550bp, you can run this command:

```
wimf -i "$PATH_TO_SEQUENCING_RUN_DIRECTORY" -a 200 -b 550
```

More generally

```
wimf -h
```

will display WIMF help and describe all different options/parameters that can be changed to adapt to other specific markers as examplified above, or can be changed to adapt to a particular sample or to experiment different settings for the same sample. In addition, some options may be chosen to adapt to the hardware used, for instance the `-c` option can be set to a higher value if the computer used has more than 8Gb of RAM (the maximum number before SeekDeep crashes will need to be empirically determined), or the `-e` option can be activated to free some disk space by removing all files if the analysis has failed or if it was interrupted by the user (with CTL+C).

Make a global report

In case you would like to gather the WIMF reports from multiple samples (i.e. report.html file and associated images) into the same directory for a more direct access, navigate into the WIMF install directory (`WIMF_V1`) and run the following command (replacing `$INPUT_DIRECTORY` with the directory containing one or several WIMF results directories = ending by `_WIMF`):

```
./mk_wimf_global_report.sh "$INPUT_DIRECTORY"
```

An output directory containing the gathered HTML reports called "GlobalReport" will be created next to "$INPUT_DIRECTORY".

Building your own local BLAST nucleotide database

Here are the steps that were undertaken to build the BLAST local database, you will need to install seqkit to follow the same procedure. NCBI rbcL sequences were downloaded in FASTA format and restricted to plants in nucleotide database using the following search terms (NOT terms to limit downloading whole chromosomes/genomes or unrelated): "rbcl OR rubisco OR ribulose-1,5,bisphosphate carboxylase/oxygenase large subunit NOT chromosome NOT linkage" and saved as `rbcL.fasta` (241'760 entries). Entries with more than 300 kb were removed (exclude remaining genomes but keep chloroplast genomes)

```
cat rbcL.fasta | seqkit seq -M 300000 > rbcL2.fasta # 241'575 sequences
```

Reformat the fasta file so that each entry has only two lines (header+sequence) as specified by the FASTA format (NCBI puts by default 70 nucleotides per line)

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);}  END {printf("\n");}' < rbcL2.f
asta > rbcL3.fasta
```

Remove the first line which is empty

```
tail -n +2 rbcL3.fasta > rbcL.tmp && mv rbcL.tmp rbcL4.fasta
```

Filter out unverified entries (there are 3748)

```
sed '/UNVERIFIED/{ N; d; }' rbcL4.fasta > rbcL5.fasta
```

Delete temporary files

```
rm rbcL.fasta rbcL2.fasta rbcL3.fasta rbcL4.fasta
```

Final number of sequences in rbcL FASTA file (`rbcL5.fasta`): 237'827. Similarly for the 16S genes, 16S gene sequences were downloaded in FASTA format from NCBI and limited to animals and fungi, sorted by length NCBI search of nucleotide database restricted to animals and fungi with the search terms: 16S rRNA OR mitochondrion OR large subunit ribosomal NOT chromosome NOT "mitochondrial 12S" NOT "small subunit" NOT "XXX" NOT "supercontig" NOT scaffold NOT contig NOT genomic saved as `16SanimalsFungi769322.fasta` (769,317 sequences). Remove entries with more than 300 kb (exclude remaining genomes but keep most mitochondrial genomes)

```
cat 16SanimalFungiRAW769322.fasta | seqkit seq -M 300000 > 16S_1.fasta # 767,207 sequences
```

Remove original file `bash rm 16SanimalFungiRAW769322.fastabash` Reformat the fasta file so that each entry has only two lines (header+sequence) as specified by the FASTA format (NCBI puts by default 70 nucleotides per line)

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);}  END {printf("\n");}' < 16S_1.f
asta > 16S_2.fasta
```

Remove first line which is empty

```
tail -n +2 16S_2.fasta > 16S.tmp && mv 16S.tmp 16S_3.fasta
```

Remove temporary file

```
rm 16S_1.fasta
```

Filter out unverified entries (there are 5230)

```
sed '/UNVERIFIED/{ N; d; }' 16S_3.fasta > 16S_4.fasta
```

Remove temporary file

```
rm 16S_2.fasta 16S_3.fasta
```

Final number of sequences in 16S animals+fungi FASTA file (`16S_4.fasta`): 761'977. Concatenate the forementioned 16S and rbcL fasta files

```
cat 16S_4.fasta rbcL5.fasta > FINAL_16SrbcL.fasta # total number of sequences: 999'804
```

Build the BLAST nucleotide database containing rbcL and 16S sequences, named 16SrbcLDB (this name must match the one in the BLAST command in `wimf`)

```
makeblastdb -input_type 'fasta' -dbtype nucl -parse_seqids -in FINAL_16SrbcL.fasta -blastd
b_version 5 -out 16SrbcLDB # set to version file so that query IDs are accepted
```

Editing SeekDeep qluster options

SeekDeep posseses many features to deal with clustering of similar reads. Some of them can be accessed by using available WIMF options (e.g. Identity threshold to form clusters of similar reads with the '-g' option) but more sophisticated ones can only be changed by editing the SeekDeep commands directly in the code of the `wimf` script. It is highly recommended to make a backup copy of the original script in order to be able to come back to it in case the code is broken following edits.

Description of results

When the WIMF analysis is successfully completed, the outputs are located in `$PATH_TO_SEQUENCING_RUN_DIRECTORY_`**WIMF** directory which contains the following **`directories/`** and **`subdirectories/`** (in bold) and `files`:

- `LOGS` contains log files
  - `WIMF.log`: same as standard output, read it with `cat WIMF.log`.
  - `WIMFcompact.log`: a compacted version of `WIMF.log`, to be read with basic text editors.
  - `LowFiltReads.log`: list of samples with not enough filtered reads to carry the full analysis*
  - `LowRawReads.log`: list of samples with not enough raw reads to carry the full analysis* * may not be present depending on the data analyzed
- **`Filt1Samples/`** contains typical samples which passed the full analysis
- `BC_XX` sample XX directory containing the following subdirectories:
  * `REPORT` contains WIMF report in HTML format (`report.html`) as well as a subdirectory (**`HTML_files/`**) containing the files used by the HTML report * **`01_QC_NanoPlot/`** contains NanoPlot outputs for raw reads (plots+statistics)
  * **`02_Filtering_NanoFilt/`** contains filtered reads (`filtSeqs.fastq` and `filtSeqs.fasta`) as well as NanoPlot outputs for filtered reads * **`03_clusterConsensus/`** contains SeekDeep qluster outputs including consensus sequences for the clusters obtained (`output.fastq` and `output.fasta`) * **`04_BLASTclusters/`** contains the following files: * `BLASTresults.out` BLAST results (10 per query) using cluster consensus sequences as queries against the 16SrbcL database * `firstHits.txt` the top BLAST hits (one per query) * `clustersSummary.txt` based on BLAST first hits, each line corresponds to one species and is composed of its name, the number of reads supporting this species, the numbers of clusters and BLAST statistics (in order: % identity to hit, query coverage, alignment length, number of mismatches, number of gap openings, e-value, bit score) determined by a weighted average using the number of reads per cluster as weights for each cluster associated with this species. An internal filter removes hits for which the query coverage is below 90% and the % identity to the hit below 95% before producing this file (most of the time this removes clusters supported by only one read - = singletons - of rather low identity). * `speciesTable.csv` comma separated file containing the name of each species detected and their percentage of reads relative to all reads used (i.e. sum of reads that make the clusters for all species indicated in this file), may be used as input for webpage graphs * `uniqueIDs.txt` cluster consensus sequences IDs (is used to get first BLAST hits)
- **`Filt2Samples/`** contains very high quality samples which passed the full analysis only after the second quality filter, refer to **`Filt1Samples/`** for description of subdirectories
- **`Filt3Samples/`** contains very high quality samples which passed the full analysis only after third quality filter, refer to **`Filt1Samples/`** for description of subdirectories
- **`LowFiltReadsSamples/`** contains samples which did not pass the full analysis because there were not enough filtered reads, includes a report only on the raw and filtered data (without species determination).
- **`LowRawReadsSamples/`** contains samples which did not pass the full analysis because there were not enough raw reads, these include a report only on the raw data.

Notes: **`LowFiltReadsSamples/`** only have plots for raw and filtered data, **`LowRawReadsSamples/`** only plots for raw data. If you would like to share would like to share a report and see it in another computer, share the **`REPORT`** directory including the subdirectory **`HTML_files/ HTML_files/`** which includes plots and javascript definitions to build the barplot of species.

Interpretation of results

Look at the HTML report from a sample to see which species plant and/or animal species were detected based on the WIMF pipeline. For average quality samples (as well as for very and extremely high quality samples), the `SpeciesTable.csv` file will provide scientific species names and the % of reads from clusters matching to that species. Google the name (you can copy it from the species table or from the statistics table of the report) to know which organism it corresponds to. It is possible that the name is not what you expect (especially for plants), in this case see if the species found is from the same plant family in wikipedia (see limitations section below). You can get more insights by looking at the `BLASTresults.out` file to see if the species you expect to find is present among the first results (the first one is used by WIMF, in case several of the first results have identical scores, only the first result is used by WIMF although in this case the others are statistically equally valid). Since the BLAST results include only the first 100 results for each cluster, see if the species you expect to see is among them. If not (particular if all 100 hits have the same score), you can retrieve the cluster consensus sequences in FASTA format (i.e. copy the 5-10 first headers and sequences which contain the largest clusters from the `output.fasta` from the `03_clusterConsensus/` directory, the last number in the header correspond to how many reads were used to build this cluster) and BLAST them by pasting them as queries online at the NCBI website (choose the "Max target sequences" under "Algorithm parameters" at the bottom of this page to 250 to show 250 alignments).

Limitations

Although the identification of species at the family level works well, it is not possible to distinguish different species of the same family because they are identical or close to identical sequences at the amplicon used. This is in particular the case with plant families such as the *Solanaceae* (e.g. potato/chili pepper/paprika), *Apiaceae* (e.g. carrot/parsley/celery), *Fabaceae* (soybean/lentils/other beans) and *Poaceae* (e.g. wheat, wheatgrass and other grasses). For animals, the distinction between species of fish of the *Gadidae* family (e.g. cod/whiting/pollack) or between cattle species (e.g. cow and Zebu) is also ambiguous for similar reasons. Consider targetting a second gene target in order to resolve part of the within-family idenfication ambiguities. The long read technology of the MinIon is under exploited (amplicon sizes of 250-390bp), maybe find universal primers that amplify a longer stretch of DNA (full 16S rRNA gene, provided this improves specificity by analyzing varying regions only), or sequence concatenated amplicons (this would help with error correction of reads, see NanoAmpli-Seq article. With the actual protocol, add PCR product purification: in order to sequence specifically sequences from the amplicons, sequences of the PCR products of each primer pair could be migrated on gel and DNA purified from the band cut from the gel (unless for some reason this approach is not suited for Minion sequencing chemistry). It would be great to record the background (i.e. previous experience with DNA extractions or not) of workshop participants in order to see if samples for which sequencing failed can be explained by this. Based on SwissDecode data, some experiments performed better some worse so the experimentor's background may not explain everything. Think to record the order in which the different samples were blended (in order to explain possible contaminations). To avoid cross contamination of samples: either test and improve the cleaning of the blender to remove all residual materials or, even better, use a bead beater with individual tubes for each sample.

Perspectives

A future version of WIMP using data with additional gene markers in addition to the 16S rRNA gene and the *rbcL* gene would allow greater accuracy. Importantly the choice of markers will dictate how to implement the analysis, in particular the specificity of the marker to certain species or not will play an important role. For instance, markers specifically designed to distinguish species of a given family only (fictitious example: the *Solanaceae* plant family - which includes tomato/pepper/potato among others) or even strains (fictitious examples: cattle/chicken breeds) would need to be used in conjonction with the *rbcL* and 16S rRNA genes, respectively, only for these families/species. Other "general" markers (i.e. that in principle work with most species) similar to the 16S and *rbcL* markers may allow to resolve some ambiguities but not all of them. For plants, the *matK* gene may be an interesting candidate to identify species within plant families although it has been reported not to work with certain plant clades (see Hollingsworth et al., 2009). Another possibility would be to extend WIMF to also screen for bacterial species with an existing 16S rRNA gene marker that has been heavily used previously in many metagenomic amplicon sequencing studies. This would be of particular interest for fermented foods or to potentially detect pathogenic bacteria due to impromper storing or cold chain failure. Similarly, a marker for species of the *Fungi* kingdom (e.g. ITS for **i**nternal **t**ranscribed **s**pacer) could allow to distinguish mushroom species as well as yeast-related organisms such as molds that may be present folowing food spoilage.
WIMF was developed in one month, there are possible code optimizations to avoid redundancy (e.g. making a common function for the different filters that takes as arguments the quality threshold as well as a common function for the clustering of the reads filtered at different quality thresholds using the error quality ranges as arguments) and to make the code more consistent across the whole script or faster (e.g. unnecessary uses of `cat` instead of `<`). In addition the BLAST nucleotide database could be optimized to include only target regions of the amplicons. In the present version, there are full sequences of the 16S rRNA and rbcL genes (1.6kb long) as well as full mitochondrion genomes (up to 300kb) and redundancy for certain organisms (i.e. multiple sequences of the same gene corresponding to the same species). This could allow the database files to take much less hard drive memory not only for faster downloading but possibly also to speed up BLAST searches. No attempts were made to parallelize processes except using options provided by the different softwares used when possible. Finally, the HTML report provides numerous statistics but could be improved in terms of visualization / viewing design.

Extra remarks

WIMF has been developed starting with demultiplexed FASTQ files obtained with the standard software from Oxford Nanopore Technologies and it was not assessed during this work how many reads could not be assigned to a given sample. It may be interesting to test the DeepBinner open source demultiplexing tool in order to see how well it performs relative to the default software from Oxford Nanopore Technologies. WIMF v1.0 performs the full analysis from beginning to the end, future versions should divide the parts of the analysis (e.g. quality control/filtering/clustering/BLAST...) so that the user can decide to stop at any point and resume the analysis from this point. Due to the highly heterogeneous nature of the demultiplexed FASTQ files (some had only a few raw reads, others had over 300'000 with various read lengths and qualities), it is difficult to find a one-for-all solution that works for all samples. WIMF works with the several hundred samples tested but tuning parameters may be needed for your own samples. It was a choice to focus on reads with quality above Q10 and lengths corresponding to expected amplicon sizes. While this worked for most samples, it is possible that in some cases where WIMF says that there are too few filtered reads to carry on the analysis that taking reads of lower qualities or of smaller length could allow to extract relevant information. However this was not investigated here and incorporating lower quality and smaller read lengths could result in more ambiguity to determine the true original amplicon sequence. Although options have been implemented to allow the user to set custom settings, these were not heavily tested.

Troubleshooting guide

Installation problems

Setting Python3 as the default Python interpreter

To check your version of Python, type the following command in your terminal:

```
python --version
```

Which should return a version of Python above 3.5, for instance:

```
Python 3.5.2
```

If you get a version of Python below 3.5 with the previous command, Python3 is either not installed or, alternatively, it is possible that Python3 is installed but not used as default Python interpreter. You can check if Python3 is installed with this command (notice the "3" appended to `python`):

```
python3 --version
```

If you do get a Python3 version above 3.5 with the previous command, this means that you need to set Python3 as your default Python interpreter (see below). If you get a "`command not found`" error, Python3 needs to be installed. To set Python3 as default Python interpreter, you need to add the following line:

```
alias python=python3.5
```

to your ~/.bashrc profile (edit the `~/.bashrc` file using your favorite basic text editor and save changes) so that when you type `python` in the terminal, this becomes equivalent to entering the command `python3.5` . In order for this change to take effect immediately (without the need of rebooting) you can enter:

```
source ~/.bashrc
```

Notes: do not put `alias python=python3.5.2` if you have Python v3.5.2 installed but instead use only the first two digits as above, similarly if you have Python v3.7.3 installed enter `alias python=python3.7` and not `alias python=python3.7.3`

Making programs accessible from any directory

If for example BLAST is installed at `/home/user/WIMF_v1/BLAST_install/ncbi-blast-2.9.0+/bin` you need to execute this (adapt this command to the directory where your BLAST install is located):

```
echo "export PATH=$PATH:/home/user/WIMF_v1/BLAST_install/ncbi-blast-2.9.0+/bin" >> ~/.bashrc
source ~/.bashrc
```

You may need to do this for some of the required softwares depending on how they were installed if you see that they lead to an "command not found" error message.

Installing gcc7 and g++7 compiler (required to install SeekDeep)

Here are the commands I ran to install gcc7 and g++7 compilers in order to install SeekDeep (see Requirements section), it worked on my system but it is not guaranteed to work on yours:

```
sudo add-apt-repository ppa:george-edison55/cmake-3.x
sudo apt-get update
sudo apt-get install cmake
sudo add-apt-repository ppa:jonathonf/gcc-7.1
sudo apt-get update
sudo apt install gcc-7
sudo apt install g++-7
```

Problems running WIMF

Some samples were skipped during the analysis

While it is a normal behaviour of WIMF not to analyze certain samples (e.g. samples with very few reads), it is possible that WIMF skips certain samples which could indeed be analyzed. If this happens, you will see a `BC_XX` directory (with XX being numbers) in the main WIMF results directory. This will indicate at which sample the analysis stopped (provided temporary files are kept which is the default behavior unless using the `-e` option). In this case, you can use the following command from the installation directory (`WIMF_V1`) to retrieve a compact log of the run in order to obtain error messages and to find the cause of the problem (possibly in this troubleshooting guide):

```
./logCompactor.sh $PATH_TO_SEQUENCING_RUN_DIRECTORY/LOGS/WIMF.log
```

Note: replace `$PATH_TO_SEQUENCING_RUN_DIRECTORY` with your actual path. The logCompactor will produce a file called `WIMFcompact.log` next to `WIMF.log`. The compact log file is generated by default for all runs which completed successfully. In case of interruption the commands above are needed to obtain a compact log.

Permission denied error

If you get the following error message while attempting to run `wimf -i "$PATH_TO_SEQUENCING_RUN_DIRECTORY"`:

```
bash: wimf: Permission denied
```

Make the script executable on your computer with the following command from the 'WIMF_V1' installation directory:

```
chmod u+x wimf
```

And retry to launch WIMF with `wimf -i "$PATH_TO_SEQUENCING_RUN_DIRECTORY"`

To obtain the full path of the sequencing directory, navigate in the terminal (with `cd`) until you are inside the sequencing run directory, then execute

```
pwd
```

The printed path corresponds to `$PATH_TO_SEQUENCING_RUN_DIRECTORY`

SeekDeep memory overlimit

If you get the following error message or if your computer freezes (cannot move mouse cursor normally, slows down):

```
20432 Killed  SeekDeep qluster --fastq ${RUN_DIRECTORY}_WIMF_QC/$BC_DIRECTORY/02_Filtering_...
```

This means that you went beyond the memory limit of 8Gb of RAM. Try to relaunch WIMF while keeping all other programs closed (internet navigator, PDF reader etc). If you still get the error, try to lower the threshold of the maximum number of reads to produce clusters (default empirically set to 10'050 reads using a PC with 8Gb of RAM)

SeekDeep Error in aligning

The following error is pretty rare (hundreds of samples were tested and it happened only with a pair of reads from four samples):

```
Clustering at 96.5% identity, quality thresholds Q12, Q10... static void njhseq::alignCalc
::runNeedleDiagonalSave(const string&, const string&, uint32_t, uint32_t, njhseq::alnParts
&), error in aligning:
CTAGCATCGTCGCTGTAACGATCAAGACTGAGTAGACCGTCAGTCCATACAGTTGTCCATGTACCAGTGAAGATTCAGCAGCTACCGCGG
CCCTGCTTCCTCAGGTGGAACTCAGGTTGAGAGTTACTCGAAATGCTGCCAAAATATCGGTATCTTTGGTTTCATAGTCAGGGTATAATA
AGTCAATTTATAATCTTTAACTTCAGCTTTGAATCCAACACTTGCTTTAGTCTCTGTTTGTGGTGACATG
AACACATTACCTACAGTAAGGTAAACATGTTAGTAACAGAACCTTCTTCAAAAGGTCCTACAGGGGGTAGCTACATAAAGGCAATAAAAT
CGACTTTTTTTCTTCTCCAGCAACAGGCTCGATGTGGTAGCATCGCCCTTTGTAACGATCAAGACTGGGTAAGACCGTCCAGTCCATACA
GTTGTCCATGTACCAGTAGAAGATTGGCAGCTACCGCGGCCCCTGCTTCCTCAGGTGGAACTCCAGGTTGAGGAGTTACTCGAAATGCTG
CCAAAATATCAATTATCTTTGGTTTTCATAGTCAGGAGTATAATAAGTCAATTTATAATCTTTAACACACCGAAACTTTGAATCCAACAC
TTGCTTTAGTCTCTGTTTGTGGTGACATG
```

The reason for this error is not clear, for some reason SeekDeep could not align the two sequences that are printed. The solution is to remove those two reads from the dataset. You can use `grep` on the concatenated FASTQ file to find each read corresponding to each sequence (note that the two sequences are separated by a newline in the error message shown above) using each sequence as the pattern in `grep` and delete them manually (don't forget to save the FASTQ file after removing these two reads).