DECONTAMINER INSTALLATION GUIDE

1. DOWNLOAD THE SOURCE CODE

The decontaMiner code can be downloaded from the page:

http://www-labgtp.na.icar.cnr.it/DecontaMiner

Copy the `tar.gz` file in the chosen directory. Extract the archive there by using the command:

```
tar -xvf decontaminer.tar.gz
```

A new directory called `decontaMiner_1.2` will be created in `{your_installation_path}` directory.

2. CHECK AND INSTALL THE DEPENDENCIES

The decontaMiner pipeline is written in the *Bash* shell scripting language. DecontaMiner uses several external tools to perform the conversion, filtering, and alignment steps. It relies on widely used tools that are almost a *de facto* standard when dealing with NGS data alignment and processing. The last steps of the pipeline are based on Perl scripts. Hence the following dependencies must be checked and satisfied.

2.1 DECONTAMINER DEPENDENCIES

Both the following dependencies are **mandatory** for `decontaMiner` to work.

- **BASH AND AWK**
  The decontaMiner pipeline structure is written in `Bash` shell scripting. It makes also use of the `awk` program, available within the standard Bash shell environment.

- **PERL**
  The decontaMiner filtering and collecting scripts are written in the `Perl 5.1` programming language. Perl must be present and installed as a binary executable in the user system (that is, issuing the `perl -v` command at shell prompt must display the Perl version and info). Perl is downloadable at:
  https://www.perl.org/get.html

2.2 EXTERNAL DEPENDENCIES

The NCBI BLAST software is mandatory. Depending on the input provided to the pipeline, and on the chosen filtering analyses, one or more of the other external dependencies should be satisfied.

- **BLAST**
  The NCBI BLAST is used to align the unmapped sequences versus the organisms databases. It can be downloaded at the NCBI ftp server, whose link is at:

https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download

Please do note: decontaMiner has been tested with BLAST version 2.2.29. After a successful installation the user must specify the location of the **BLASTn** executable (e.g. full installation path and name) in the configuration file, as explained in section 4.

- **SAMTOOLS**
  Samtools is used to convert files from BAM to FASTQ format. It can be downloaded at:
  http://www.htslib.org/download/

  Please do note: decontaMiner has been tested with Samtools version 1.3.1. After a successful installation the user must specify the location of the executable (e.g. full installation path and name) in the configuration file, as explained in section 4.

  If you are providing input files in FASTQ or FASTA format, you don't need to have this dependency installed.

- **FASTX**
  The `fastq_quality_filter` script contained in the FASTX-toolkit is used to quality filter the files in FASTQ format. It can be downloaded at:
  http://hannonlab.cshl.edu/fastx_toolkit/download.html

  Please do note: decontaMiner has been tested with FASTX toolkit version 0.0.13. After a successful installation the user must specify the location of the fastq_quality_filter executable (e.g. full installation path and name) in the configuration file, as explained in section 4.

  If you don't need to filter the reads by quality, you don't need to have this dependency installed.

- **SORTMERNA**
  SortmeRNA is used to efficiently remove human ribosomal and mitochondrial sequences from the data. It can be downloaded at:
  http://bioinfo.lifl.fr/sortmerna/sortmerna.php

  Please do note: decontaMiner has been tested with SortMeRNA version 2.1. After a successful installation the user must specify the location of the executable (e.g. full installation path and name) in the configuration file, as explained in section 4.

  If you're not filtering out the ribosomal and mitochondrial sequences, you don't need to have this dependency installed.

3. DOWNLOAD AND BUILD THE CONTAMINATING SEQUENCES DATABASES

DecontaMiner is based on BLASTn to align unmapped reads to contaminating organisms. The supported contaminating organisms are **Bacteria**, **Fungi** and **Viruses**. Also, The `decontaMiner` pipeline relies on the sortMeRNA tool to filter out the ribosomal and mitochondrial sequences.

3.1 RIBOSOMAL AND MITOCHONDRIAL SEQUENCES

The databases containing the sequences must be provided in the compressed format required by sortMeRNA .

3.1.1 DOWNLOAD THE FASTA SEQUENCES

The human mitochondrial sequence (complete genome) can be downloaded from the NCBI website:

https://www.ncbi.nlm.nih.gov/nuccore/251831106/

The human ribosomal sequences must be downloaded from the NCBI website, by selecting the database type as "nucleotide" and submitting the following search:

```
"Homo sapiens"[Organism] AND biomol_rrna[PROP]
```

Download both the mitochondrial and the ribosomal sequences, and create a single FASTA file containing all them.

3.1.2 BUILD THE DATABASE

Once the file in step 3.1.1 is competed, the sortMeRNA utility must be used to generate the database in the correct format. See the sortMeRNA user guide for further details.
Example:

```
./indexdb_rna --ref ./rRNA_databases/Human_ribo_mito.fasta /index/rRNA_db -v
```

3.2 CONTAMINATING ORGANISMS SEQUENCES

The databases containing the genomic sequences must be provided in the NCBI BLAST compressed format.

3.2.1 DOWNLOAD THE FASTA SEQUENCES

The FASTA sequences can be downloaded using the NCBI ftp site
ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq

In the refseq folder the bacteria, fungi and viral subfolders contain the assemblies (`.fna.gz` format) for each of the species. Please follow the instructions given by NCBI FAQs section to automate the downloading process:

https://www.ncbi.nlm.nih.gov/genome/doc/ftpfaq/#allcomplete

Concatenate all the FASTA files into a single file (using for instance the `cat` command).

3.2.2 BUILD THE DATABASES

The database is built from the FASTA file generated in the previous step by using the `makeblastdb` utility of the NCBI BLAST toolkit. For details on the parameters please check table C8 of the following document:

https://www.ncbi.nlm.nih.gov/books/NBK279675/

The `-in` parameter is the file generated at the end of the step 3.2.1. The `-title` parameter will give the name to the database. This name must be specified in the configuration file, as explained in section 4.

Example:

```
 /opt/software/ncbi-blast-2.2.29+/bin/makeblastdb -in all.fna -out Bacteria -
dbtype nucl -title "Bacteria" -parse_seqids -max_file_sz='20GB'
```

4. SET UP THE DEPENDENCIES WITH THE CONFIG FILE

The first decontaMiner script (i.e. `decontaMiner.sh`) requires in input a configuration file. This file must be passed using the `-c` flag (see the user guide) and has a fixed structure that **must not** be changed. An example file is contained in the folder:

```
{your_installation_path}/decontaMiner_1.2/config_files/configure.txt
```

This file is also shown in Figure 1.

A configuration file must contain two different sections (highlighted in **red** in Figure 1), one for each group of dependencies:

- the **external software** dependencies section, `[ext_soft]`
- the **contaminating databeses** section, `[cont_db]`

Please do note: the names of the sections MUST NOT BE CHANGED!

Comments in the file are marked at the beginning of the line with a semicolon (;). They are all in black in the example file of Figure1. Dependencies are set by the user specifying a pair in the format: `dependency_name = {value_of_the dependency}`.

Please do note: the dependency names MUST NOT BE CHANGED!

Each dependency name is unique, and is specific to one of the (already described) sections. In Figure 1 they are highlighted in **blue** (the external software section) and **orange** (the contaminating organism section).

Please do note: the dependencies MUST NOT BE MOVED TO A DIFFERENT SECTION!

The user must specify the values (after the equal sign, in the rightmost part of the pairs) taking into account the directories detected and/or specified in the 'CHECK AND INSTALL THE DEPENDENCIES' paragraph of this guide. The parts that can be changed by the user are indicated in **light blue** in the Figure 1 example file.

4.1. DEPENDENCIES IN THE EXTERNAL SOFTWARE SECTION `[ext_soft]`

The users must specifiy the **absolute** paths to the external softwares. In detail:

4.1.1. `SAMTOOLS_EXEC={your_path_to_samtools_exe}`. Full path to the Samtools executable directory.

4.1.2. `FASTX_EXEC={your_path_to_fastx_exe}`. Full path to the FastX toolkit utility `fastq_quality_filter` binaries.

4.1.3. `BLASTN_EXEC={your_path_to_blastn_exe}`. Full path to the NCBI BLAST suite binaries.

4.1.4. `SORTMERNA_EXEC={your_path_to_sortmerna_exe}`. Full path to the sortMeRNA toolkit binaries.

```
; Decontaminer configuration file.

; Section 1: ext_soft.  Paths to the executables of external softwares.

[ext_soft]

; SAMTOOLS INSTALLED EXECUTABLE PATH (for samtool sort, bam2fq)
SAMTOOLS_EXEC= /home/software/samtools-1.3.1/samtools

; FASTX fastq_quality_filter EXECUTABLE PATH (for fastq_quality_filter)
FASTX_EXEC=  /opt/software/bin/fastq_quality_filter

; BLASTN INSTALLED EXECUTABLE PATH(for megablast)
BLASTN_EXEC= /opt/software/ncbi-blast-2.2.29+/bin/blastn

; SORTMERNA INSTALLED EXECUTABLE PATH
SORTMERNA_EXEC = /opt/software/bin/sortmerna

; Section 2: cont_db.  Paths to the databases of contaminating sequences

[cont_db]

; HUMAN RIBOSOMAL SEQUENCES, SORTEMERNA FORMAT
; PATH OF THE DIRECTORY CONTAINING THE FASTA FILE (.fasta) AND THE
INDEXES (.idx*)
RIBO_DB=/home/software/DECO_DB/Human_rna
; NAME OF THE DB ([name].fasta AND INDEXES  ([name].idx*)
RIBO_NAME=rRNA

; DATABASE OF BACTERIA, BLAST FORMAT
; PATH OF THE DIRECTORY CONTAINING THE COMPRESSED FILES
BACTERIA_DB=/home/software/DECO_DB/Bacteria
BACTERIA_NAME=Bacteria

; DATABASE OF FUNGI, BLAST FORMAT
; PATH OF THE DIRECTORY CONTAINING THE COMPRESSED FILES
FUNGI_DB=/home/software/DECO_DB/Fungi

FUNGI_NAME=Fungi

; DATABASE OF VIRUSES, BLAST FORMAT
; PATH OF THE DIRECTORY CONTAINING THE COMPRESSED FILES
VIRUSES_DB=/home/software/DECO_DB/Viruses

VIRUSES_NAME=Viruses
```

Figure 1. Configuration file example.

4.2. DEPENDENCIES IN THE CONTAMINATING SOFTWARE SECTION `[cont_db]`

For each contaminating organisms, two variables must be specified. The first one is the **absolute** path of the directory in which both the database and the index files in **compressed BLAST format** are located. The second is the database name, as specified during the database creation. See the 'CHECK AND INSTALL THE DEPENDENCIES' paragraph of this guide for details on how to get and install the databases.

4.2.1. `RIBO_DB={your_path_to_ribo_db}`. **Full path** to the directory containing both the compressed files (db and index) of the human ribosomal and mitochondrial sequences.
`RIBO_NAME={name_of_the_db`. **Name** of the human (ribosomal and mito-chondrial) sequences database.

4.2.2. `BACTERIA_DB={your_path_to_bacteria_db}`.**Full path** to the directory containing both the compressed files (db and index) of the bacteria sequences.
`BACTERIA_NAME={name_of_the_db}`. **Name** of the bacteria sequences database

4.2.3. `FUNGI_DB={your_path_to_fungi_db}`.**Full path** to the directory containing both the compressed files (db and index) of the fungi sequences.
`FUNGI_NAME={name_of_the_db}`. **Name** of the fungi sequences database

4.2.4. `VIRUSES_DB={your_path_to_viruses_db}`.**Full path t**o the directory containing both the compressed files (db and index) of the viruses sequences.
`VIRUSES_NAME={name_of_the_db}`. **Name** of the viruses sequences database

DECONTAMINER v1.2 USER GUIDE

1. INTRODUCTION

This is the user guide for the `decontaMiner` tool (version 1.2). The suite is based on the following bash scripts:

- `decontaMiner.sh`: starts the analisys pipeline on the directory containing the unmapped read files. It relies on external software such as megablast (see the Installation guide for all the details), and contains several sub programs written both in bash and Perl. `decontaMiner.sh` converts each of the input files from `bam` to `fastq` format, and from `fastq` to `fasta`, while filtering out the reads falling under the specified quality threshold. Then human ribosomal and mitochondrial reads are removed. Finally, it starts a BLASTn alignment versus the chosen contaminating organisms.

- `filterBlastInfo.sh`: the blast alignments resulting from the previous script are then filtered, and are considered valid if and only if they satisfy **all** the following criteria:
  - alignment quality: a valid alignment must be compliant with the match length, mismatch and gap number settings;
  - genus consistency: a read must not align on more than a genus
  - pairing consistency (**only for paired end data)**: both a read and its mate must align on the same genus

  The output of this script are several. Each alignment is either considered ambiguous, when not fulfilling the above criteria, or valid. Blast table format of the alignments for both the classes are provided. Valid matches on the contaminating organisms are counted and a summary of the results is also stored.

- `collectInfo.sh`: this is the parser for the information extracted with the above described `filterBlastInfo` script. Information on contaminating organisms can be obtained for each of the unmapped files given in input to the `decontaMiner` script. Contaminating organisms can be filtered according to the match count (i.e. the number of reads matching the organism). For bacterial and fungine contamination, output files are produced both at the genus and at the species level. A summary of the contaminations across **all** the samples is also provided. As from the DecontaMiner version 1.2, a HTML static report - based on the above described outputs - is also generated, with the aim to provide a more immediate visualization of the results.

2. HOW TO RUN DECONTAMINER

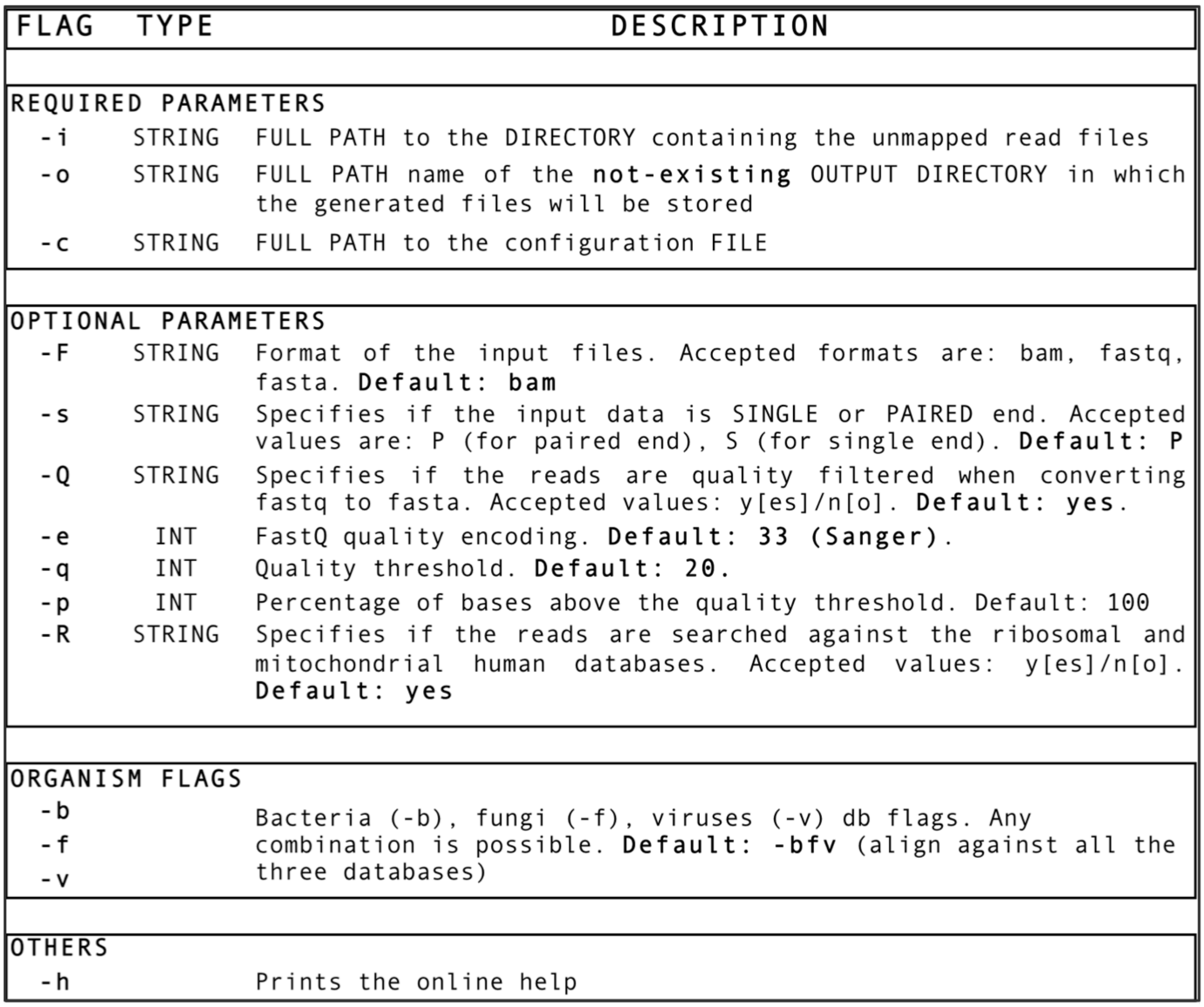
First, run `decontaMiner.sh`. Once terminated, the `filterBlastInfo.sh` script must be executed. Lastly the `collectInfo.sh` script must be run (at least once, but might also be run several times to get different views of the data).

IMPORTANT NOTE: do NOT move or change the names of the output files and directories before completing the whole pipeline!

1.1 RUN THE decontaMiner.sh SCRIPT

INPUT: a directory containing one or more files of unmapped reads. They can be in BAM, FASTQ or FASTA format. The default type is BAM. To specify other format please use the `-F` flag. For paired end data, each of the input files must contain the read of **both strands**. If data are single end, please use the `-s` flag. See Figure 2 for details on the input flags and values.

USAGE: `./decontaMiner.sh [required parameters] [optional parameters] [organism flags]`



FLAG	TYPE	DESCRIPTION
REQUIRED PARAMETERS		
-i	STRING	FULL PATH to the DIRECTORY containing the unmapped read files
-o	STRING	FULL PATH name of the **not-existing** OUTPUT DIRECTORY in which the generated files will be stored
-c	STRING	FULL PATH to the configuration FILE
OPTIONAL PARAMETERS		
-F	STRING	Format of the input files. Accepted formats are: bam, fastq, fasta. **Default: bam**
-s	STRING	Specifies if the input data is SINGLE or PAIRED end. Accepted values are: P (for paired end), S (for single end). **Default: P**
-Q	STRING	Specifies if the reads are quality filtered when converting fastq to fasta. Accepted values: y[es]/n[o]. **Default: yes**.
-e	INT	FastQ quality encoding. **Default: 33 (Sanger)**.
-q	INT	Quality threshold. **Default: 20.**
-p	INT	Percentage of bases above the quality threshold. Default: 100
-R	STRING	Specifies if the reads are searched against the ribosomal and mitochondrial human databases. Accepted values: y[es]/n[o]. **Default: yes**
ORGANISM FLAGS		
-b -f -v		Bacteria (-b), fungi (-f), viruses (-v) db flags. Any combination is possible. **Default: -bfv** (align against all the three databases)
OTHERS		
-h		Prints the online help

Figure 2. List of the `decontaminer.sh` script parameters

USAGE EXAMPLES:

1. PAIRED END BAM, RIBO FILTERING, ALL THE DATABASES

```
./decontaMiner.sh -i /home/data/unmapped_reads -o /home/data/output -c /home/software/decontaMiner_1.2/config_files/configure.txt
```

2. SINGLE END FASTQ, NO RIBO FILTERING, ONLY BACTERIA AND FUNGI

```
./decontaMiner.sh -i /home/data/single_end_fastq/ -F FASTQ –s S -o /home/data/outputFQ -c /home/software/decontaMiner_1.2/config_files/configure.txt -bf –R n
```

OUTPUT: the script creates in the specified output directory `{output_directory}` several folders, to better organise the output of all the steps, see Figure 3. More precisely, an `INTERMEDIATE_FILES` folder collects the output of the conversion, sorting, and ribosomal filtering. For each input file, a new file with the same name and different suffix is created and stored into the appropriate folder. The table files coming from the BLAST alignment step are collected into the `RESULTS` folder, distributed into the `BACTERIA`, `FUNGI`, or `VIRUS` folders depending on the specified organism flags. A `TEMP` folder to hold the shell script generated by the pipeline is created in the directory from which the user calls the script.

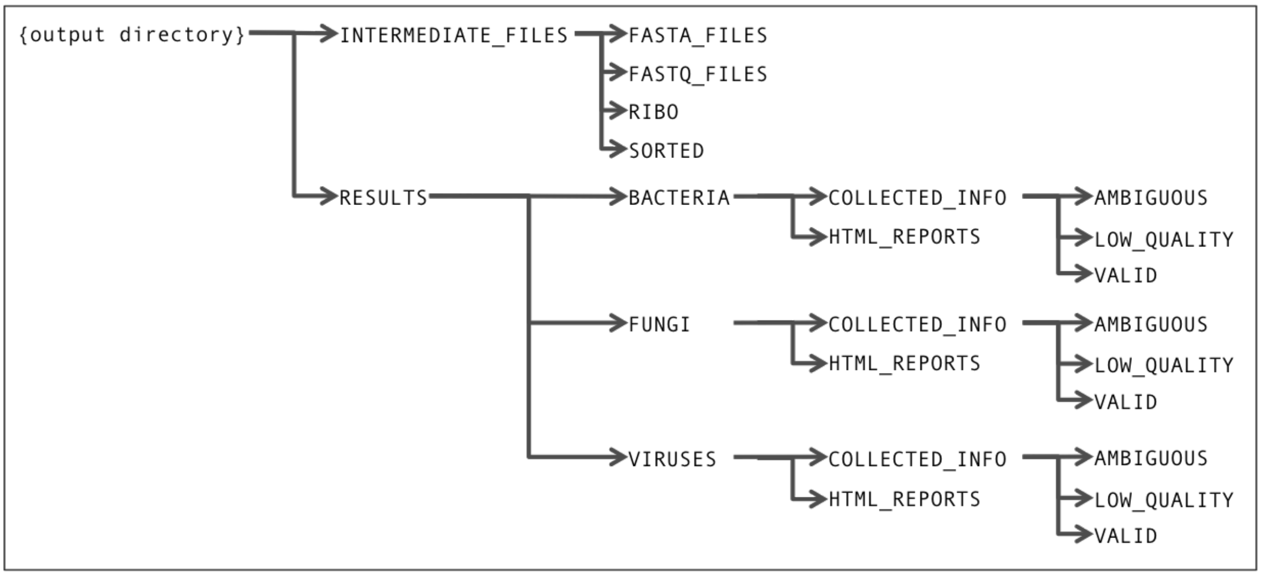
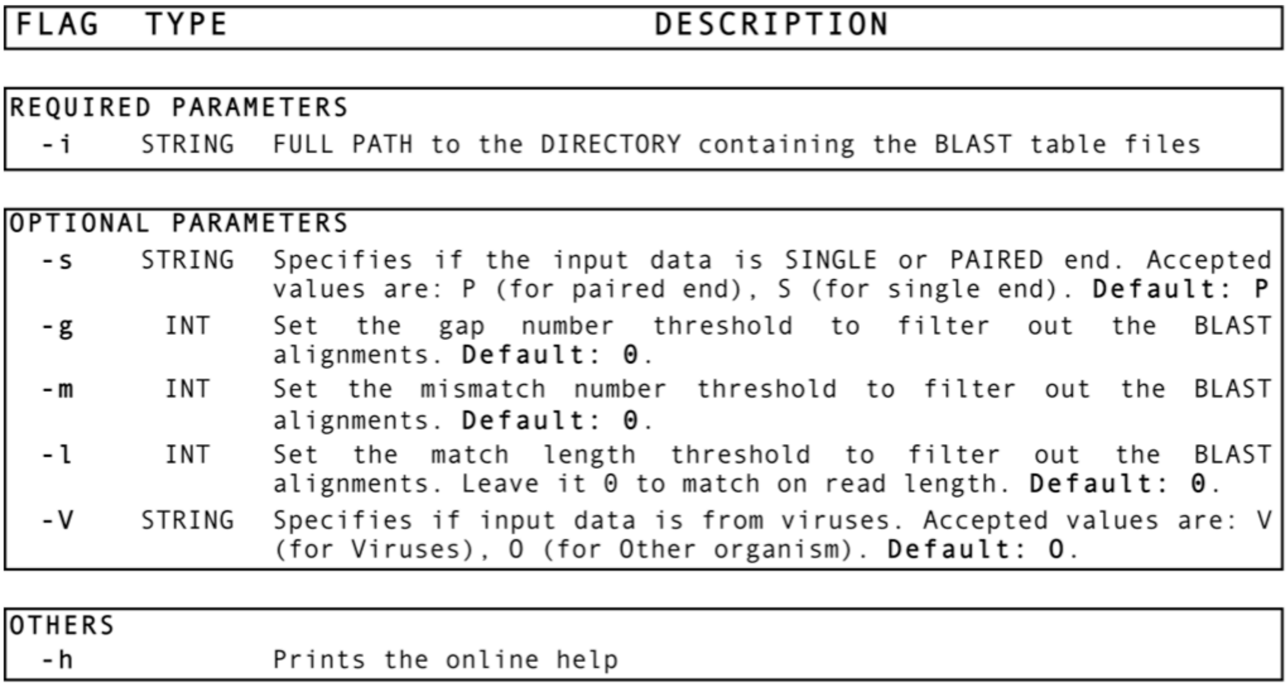


Figure 3. Output folders after the successful execution of the complete `decontaminer` pipeline.

1.2 RUN THE filterBlastInfo.sh SCRIPT

INPUT: a directory containing the BLASTn alignments of the unmapped reads versus a single contaminating organism. The first script of the pipeline generates, into the `RESULTS` folder, a directory for each of the selected organisms. The `BACTERIA`, `FUNGI`, and `VIRUSES` folders contain the output files of the BLAST alignment steps. The `filterBlastInfo.sh` script must be run on each of these folders. It is possible to specify the parameters used to filter out the low quality alignments using the flags `-g`, `-m`, `-l`. When processing alignments deriving from the comparison with viruses the user must declare it by setting the `-V` flag.

USAGE: `./filterBlastInfo.sh [required parameters] [optional parameters]`



```
FLAG   TYPE                      DESCRIPTION

REQUIRED PARAMETERS
  -i    STRING  FULL PATH to the DIRECTORY containing the BLAST table files

OPTIONAL PARAMETERS
  -s    STRING  Specifies if the input data is SINGLE or PAIRED end. Accepted
                values are: P (for paired end), S (for single end). Default: P
  -g     INT    Set the gap number threshold to filter out the BLAST
                alignments. Default: 0.
  -m     INT    Set the mismatch number threshold to filter out the BLAST
                alignments. Default: 0.
  -l     INT    Set the match length threshold to filter out the BLAST
                alignments. Leave it 0 to match on read length. Default: 0.
  -V    STRING  Specifies if input data is from viruses. Accepted values are: V
                (for Viruses), O (for Other organism). Default: O.

OTHERS
  -h            Prints the online help
```

Figure 4. List of the `filterBlastInfo.sh` script parameters

USAGE EXAMPLES:

1. PAIRED END, BACTERIA FOLDER, DEFAULT QUALITY PARAMS

```
./filterBlastInfo.sh -i /home/data/output/RESULTS/BACTERIA
```

2. SINGLE END, FUNGI FOLDER, ONE GAP AND ONE MISMATCH ALLOWED

```
./filterBlastInfo.sh -i /home/data/output_fq/RESULTS/FUNGI –sS –m1 –g1
```

3. SINGLE END, VIRUSES FOLDER, MATCH LENGTH DIFFERENT FROM READ LENGTH

```
./filterBlastInfo.sh -i /home/data/output_fq/RESULTS/FUNGI –sS –VV –l100
```

OUTPUT: the script creates into the input directory a new folder called `COLLECTED_INFO` (see Figure 3) that, in turn, contains three new folders: `AMBIGUOUS`, `LOW_QUALITY` and `VALID`:

- AMBIGUOUS: contains the table files with the alignments not fulfilling the genus and pairing constraints (see the INTRODUCTION paragraph). It also stores the textual files (with the `_ambiguous_stats.txt` suffix), in which an entry for each read describes the reasons of the ambiguity.

- LOW_QUALITY: contains the table of the alignments not fulfilling the quality constraints.

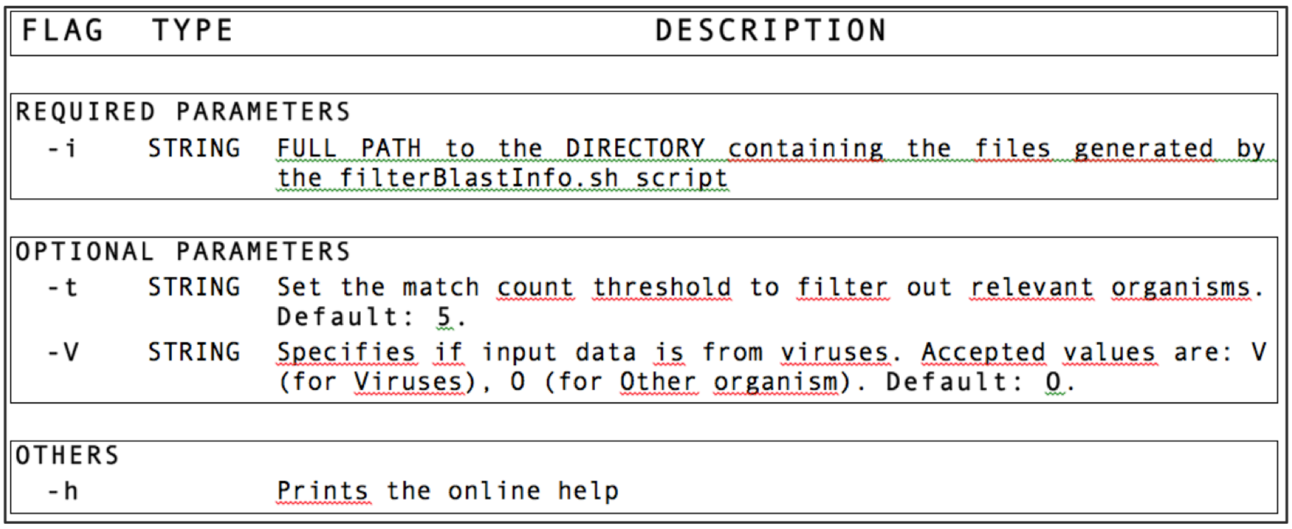
- VALID: contains the table of the alignments fulfilling both the genus/pairing and the quality constraints. Two additional files are generated for each input file:
  - A textual file (with the `_{organisms}_subject_counts.txt` suffix) containing for each contaminating species the number of match counts, i.e. the number of reads aligning on it.
  - A textual file (with the `_{organisms}_alignments_stats_by_species.txt` suffix) containing the number of times a read aligns on a specific species.

In the `COLLECTED_INFO` folder a textual file (with the `_{organisms}_stats.txt` suffix) is also generated, which contains some statistics about each of the processed file (e.g. number of ambiguous/low quality/valid alignment and reads, number of organisms matched).

1.3 RUN THE collectInfo.sh SCRIPT

INPUT: a directory containing all the textual files of the match counts per organism, i.e. the files with the `_{organisms}_subject_counts.txt` suffix. The input is hence one of the `VALID` directories (see also Figure 3) generated by the `filterBlastInfo.sh` script. Also the statistics files (with the `_{organisms}_stats.txt` suffix) contained in the `COLLECTED_INFO` directory are used. When processing alignments deriving from the comparison with viruses the user **must** declare it by setting the `–V` flag. It is also possible to set the match count threshold, used to consider a species as contaminating, by using the `–t` flag.

USAGE: `./collectInfo.sh [required parameters] [optional parameters]`



FLAG	TYPE	DESCRIPTION
REQUIRED PARAMETERS		
-i	STRING	FULL PATH to the DIRECTORY containing the files generated by the filterBlastInfo.sh script
OPTIONAL PARAMETERS		
-t	STRING	Set the match count threshold to filter out relevant organisms. Default: 5.
-V	STRING	Specifies if input data is from viruses. Accepted values are: V (for Viruses), O (for Other organism). Default: O.
OTHERS		
-h		Prints the online help

USAGE EXAMPLES:

Figure 5. List of the `collectInfo.sh` script parameters

1. BACTERIA DATA, GROUPING BY SPECIES, DEFAULT THRESHOLD

```
./collectInfo.sh -i /home/data/output/RESULTS/BACTERIA/COLLECTED_INFO/VALID
```

2. FUNGI DATA, GROUPING BY GENERA, NEW THRESHOLD

```
./collectInfo.sh -i /home/data/output/RESULTS/FUNGI/COLLECTED_INFO/VALID -t100
```

3. VIRUSES DATA

```
./collectInfo.sh -i /home/data/output/RESULTS/VIRUSES/COLLECTED_INFO/VALID  -VV
```

OUTPUT: the script creates into the input directory several textual files, hereafter described:

- a summary file for each of the unmapped file in input. The file contains the information on the contaminating organisms above the chosen threshold. They are named using the suffix: `_subject_summary_GG_CT_XX.txt`, where `GG` is the grouping type. GG is `sp` or `ge` (respectively genus and species) for bacteria and fungi, whereas is always `all` for viruses. The `XX` is the match count threshold.

- the files gathering the information **on all the processed files**. They are called `barPlotInfo_GG_CT_XX.txt` and `barPlotInfo_STAT_GG_CT_XX.txt`, with the same meaning as above for `GG` and `XX`. The first is a matrix: the rows are the contaminating organisms, the columns all the processed samples, and the elements are the percentages of organism found in a sample. This file might be used to generate a BarPlot of the distribution of the contaminating organisms among the input samples. In the second file the number of total reads and contaminating organisms are reported for each input sample.

The script generates also one or more static `HTML` pages based on the textual files above described.
Those files are located in the `HTML_REPORTS` folder of each organism type (see Figure 3). Each report is made of a single index page, and of a folder containing one html file for each of the samples in the input dataset. The main page is called `index_GG_CT_XX.html`, the folder `SAMPLES_HTML_PAGES_GG_CT_XX.html`, and the files inside are named `{sample_name}_GG_CT_XX.html`. The `GG` and `XX` suffixes have the same meaning as above. For bacteria and fungi, reports are generated for both the grouping conditions (i.e. by genus and by species), whereas for viruses only a report is produced.

Open the index file in your browser to visualize the information of all the samples, and to navigate in the individual sample pages.