Supplementary Document 1: Databases Construction

1. Download of reference genomes and databases

1.1. NCBI NT database

NCBI BLAST Command-Line Applications tool was used to download this database. The corresponding FASTA and sequence accession to taxID relationship (seq2taxid) files were obtained in a format compatible with Kraken2. These files were examined to filter sequences in the FASTA file without relationships in the seq2taxid file and vice versa using self-made scripts (check_seq2taxid_accession.py).

Launch the following single commands in the bash shell:

```
#Download NT BLAST database
update_blastdb.pl --decompress nt [*]
#Get multi-FASTA file
blastdbcmd -entry all -db nt -out nt.fa
#Get seqs2taxids relationships in Kraken2 format
blastdbcmd -db nt -entry all -outfmt "%a %T" | awk -v OFS='\t' '{print "TAXID", $1,
$2}' >> nt_seq2taxid_raw.txt
#Remove empty seq2taxid relationships
##Get FASTA file sequence accessions
cat nt.fa | grep '>' | sed 's/>//g' | awk '{print $1}'> nt_accessions.txt
##Run check_seq2taxid_accession.py(script)
python3 check_seq2taxid_accession.py -c nt_accessions.txt -r nt_seq2taxid_raw.txt -o
nt -f kraken2
```

The "check_seq2taxid_accession.py" script will generate 3 result files: raw+cab file (common relationships), raw_only file (relationships found only in the raw seqs2taxids file) and cab_only file (with sequences accessions found only in the cab file). In this case, the raw+cab resulting file is used as seq2taxid file.

1.2. Microbiome enrichment studies

1.2.1. NCBI GenBank studies

Download process

In the case of studies from NCBI GenBank, all available genomes at any assembly level related to each microbiome study were downloaded through the NCBI Website (https://www.ncbi.nlm.nih.gov/). Flags “Latest GenBank” and “Exclude anomalous” were used to obtain the corresponding assembly FASTA files. These FASTA files were decompressed and combined (using the cat command), generating a multiFASTA file for each microbiome study. Studies HMP, FDA_ARGOS, CGR, BIO-ML and Almeida were downloaded in this manner.

Generating seq2taxid files

The taxonomizr package (version0.5.3; https://CRAN.R-project.org/package=taxonomizr) was used to obtain the corresponding seq2taxid relationships. The NCBI's Taxonomy Database was downloaded in April 2020, which was used to create an SQLite database following taxonomizr's documentation. The seq2taxid files were created using self-made scripts and transformed to Kraken2 format.

Launch the following single commands in the bash shell:

```
#Get FASTA file sequence accessions
cat study_multiFASTA.fa | grep '>' | sed 's/>//g' | awk '{print $1}'>
study_accessions.txt
#Get seqs2taxids relationships with Rscript
Rscript accession2taxid.R study_seq_accessions_file[with extension] out_name[with out
extension]
#Convert to Kraken2 format
cat out_file.txt | awk -v OFS='\t' '{print "TAXID", $1, $2}' >
seq2taxid_study_kraken2.txt
```

Repeat this for each NCBI study's multiFASTA file. The "accession2taxid.R" script will generate 2 result files: out file (seq2taxid relationships) and a NA file (with the accessions that were without seq2taxid information, that is an empty relationship). The resulting out file can be used as seq2taxid file (after conversion to Kraken2 format), if no empty relationships are found.

BIO-ML exception

No study returned empty relationships except for BIO-ML. In this particular case, the remaining empty relationships were obtained with the Esearch tool from Entrez Programming Utilities tool. This is the workaround that we came up with at the time. We exploited the fact that in this particular case, the remaining sequence accessions were related to their WGS Project (which were the 6 first letters of every sequence accession). First, all remaining accessions were processed to obtain only one accession per assembly. Then Esearch was used to obtain their taxIDs. Some empty results were returned, and so these few exceptions were searched through the NCBI Website (https://www.ncbi.nlm.nih.gov/) and manually corrected. Then this information was used to relate the project taxID of the assemblies with their accessions. Finally, Seq2taxid files were merged and transformed to Kraken2 format.

Launch the following single command in the bash shell:

```
#Get one accession per project
cut -c-6 seq2taxid_BIO-ML_NA.txt | sort | uniq | sed 's/$/000001/' >
accessions_project_BIO-ML_NA.txt
```

Run the following bash code to get the remaining project taxIDs with Esearch:

```bash
#!/bin/bash
NAS_FILE='accessions_project_BIO-ML_NA.txt'
OUT_FILE='proyect2taxid_BIO-ML_NA.txt'
for acc in $(cat $NAS_FILE)
do
  project=$(echo $acc | cut -c-6)
  taxid=$(esearch -db nuccore -query $acc | \
  elink -target taxonomy | esummary | \
  xtract -pattern DocumentSummary -element TaxId)

  printf "$project\t$taxid\n" >> $OUT_FILE
done
```

Run the following Python3 code to relate project taxIDs with their accessions:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import pandas as pd
##Open output file
out_put_rel=open('seq2taxid_BIO-ML_NA_corrected.txt', 'w')
##Open input files and process lines
tabla_rel=pd.read_csv('proyect2taxid_BIO-ML_NA.txt', sep='\t',header=None)
with open('seq2taxid_BIO-ML_NA.txt', 'r') as input_lines:
  lines=input_lines.readlines()
  for line in lines:
    project=line[0:6]
    index=tabla_rel.index.get_indexer_for((tabla_rel[tabla_rel[0]==project].index))[0]
    taxId=int(tabla_rel[1][index])
    frase=line.replace('\n','')+'\t'+str(taxId)+'\n'
    out_put_rel.write(frase)
out_put_rel.close()
```

Launch the following single command in the bash shell:

```bash
# Combine seq2taxid files and transform to Kraken2 format
cat seq2taxid_BIO-ML.txt seq2taxid_BIO-ML_NA_corrected.txt | awk -v OFS='\t' '{print
"TAXID", $1, $2}' > seq2taxid_BIO-ML_kraken2.txt
```

1.2.2. HBC study

Download process

Assemblies for the HBC study were downloaded from the European Bioinformatics Institute (EMBL-EBI) server, obtaining the corresponding FASTA files (http://ftp.ebi.ac.uk/pub/databases/metagenomics/genome_sets/hbc_genomes.tar.gz). These files were decompressed and combined (using the cat command), generating a multiFASTA file for the study.

Generating seq2taxid files

For the original FASTQ files, technical information (study_accession, tax_id, run_accession, secondary_sample_accession, and submitted_ftp) from which the assemblies were generated, was retrieved through the ENA Browser. We were able to relate "submitted_ftp" with the FASTA files names. In the case of FASTA file "12718_7_11.fa" there wasn't any "tax_id" value, but this could be obtained from ENA's Mgnify Genomes Metadata (http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_genomes/human-gut/2019_09/

genomes_metadata.tsv). Eighty-four assemblies under NCBI's taxID 77133 (uncultured bacterium) were found. Inspecting the supplementary information of the original publication, we saw that these assemblies had another taxonomy information. We processed this information to obtain a list of their taxonomy names at the species level and used the NCBI's Taxonomy Name/ID Status Report Page (https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi) to procure their taxIDs and correct their values. Finally, this information is used to create the seq2taxid file relating sequence accessions with the values of their FASTA files names in Kraken2 format. We exploited the fact that in this particular case, sequence accessions were related to their FASTA file names.

Launch the following single command in the bash shell:

```
#Get FASTA file sequence accessions
cat hbc_genomes.fa | grep '>' | awk '{print $1}' | cut -f2 -d '>' > accessions_HBC.txt
```

Run the following Python3 code to relate project taxIDs with sequence accessions, generating seq2taxid in Kraken2 format:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import pandas as pd
#Open output file
out_put_rel=open('seq2taxid_HBC_kraken2.txt', 'w')
#Open input files and process lines
tabla=pd.read_csv('hbc_sample_taxID_corrected.csv', sep=',')
with open('accessions_HBC.txt', 'r') as input_lines:
  lines=input_lines.readlines()
  for line in lines:
    cab_sample_accession=line.split('.')[1]
    index=tabla.index.get_indexer_for((tabla[tabla['id_fasta']==cab_sample_accession].index))[0]
    taxId=tabla['tax_id'][index]
    frase='TAXID\t'+line.replace('\n','').replace('>','')+'\t'+str(taxId)+'\n'
    out_put_rel.write(frase)
out_put_rel.close()
```

1.2.3. Nayfach study

Download process

For the Nayfach study, through the ENA Browser (https://www.ebi.ac.uk/ena/browser/home), the related metadata of the "Genome assembly contig sets" was downloaded, then processed to generate the download URLs for the FASTA files. The different FASTA files were then combined generating a unique multiFASTA file for the study (using the cat command).

Launch the following single command in the bash shell:

```
#Get FASTA file names from ENA metadata file
cat ena.xml | grep 'entry accession' | awk ' {print $2} ' | cut -f2 -d "=" | cut -f2 -
d'"' > Nayfach_accesion.txt
```

Run the following Python3 code to get download URLs for FASTA files and FASTA file names with-out extension (names_cut, for generating seq2taxid files later):

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
##Open output files
out_put_url=open('Nayfach_urls.txt', 'w')
out_put_cut=open('Nayfach_accession_cut.txt', 'w')
##Open input file and process lines
with open('Nayfach_accesion.txt', 'r') as input_lines:
  lines=input_lines.readlines()
  for line in lines:
    cut_code=line[0:8]
    url='ftp://ftp.ebi.ac.uk/pub/databases/ena/wgs/public/ca/'+cut_code+'.fasta.gz\n'
    out_put_url.write(url)
    out_put_cut.write(cut_code+'\n')
out_put_url.close()
out_put_cut.close()
```

Run the following bash code to download the FASTA files with wget command:

```bash
#!/bin/bash
urls=/path/to/Nayfach_urls.txt
cat $urls | xargs -n 1 -P 11 wget -q
```

Generating seq2taxid files

The related ENA metadata included the taxIDs of each FASTA file. This information was used to relate this taxIDs with the different sequence accessions, generating the corresponding seq2taxid file in Kraken2 format. We exploited the fact that in this particular case, sequence accessions were related to their FASTA file names.

Launch the following single commands in the bash shell:

```bash
#Get FASTA file taxIDs from ENA metadata
cat ena.xml | grep 'taxId' | cut -f3 -d "=" | cut -f1 -d '>' | cut -f2 -d '"' >
Nayfach_taxIds.txt
#Get sequence accessions
cat Nayfach.fa | grep '>' | awk '{print $1}' | cut -f2 -d '>' >
seq_accessions_Nayfach_2019.txt
```

Run the following Python3 code to create seq2taxid file in Kraken2 format:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import pandas as pd
##Open output file
out_put_rel=open('Nayfach_2019_seq2taxId.txt', 'w')
##Read input files and process lines
names_cut=pd.read_csv('Nayfach_accession_cut.txt', header=None)
taxIds=pd.read_csv('Nayfach_taxIds.txt',header=None)
with open('seq_accessions_Nayfach_2019.txt', 'r') as input_lines:
  lines=input_lines.readlines()
  for line in lines:
    acc_cut=line.split('|')[1][0:8]
    index=names_cut.index.get_indexer_for((names_cut[names_cut[0]==acc_cut].index))[0]
    taxId=taxIds[0][index]
    frase='TAXID\t'+line.replace('\n','')+'\t'+str(taxId)+'\n'
    out_put_rel.write(frase)
out_put_rel.close()
```

Status update

As of today, this form of downloading and processing would not be necessary, as the Nayfach study has been incorporated into the NCBI. The download and processing procedure described in section "1.2.1. NCBI GenBank studies" would be recommended instead.

1.3. Generating the Microbiome Studies Genomes Dataset

The different studies of interest (FASTA and seq2taxid files) were combined, obtaining a dataset of Microbiome Studies Genomes (MSG) that was further processed.

Launch the following single commands in the bash shell:

```
#Generate multiFASTA file
cat /path/to/HMP.fa /path/to/FDA_ARGOS.fa /path/to/CGR.fa /path/to/BIO-ML.fa
/path/to/hbc_genomes.fa /path/to/Almeida.fa /path/to/Nayfach.fa > MSG.fa


#Generate seq2taxid file
cat /path/to/seq2taxid_HMP_kraken2.txt /path/to/seq2taxid_FDA_ARGOS_kraken2.txt
/path/to/seq2taxid_CGR_kraken2.txt /path/to/seq2taxid_BIO-ML_kraken2.txt
/path/to/seq2taxid_HBC_kraken2.txt /path/to/seq2taxid_Almeida_kraken2.txt
/path/to/seq2taxid_Nayfach_kraken2.txt > seq2taxid_MSG_kraken2.txt
```

1.4. FASTA Extensions-Explanatory note

The various FASTA files were obtained from different sources. Therefore, we found examples with different FASTA file extensions (.fasta, .fa, .fna), the extension can be changed in the code without problems according to our needs.

2. Processing and construction of databases

2.1. MAG filter

In MAG studies, it is not unusual to find genomes with incomplete taxonomies that, although are resolved as species at the assembly level (usually under the 95% average nucleotide identity criterion), are not resolved as species at the taxonomic level. In this situation, artefactual species may appear, namely a species-level taxID that presents an incomplete taxonomy and could act as a catch-all taxon comprising assemblies of different species. Therefore, we examined the effects of incomplete-taxonomy MAGs on genome database enrichment. With that purpose, we filtered these genomes, retaining those that reached a genus level or were Candidatus genomes (the provisional name for well-characterized but as-yet uncultured organisms).

In these particular case, both Almeida and Nayfach studies were composed entirely by artefactual species. Therefore, an assembly genome was considered to have reached the genus level if its name contained the term " sp.", and was considered Candidatus if its name contained the term "Candidatus".

2.1.1. Filtering Almeida study

The Almeida study consisted of 1952 genome assemblies, of which 1179 reached the genus level, and 10 were Candidatus genomes. We obtained the corresponding "Assembly Details" from the NCBI Website (https://www.ncbi.nlm.nih.gov/bioproject/PRJEB26432) and filtered the undesired assemblies.

Launch the following single commands in the bash shell:

```
#Filter AssemblyDetails to get a list with the assemblies that we want to filter out
cat PRJEB26432_AssemblyDetails.txt | grep -v ' sp.' |grep -v 'Candidatus' | grep -v
'#'| awk '{print $1}' >> assemblies_noiselist_Almeida.txt
```

Run the following bash code in the directory where the assemblies FASTA files are located to generate a multiFASTA file with all the MAGs of interest:

```
#!/bin/bash
INPUT_DIR='path/to/FASTAS/directory'
NOISE_LIST='assemblies_noiselist_Almeida.txt'
cd $INPUT_DIR
for file in $(cat $NOISE_LIST)
do
  rm $file*
done
cat *.fna > Almeida_MAGs-filtered.fa
```

Launch the following single commands in the bash shell:

```
#Filter seq2taxid relationships
##Get filtered-FASTA file sequence accessions
cat Almeida_MAGs-filtered.fa | grep '>' | sed 's/>//g' | awk '{print $1}' >
Almeida_MAGs-filtered_accessions.txt
##Run check_seq2taxid_accession.py(script)
python3 check_seq2taxid_accession.py -c Almeida_MAGs-filtered_accessions.txt -r
seq2taxid_Almeida_kraken2.txt -o Almeida_MAGs-filtered -f kraken2
```

The seq2taxid file generated in the "1.2.1. NCBI GenBank studies" section for the Almeida study is used as raw file (-r). The "check_seq2taxid_accession.py" script will generate 3 result files: raw+cab file (common relationships), raw_only file (relationships found only in the raw seqs2taxids file) and cab_only file (with sequences accessions found only in the cab file). In this case, the raw+cab resulting file is used as seq2taxid file.

2.1.2. Filtering Nayfach

On the other hand, the Nayfach study consisted of 2058 genome assemblies, of which 707 reached the genus level, and 36 were Candidatus genomes. We used the NCBI's Taxonomy Name/ID Status Report Page (https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi) to procure their taxonomy names based on their taxIDs. This information is combined in a 3 column table, including their 1) FASTA name, 2) taxID and 3) taxonomy name (Nayfach_metadata.csv) that is used to filter the undesired assemblies (the first column must be the FASTA names).

Launch the following single commands in the bash shell:

```
#Filter metadata to get a list with the assemblies that we want to filter out
cat Nayfach_metadata.csv | grep -v ' sp.' | grep -v 'Candidatus' | awk '{print $1}' >>
assemblies_noiselist_Nayfach.txt
```

Run the following bash code in the directory where the assemblies FASTA files are located to generate a multiFASTA file with all the MAGs of interest:

```
#!/bin/bash
INPUT_DIR='path/to/FASTAS/directory'
NOISE_LIST='assemblies_noiselist_Nayfach.txt'
cd $INPUT_DIR
for file in $(cat $NOISE_LIST)
do
  rm $file*
done
cat *.fasta > Nayfach_filter_MAGs.fa
```

Launch the following single commands in the bash shell:

```
#Filter seq2taxid relationships
##Get filtered-FASTA file sequence accessions
cat Nayfach_filter_MAGs.fa | grep '>' | sed 's/>//g' | awk '{print $1}' >
Nayfach_MAGs-filtered_accessions.txt
##Run check_seq2taxid_accession.py(script)
python3 check_seq2taxid_accession.py -c Nayfach_MAGs-filtered_accessions.txt -r
seq2taxid_Nayfach_kraken2.txt -o Nayfach_MAGs-filtered -f kraken2
```

The seq2taxid file generated in the "1.2.3. Nayfach study" section is used as raw file (-r). The "check_seq2taxid_accession.py" script will generate 3 result files: raw+cab file (common relationships), raw_only file (relationships found only in the raw seqs2taxids file) and cab_only file (with sequences accessions found only in the cab file). In this case, the raw+cab resulting file is used as seq2taxid file.

Status update

As of today, this form of downloading and processing would not be necessary, as the Nayfach study has been incorporated into the NCBI. The filtering procedure described in section "2.1.1. Filtering Almeida study" would be recommended instead.

2.1.3. Generate the Microbiome Studies fMAG Genomes dataset

At the end of this process, we combined the rest of the microbiome studies genomes with the MAG-filtered studies (FASTA and seq2taxid files), thus generating an alternative dataset for enrichment of Microbiome Studies fMAG Genomes (MSG-fMAG) that was further processed.

Launch the following single commands in the bash shell:

```
#Generate multiFASTA file
cat /path/to/HMP.fa /path/to/FDA_ARGOS.fa /path/to/CGR.fa /path/to/BIO-ML.fa
/path/to/hbc_genomes.fa /path/to/Almeida_MAGs-filtered.fa
/path/to/Nayfach_filter_MAGs.fa > MSG-fMAG.fa
```

```
#Generate seq2taxid file
cat /path/to/seq2taxid_HMP_kraken2.txt /path/to/seq2taxid_FDA_ARGOS_kraken2.txt
/path/to/seq2taxid_CGR_kraken2.txt /path/to/seq2taxid_BIO-ML_kraken2.txt
/path/to/seq2taxid_HBC_kraken2.txt /path/to/seq2taxid_Almeida-fMAG_kraken2.txt
/path/to/seq2taxid_Nayfach-fMAG_kraken2.txt > seq2taxid_MSG-fMAG_kraken2.txt
```

2.2. Dustmasker and Nfilter

Low-complexity sequences are common features shared by different organisms' genomes, and thus less informative for their use in alignments and classification tools. So to deal with this issue and prevent false positives, many classifications tools such as BLAST programs, Centrifuge, or Kraken2 often mask these sequences by default. These classifiers use the NCBI's DustMasker program to perform the low-complexity masking. We used DustMasker (version 1.0.0; Package public 22.0.0) with default parameters on the NCBI NT database, the MSG dataset and the MSG-fMAG dataset. All the masked nucleotides from the DustMasker output were masked as N using the sed command. All completely masked sequences (100%N) were filtered using the "filterN_seqs.sh" script (which uses the "reformat.sh" script of the BBTools suite). At this point, we obtained the NT_DB database used in the comparative, the MS Set and MS-fMAG Set for enrichment.

Launch the following single commands in the bash shell:

```
#Mask sequences with dustmasker
dustmasker -infmt fasta -in unmasked.fa -outfmt fasta | sed '/^>/! s/[^AGCT]/N/g' >
masked.fa
#Filter completely masked sequences (100%N) with script
filterN_seqs.sh -p /path/to/bbmap/reformat.sh -i masked.fa -o masked_Nfiltered.fa
#Filter seqs2taxids relationships
##Get FASTA file sequence accessions
cat masked_Nfiltered.fa | grep '>' | sed 's/>//g' | awk '{print $1}'>
masked_Nfiltered_accessions.txt
##Run check_seq2taxid_accession.py(script)
python3 check_seq2taxid_accession.py -c masked_Nfiltered_accessions.txt -r
seq2taxid_unmasked_kraken2.txt -o masked_Nfiltered -f kraken2
```

The "check_seq2taxid_accession.py" script will generate 3 result files: raw+cab file (common relationships), raw_only file (relationships found only in the raw seqs2taxids file) and cab_only file (with sequences accessions found only in the cab file). In this case, the raw+cab resulting file is used as seq2taxid file.

2.3. Non redundant filter

An important characteristic of a database is non-redundancy, which offers advantages such as reducing the final database size, thus increasing the classification speed and reducing the search effort. We considered two types of redundancy: (a) When combining different databases, there may be sequences with identical accessions. (b) In the case of assemblies obtained from NCBI, there is redundancy between the RefSeq and GenBank databases. It is because the same assembly can be associated with different sequence accessions. And so, we followed a series of steps to minimize redundancies between the NT_DB and the MS and MS-fMAG Sets, respectively.

It is noteworthy that this particular filter could not be applied for the HBC and Nayfach studies, as they did not have sequence accessions found in GenBank or RefSeq at the time.

2.3.1. Get assemblies summary information

Every NCBI assembly is associated with a series of files that contain relevant information about it. To reach them, we first need to get the paths to the directories of each assembly. This information is available in the assemblies summary files associated with the GenBank database. In the case of Microbiome studies assemblies from NCBI, we downloaded the GenBank assemblies summary file from the NCBI server (https://ftp.ncbi.nlm.nih.gov/genomes/ASSEMBLY_REPORTS/assembly_summary_genbank.txt), which was then filtered, retaining only the accessions of interest.

Launch the following single commands in the bash shell:

```
ls | grep '.fna' | cut -d'_' -f1,2 >> assembly_accessions_study.txt
```

Repeat this for each NCBI study's assemblies directory and get a list of the different assembly accessions. We exploited the fact that NCBI FASTA files names include this information in their names (which corresponds to the first part of the name).

Run the following bash code to filter the GenBank assemblies summary file (assembly_summary_genbank.txt):

```
#!/bin/bash
FILE1='GenBank_assemblies_Almeida.txt'
FILE2='GenBank_assemblies_BIO-ML.txt'
FILE3='GenBank_assemblies_CGR.txt'
FILE4='GenBank_assemblies_FDA_ARGOS.txt'
FILE5='GenBank_assemblies_HMP.txt'
FILE_A='assembly_summary_genbank.txt'
OUTPUT='assembly_summary_genbank_ennrich.txt'
for assem in $(cat $FILE1 $FILE2 $FILE3 $FILE4 $FILE5)
do
    line=$(cat $FILE_A | grep $assem)
    printf "$line\n" >> $OUTPUT
done
```

2.3.2. Download assembly report files

Every NCBI assembly has an assembly report file that contains the relationships between GenBank and RefSeq sequence accessions for the assembly.

Run the following bash code to download the various report files:

```
#!/bin/bash
FILE1='assembly_summary_genbank_ennrich.txt'
for url in $(cat $FILE1 | awk -F $'\t' '{print $20}'| sed
's/$/\/*_assembly_report.txt/')
do
    wget $url
done
```

2.3.3. *Get GenBank-RefSeq sequence accessions lists*

The obtained files were processed to generate a list of GenBank-RefSeq sequence accessions for the NCBI assemblies of the dataset of Microbiome Studies Genomes (MSG).

Run the following bash code to generate the accessions list:

```
#!/bin/bash
DIREC_IN='/path/to/assembly/reports/directory'
OUT_FILE='GenBank-RefSeq_sq_accessions_list.txt'
cd $DIREC_IN
for file in $(ls | grep 'assembly_report.txt')
do
  cat $file | grep -v '^#' | sed '/^ *$/d' | \
  awk -v OFS='\t' '{print $5, $7}'| \
  sed 's/\t/\n/g' >> $OUT_FILE
done
```

2.3.4. *Filter redundancies*

In an attempt to minimize redundancies, we used this list to filter the NT_DB database by using the "filterbyname.sh" script of the BBTools suite. Additionally, the seq2taxid file generated in the "2.2. Dustmasker and Nfilter" section for the NT_DB is filtered using the "check_seq2taxid_accession.py" script.

Launch the following single commands in the bash shell:

```
#Filtering sequences from FASTA file
filterbyname.sh in=nt_masked_Nfiltered.fa out=fNT_DB.fa names=GenBank-
RefSeq_sq_accessions_list.txt include=f
#Filter seqs2taxids relationships
##Get FASTA file sequence accessions
cat fNT_DB.fa | grep '>' | sed 's/>//g' | awk '{print $1}'> fNT_DB_accessions.txt
##Run check_seq2taxid_accession.py(script)
python3 check_seq2taxid_accession.py -c fNT_DB_accessions.txt -r
seq2taxid_nt_masked_Nfiltered_kraken2.txt -o fNT_DB -f kraken2
```

The "check_seq2taxid_accession.py" script will generate 3 result files: raw+cab file (common relationships), raw_only file (relationships found only in the raw seqs2taxids file) and cab_only file (with sequences accessions found only in the cab file). In this case, the raw+cab resulting file is used as seq2taxid file.

Finally, we combined the resulting fNT_DB (FASTA and seq2taxid files) with the MS Set, obtaining the NT-MS_DB database used in the comparative. In parallel, the same was done with the MS-fMAG Set, obtaining the NT-MS-fMAG_DB database.