Pangenomic, phylogenetic and functional analysis pipeline

Gianuario Fortunato and Libe Renteria

November 2024

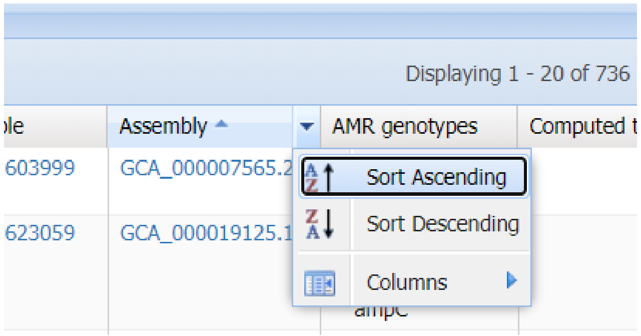
This pipeline outlines the steps for retrieving, annotating, and analyzing the genomes of Pseudomonas putida to identify core and accessory genes using pangenome analysis tools. It can be adapted for other bacterial species as well.

1 Download genomic sequences

We will first download all the genomes of interest form a trustable database. In our case, as our species of interest is *Pseudomonas putida*, the NCBI Pathogen Database will provide the necessary genomic data.

IN BRIEF

- **Source:** NCBI Pathogen Database
- **Task:** Download all available whole genome sequence for the species of interest. In our case, we will use *Pseudomonas putida*.
  In order to obtain the sequenced **whole genomes** we will need to filter the NCBI Pathogen entries by assembly. We will only get the genomes that are completely sequenced.



Filtering NCBI Pathogens entries by assembly

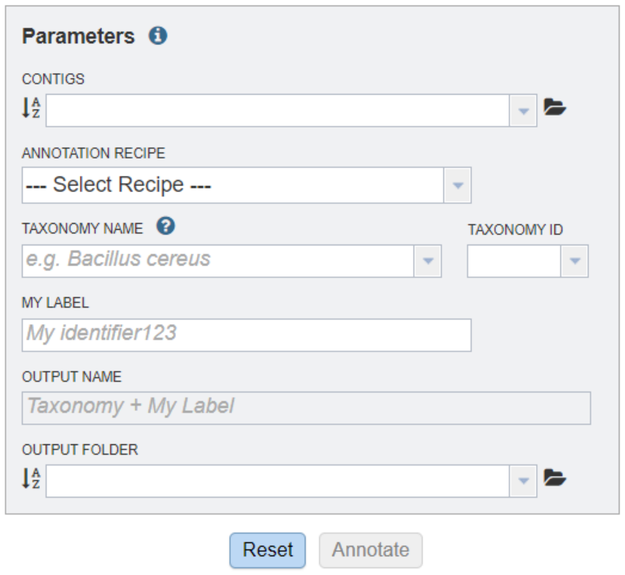
- **Output:** Genomic sequences in FASTA format

2 Genome annotation

2.1 Homogenize annotations

Even if some or all of the downloaded genomes are annotated, we advise to reannotate all the obtained genomes with the same annotating tool, in order to homogeneize our data.
In our case, we will annotate the genomes with the Bacterial and Viral Bioinformatics Resource Center (BV-BRC). For this, we will need to sign up and create an user in their web page, in order to obtain a workspace.

For each one of the genomes, we will use the Genome annotating BV-BRC tool. First we will create a folder in our Workspace that will contain all the annotation results.



Genome annotation window in BV-BRC

In the CONTIGS part, we will upload the genomes downloaded from NCBI, one by one.
In ANNOTATION RECIPE we will choose Bacteria/Archaea for our case, as we are analysing bacterial species.
In TAXONOMY NAME we will choose our species of interest, in our case *Pseudomonas putida*.
In MY LABEL we will choose the name of the annotation. In our case we will name each annotation with the name of the straing that is being annotated. Finally, in OUTPUT FOLDER we will choose the already created folder for the obtained results. I brief:

IN BRIEF

- **Tool:** BV-BRC (Bacterial and Viral Bioinformatics Resource Center)(www.bv-brc.org)
- **Task:** Upload the downloaded genomes to BV-BRC for genome annotation.
  - Create a workspace in BV-BRC
  - Upload the downloaded genomes
  - In "TOOLS AND SERVICES –¿ Genome annotation" choose the uploaded genomes and the corresponding species taxonomy name.
- **Output:** annotated genomes in General Feature Format (.gff) files

2.2 Gff to Gff3

The obtained output after the BV-BRC annotations are the .gff files of all the genomes. These files contain the annotation information of each genome, but in order to continue with the downstream analysis, we will need to convert these files into .gff3 files.
These .gff3 files include both annotation information and the genome sequence in FASTA format. The file first presents the annotation details, followed by a line labeled ##FASTA, after which the genome sequences in FASTA format are provided.
In order to do this we will download all the created .gff files, and we will put them **along the FASTA files of each genome** in a **zipped folder**. The name of both types of files should be the same for each strain, except the extension. Then we will run the following Google Colab file. In the following lines the script will be explained, in order to understand the output:

This first cell will open a prompt to select and upload the .zip file

```
import os
#upload folder with .fna and .gff
```

```python
from google.colab import files
uploaded = files.upload()
```

In this case the folder is called Data_for_creating_gff3_(Patricgff+originalfasta). Change the name of the folder in the following cells for the name of your folder.
When the folder is unzipped, we obtain a folder called Data_for_creating_gff3_(Patricgff+originalfasta) inside another folder called Data_for_creating_gff3_(Patricgff+originalfasta)
This is why we refer to the folder as:
Data_for_creating_gff3_(Patricgff+originalfasta)/Data_for_creating_gff3_(Patricgff+originalfasta).

After choosing the zip folder containing the .gff files, we weill unzip such folder with the following code:

```python
#unzip the folder
!unzip "Data_for_creating_gff3_(Patricgff+originalfasta).zip" -d "
    Data_for_creating_gff3_(Patricgff+originalfasta)"
```

After unziping the .gff folder, we will make sure that the content of the folder is right:

```python
# See the files in the extracted directory
folder_path = "Data_for_creating_gff3_(Patricgff+originalfasta)/
    Data_for_creating_gff3_(Patricgff+originalfasta)"
for filename in os.listdir(folder_path):
    print(filename)
```

In order to create the .gff3 files we need to make sure that there are no existing .gff3 files in the folder. If we have some important .gff3 files we should move them to another folder, as the following code will remove all existing ones. Also, if we run all the code and see that we have made some error, and we want to re-run it, we will have to remove the created .gff3 files and start again.

```python
# Loop through the files and remove .gff3 files
for filename in os.listdir(folder_path):
    if filename.endswith(".gff3"):
        os.remove(os.path.join(folder_path, filename))
        print(f"Removed: {filename}")
```

It could also happen that the .gff files downloaded from bv-brc have a double extension. If this is the case, we should run the following code in order to remove the second extension:

```python
# Loop through each file in the folder
for filename in os.listdir(folder_path):
    # Check if the file has a double .fna extension
    if filename.endswith(".fna.fna"):
        # Define the original and new file paths
        original_path = os.path.join(folder_path, filename)
        new_filename = filename.replace(".fna.fna", ".fna")
        new_path = os.path.join(folder_path, new_filename)

        # Rename the file
        os.rename(original_path, new_path)
        print(f"Renamed: {filename} to {new_filename}")
```

After making sure that the files in our folder are the desired .gff and .fasta files, we can start converting them into .gff3 files. the following code takes the content of each .gff file and copies it in our new .gff3 files. Then it adds a ##FASTA line followed by the fasta sequences taken from the .fasta files. If the files contain some naming error a Warning message is displayed.

```bash
%%bash
#this first line above changes the google colab language from python to bash
```

```

cd "Data_for_creating_gff3_(Patricgff+originalfasta)/Data_for_creating_gff3_(
    Patricgff+originalfasta)"
for gffile in *.gff
do
  #take the name of the .gff files without extension
  base_name="${gffile%.gff}"
  #fasta file will be the one with the same name
  fastafile="${base_name}.fna"
  #call the new .gff3 file with the asme name but with .gff3 extension
  newfile="${base_name}.gff3"
  if [[ -f "$fastafile" ]]
  then
    #add content of the .gff file
    cat "$gffile" > "$newfile"
    #add the ##FASTA line after
    echo '##FASTA' >> "$newfile"
    #finally add the genomic sequence in FASTA format
    cat "$fastafile" >> "$newfile"
    echo "Processed: $base_name"
  else
    #if the FASTA file and the gff file do not have the same name, or a FASTA
       file is missing:
    echo "Warning: FASTA file $fastafile not found for $gffile"
  fi
done
```

After this we will obtain a folder called 'Gff3_files.zip', where we will have all our .gff, .fasta and .gff3 files when we run the following code.

```
#download generated .gff3 files
import shutil
from google.colab import files

# Compress all processed files in the folder into a zip archive
shutil.make_archive('Gff3_files', 'zip', 'Data_for_creating_gff3_(Patricgff+
    originalfasta)')

# Download the zip file
files.download('Gff3_files.zip')
```

IN BRIEF

- **Task:** convert the annotated .gff files to .gff3 format, in order to process them with Roary. This requires:
  - Appending a '##FASTA' line at the end of the .gff file
  - Adding the corresponding FASTA sequence of the genome to each file after the ##FASTA line.
- **Steps:**
  - In order to convert all of the .gff files at once, first we have to save the .gff files in one directory, as well as the genome fasta sequences. The name of both files should be the same for each strain, except the extension: *XXXXXX.gff and XXXXXX.fna*
  - To access the script to convert the .gff files into .gff3 files click here.
- **Output:** Folder with .gff, .fasta and .gff3 (.gff files with embedded FASTA sequences)

3 Pangenome analysis

After annotating all the obtained genomes, we will proceed with the pangenome analysis with Roary. The program will accept the created .gff3 files as input, and it will divide the genes of our genomes of interest into core and accessory genes.

Core genes are the ones that are present in all of the genomes, while accessory genes are the ones that are not present in all of them.
In order to be able to use this program we will need to create a conda environment in our Linux terminal and download it there. For this, we will run the following code in our Linux terminal:

```
conda create -n roary
conda activate roary
conda install bioconda/label/cf201901::roary
```

Once Roary is installed, we will use a command with the following parameters to run it **while we are in the directory with all the .gff3 files**.

- **-e:** is for creating a multifasta alignment of core genes using PRANK
- **-z:** is for saving intermediate files
- **-r:** is for creating R plots
- **-mafft -p 10:** is for generating a core alignment using 10 threads

```
roary -e -z -r -mafft -p 10 *.gff3
```

IN BRIEF

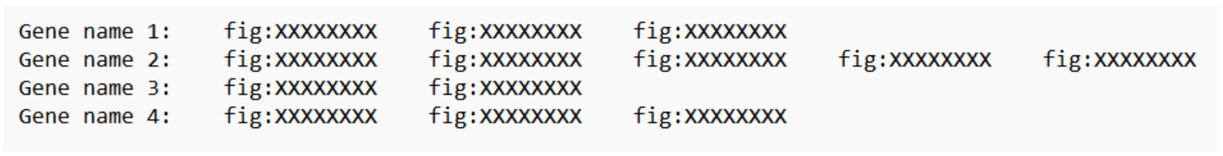
- **Tool:** Roary (Roary Pangenome Analysis Tool) (web page)
- **Task:** Perform pangenome analysis with all genomes using Roary
  - Input: .gff3 files from previous step
  - Roary will group genes into core (present in all genomes) and accessory (present in NOT all genomes) categories.
  - If there is not a conda environment with Roary installed, create it
  - Once it's installed, use the following command to run roary in the directory with all the .gff3 files:
    * roary -e -z -r -mafft -p 10 *.gff3
- **Output:** core and accessory gene files, gene presence/absence matrices and graphical summaries of the pangenome

After running roary, we will obtain the core and accessory gene files, gene presence/absence matrices and graphical summaries of the pangenome.

Now, we want to obtain the accessory gene's sequences to be able to annotate them and know what functions they are associated with. First we have to **go to the folder with all our.gff files in the terminal.** Then, we obtain the names of the accessory genes with the following code:

```
query_pan_genome -a complement -o accessory_genes.txt *.gff
```

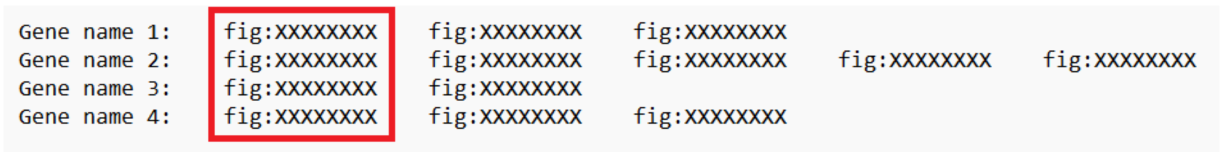
The output we get when we run the code above consists of a file called *accessory_genes.txt*, which contains the names of all the accessory genes and the FASTA headers of all the accessory genes.
The *accessory_genes.txt* file should have the following structure:



```
Gene name 1:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 2:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 3:    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 4:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
```

Structure of accessory_genes.txt file

As we can see in the picture above, each accessory gene can have several fasta headers (fig:XXXXXX). This means that even if accessory genes are not present in all the genomes, they can be present in more than one genome, or more than once in the same genome.
To enable the annotation of all accessory genes, we require a single protein sequence for each gene. Therefore, we will use the protein sequence from the first FASTA header for each accessory gene. In the following image we can see the selected FASTA headers inside the red square:



```
Gene name 1:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 2:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 3:    fig:XXXXXXXX    fig:XXXXXXXX
Gene name 4:    fig:XXXXXXXX    fig:XXXXXXXX    fig:XXXXXXXX
```

FASTA headers that will be extracted inside the red square

In order to extract these FASTA headers to later extract their protein sequences, we will run the following script:
CODE IN LINUX COMPUTER
In this script, we will process the *accessory_genes.txt* file by splitting each gene name at the last : symbol. This will separate the gene name from the rest of the data. Then, from the remaining fields, we will select the first element, which corresponds to the first FASTA header.
CODE IN LINUX COMPUTER
When we obtain our new file with all the accessory genes FASTA headers, called *accessory_fasta*, we will first add a symbol as the first symbol of the headers, just to give them the FASTA format. The following code will do this:

```
sed 's/^/>/' accessory_fasta > accessory_fasta_headers
```

Now that we have our FASTA headers for every accessory genes in the *accessory_fasta_headers* file, we will retrieve their protein sequence. For this, we will create a multifasta file that will contain all the FASTA sequences of all the proteins.
First we will go to a folder where all our FASTA proteins are located, and we will run the following code:

```
cat *.faa > allfassta.faa
```

Now that we have all our accessory gene names in *accessory_fasta_headers* and all our FASTA sequences in *allfasta.faa*, we will run the following code in order to retrieve the sequences of each FASTA header:

```
awk 'NR==FNR {arr[$0]; next} /^>/ {in_section = ($0 in arr) && !
   printed[$0]; if (in_section) {print; printed[$0] = 1}} !/^>/ &&
   in_section' accessory_fasta_headesr allfasta.faa >
   accessory_proteins.faa
```

This awk command extracts the protein sequences of accessory genes listed in accessory_fasta_headers from allfasta.faa and saves them to accessory_proteins.faa.
After having all our accessory genes and their protein sequences in *accessory_proteins.faa*, we will

zip the file in order to be able to use it as an input for eggNOG-mapper. For this we will use the following code:

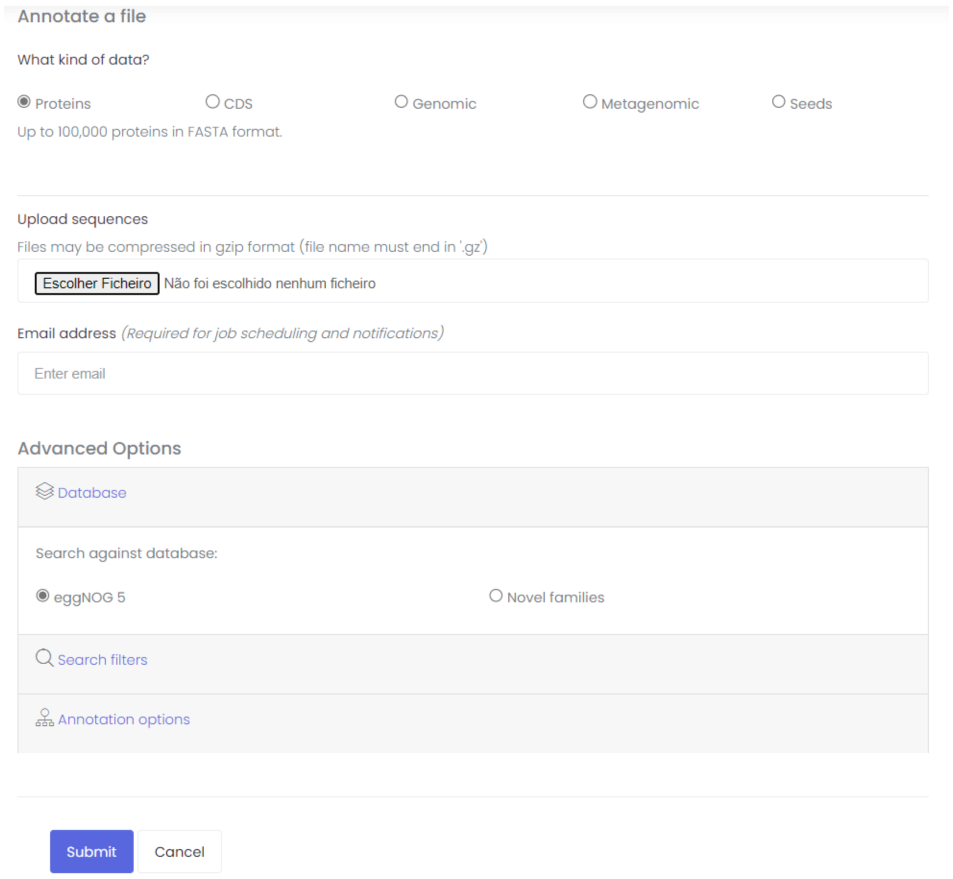
```
gzip -k accessory_proteins.faa
```

IN BRIEF

- **Task:** Obtain accessory genes and sequences for all genomes
  - query_pan_genome -a complement -o accessory_genes.txt *.gff
  - This command will create a file, *accessory_genes.txt*, that will list accessory genes along with the corresponding FASTA sequence identifiers for each genome that contains the gene. For example: Accessory_Gene_1: fig12345 fig67890 fig54321 Accessory_Gene_2: fig23456 fig78901
  - To ensure that each accessory gene is listed only once, we will extract the sequence corresponding to the first genome (i.e., the first figXXXXXX) that contains each accessory gene.
    - We will apply the following code:
    - **CODE IN LINUX COMPUTER**
  - We will combine all the proteomes in one file that includes all the proteic fasta sequences of all the genomes:
    - cat *.faa combined_proteome.faa
  - Retrieve the protein sequences of the obtained accessory gene names with the code that will appear when you click here.
  - After obtaining the document with the protein sequences of the accessory proteins, we will convert them into the .gz extension in order to give them as an input to eggnog mapper.
    - gzip -k accessory_proteins.faa
- **Output:** A zipped file (*accessory_proteeins.faa.gz*) containing all the accessory protein sequences, ready to be analyzed by the downstream analysis.

4 Functional analysis

Once we already have our accessory genes prepared with their protein sequences in our *accessory_proteins.fasta.gz* file, we will upload them onto the EggNOG-mapper web page, in order to annotate them.
In the webpage, we will choose our *accessory_proteins.fasta.gz* file, specify that it is proteic data and we will write our email adress in order to know when the job is finished.


Annotate a file

What kind of data?

(● Proteins) ○ CDS ○ Genomic ○ Metagenomic ○ Seeds

Up to 100,000 proteins in FASTA format.

Upload sequences

Files may be compressed in gzip format (file name must end in '.gz')

Escolher Ficheiro Não foi escolhido nenhum ficheiro

Email address *(Required for job scheduling and notifications)*

Enter email

Advanced Options

Database

Search against database:

(● eggNOG 5) ○ Novel families

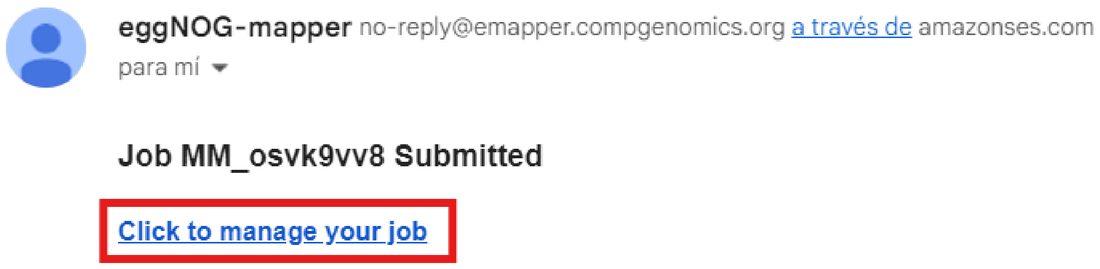
Search filters

Annotation options

Submit Cancel

EggNOG-mapper annotation site.

When we fill all the necessary parts, we will submit the job. An email will be delivered after that, and we will open it and start the job.


eggNOG-mapper no-reply@emapper.compgenomics.org a través de amazonses.com

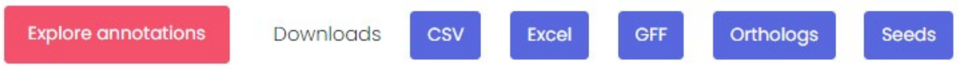
para mí

Job MM_osvk9vv8 Submitted

Click to manage your job

EggNOG-mapper Job submission e-mail. Click the line inside the red square.

When the job is finished, we will obtain different types of outputs:



All the outputs of the EggNOG-mapper annotation.

We will download the Excel document that contains all the annotations of the accessory genes. In order to see which of the annotated accessory genes are related to bioremediation, we will filter the results based on different metrics.

First, we will choose some **GO terms** that are related with bioremediation, in order to extract the genes that contain them.

Then, we will also extract the genes that are related with **KEGG pathways** involved in bioremediation. Finally, we will create a list with **key words** related with bioremediation in order to also filter according to the annotation names.

The filtering of the EggNOG-mapper annotation results will be done with the Google Colab script that can be accessed when you click here.
This code contains various GOterms, KEGG pathways and key words related with bioremediation; and it extracts all genes that contain said parameters into a new file that you can choose.

IN BRIEF

- **Tool:** EggNOG-mapper (web page)
- **Task I:** We will annotate all the accessory genes with EggNOG-mapper, in order to identify the genes that are related with bioremediation for each bacteria.
- **Task II:** As we have a lot of accessory genes annotated, we will filter them in order to only obtain the ones that are potentially related to bioremediation.
  - To access the Google Colab script that will filter the results, click here.
- **Output:** Excel containing the bioremediation related accessory genes of our genomes, annotated by EggNOG-mapper.

5 Phylogenetical analysis

Finally, we will perform a phylogenetical analysis by core MultiLocus Sequence Typing (cMLST). This part of the analysis will be carried out with the chewBBACA (web page) tool. We will use the core genes of the *Pseudomonas putida* genomes to perform this analysis, as these will give us more information than the usual seven housekeeping genes that are used for this type of analysis. As an output of the analysis, we will obtain a minimum spanning tree that will provide visual representation of the phylogeny between our *Pseudomonas putida* genomes.

In order to start with the analysis, we will need the FASTA files of the genomes of the study that we obtained early in the pipeline (here).

We will create a folder and insert all our fasta files in our Linux terminal:

```
mkdir FASTA_files
mv *.fasta FASTA_files
```

In order to be able to use this program we will need to create a conda environment in our Linux terminal and download it there. For this, we will run the following code in our Linux terminal:

```
conda deactivate #if we are in another environment
conda create -n chewBBACA
conda activate chewBBACA
conda install bioconda::chewbbaca
```

Now we will create a whole genome MLST schema, that will define the set of loci that will be compared between different strains. Reference schemes are preferred because they ensure a certain degree of standarization, but in our casee we will create our own scheme with our genomes.

```
chewBBACA.py CreateSchema -i ./FASTA_files/ -o wgMLST_scheme
```

Then, we proceed to the allele calling, in order to assign specific alleles to predefined loci in our set of bacterial genomes, and see the similarities and differences between them. This is the running code:

```
chewBBACA.py AlleleCall -i ./FASTA_files/ -g wgMLST_scheme/
   schema_seed/ -o results32_wgMLST --cpu 10
```

The output result would be something like the table below:

Strain A	Allele 1	Allele 4	Allele 2
Strain B	Allele 1	Allele 3	Missing
Strain C	Allele 2	Allele 4	Allele 3

After this, we will extract the paralogous genes that might confuse allele calling, as they have multiple copies in a genome.

```
chewBBACA.py RemoveGenes -i results32_wgMLST/results_alleles.tsv -g
    results32_wgMLST/paralogous_counts.tsv -o results32_wgMLST/
   results_alleles_NoParalogs.tsv
```

Finally, we extract the core genes to compare them between bacteria. We do this in order to be able to make stable comparisons, as we know all core genes are shared between all the genomes, and are les prone to change over time.

```
chewBBACA.py ExtractCgMLST -i results32_wgMLST/
   results_alleles_NoParalogs.tsv -o results32_wgMLST/cgMLST
```

After this we will be able to build our mínimum spanning phylogenetical tree with PHYLOViZ online (web page)

IN BRIEF

- **Tool:** chewBBACA (web page)
- **Task:** Perform phylogenetical analysis to obtain phylogenetical tree with core genome Multi-Locus Sequence Typing.
  - Input: FASTA files of the genomes of our database
  - Create conda environment and install chewBBACA
  - Create wgMLST schema with all our genomes
  - Perform the allele calling, in order to see the similarities and differences
  - Extract paralogous genes that may bias the result
  - Extract the core genes of the result, to obtain the cgMLST.
  - Build a minimum spanning tree in PhyloViz (web page)
- **Output:** We will obtain a mínimum spanning tree that will reflect the phylogenetic relationships between the different strains.