🧬 Insignia Pipeline Instructions (Alignment + Match Cover Phase)

✅ Objective

Identify shared genomic regions between a target genome and multiple background genomes using the `mummer` and `mcover` tools from the Insignia pipeline.

📁 File Organization

Assume the following folder layout:  
  
insignia-1.2025/  
├── programs/ ← compiled Insignia tools (e.g., mcover, mcover-intersect, common-mer)  
└── test-data/ ← genome FASTA files, MUMmer outputs, and match files

0️⃣ Clean Genomes Using `fasta-clean.pl`

Before alignment, clean and prepare all genomes using the `fasta-clean.pl` script.  
  
From inside the `programs/` directory, run the following command for each genome:

perl fasta-clean.pl ../test-data/raw\_input.fna > ../test-data/output.clean.fna

For example:

perl fasta-clean.pl ../test-data/bthu.fna > ../test-data/bthu.clean.fna  
perl fasta-clean.pl ../test-data/bamy.fna > ../test-data/bamy.clean.fna  
perl fasta-clean.pl ../test-data/blich.fna > ../test-data/blich.clean.fna  
perl fasta-clean.pl ../test-data/bpum.fna > ../test-data/bpum.clean.fna  
perl fasta-clean.pl ../test-data/bsub.fna > ../test-data/bsub.clean.fna

🔹 This removes low-quality characters and formats sequences properly for MUMmer.

1️⃣ Prepare Genome FASTA Files

Ensure the cleaned FASTA files exist:  
- bthu.clean.fna ← target genome (B. thuringiensis)  
- bamy.clean.fna, blich.clean.fna, bpum.clean.fna, bsub.clean.fna ← background genomes

2️⃣ Build Normalized Genome Database with `build.py`

After cleaning the genomes, use the `build.py` script to normalize headers, extract identifiers, and create a master FASTA database.

From inside the `db\_seq/` directory, run:

```bash

python3 build.py

```

🔹 This script:

- Scans `../test-data/` for all `\*.clean.fna` files

- Extracts the \*\*first word\*\* as the unique identifier (e.g., `NC\_009848.4`)

- Combines the \*\*next two words\*\* into a normalized species label (e.g., `Bacillus\_pumilus`)

- Writes new `.ins.fna` files with headers like `>NC\_009848.4 Bacillus\_pumilus`

- Concatenates all entries into `insigniadb.fna`

- Logs all changes in `insigniaindex.csv`, `insignia\_build.log`, and `insignia\_version.log`

⚠️ If any header fails to parse, the build is aborted and the error is logged in `insignia\_error.log`.

2️⃣ Generate Maximal Matches with `mummer`

From inside the `test-data/` directory:

mummer -maxmatch -b -c -n -F bthu.clean.fna bamy.clean.fna > bthu\_vs\_bamy.mums  
mummer -maxmatch -b -c -n -F bthu.clean.fna blich.clean.fna > bthu\_vs\_blich.mums  
mummer -maxmatch -b -c -n -F bthu.clean.fna bpum.clean.fna > bthu\_vs\_bpum.mums  
mummer -maxmatch -b -c -n -F bthu.clean.fna bsub.clean.fna > bthu\_vs\_bsub.mums

🔹 These `.mums` files contain the exact matches between target and each background genome.

3️⃣ Build Match Covers with `mcover`

From the `programs/` directory:

./mcover < ../test-data/bthu\_vs\_bamy.mums > ../test-data/bthu\_vs\_bamy.match  
./mcover < ../test-data/bthu\_vs\_blich.mums > ../test-data/bthu\_vs\_blich.match  
./mcover < ../test-data/bthu\_vs\_bpum.mums > ../test-data/bthu\_vs\_bpum.match  
./mcover < ../test-data/bthu\_vs\_bsub.mums > ../test-data/bthu\_vs\_bsub.match

🔹 Each `.match` file represents regions in B. thuringiensis that match a background genome.

🔜 Next Steps

You’re now ready for:  
- Creating `.T` (Target) and `.B` (Background) ID lists  
- Running `mcover-intersect` to find common regions  
- Running `common-mer` to extract shared k-mers