**GUI**

**How it would work:**

1. **Write Galaxy tool XML wrappers**
   * For each script (run\_pipeline\_module1.sh, run\_pipeline\_module2to6.sh), create a Galaxy **tool XML** definition.
   * This XML will describe:
     + Inputs (paths, parameters like INPUTDIR, ADAPTERFILE, etc.)
     + Outputs (processed files, reports)
     + How to call the script (the same bash command you have in the sbatch).
2. **Configure Galaxy to use SLURM**
   * In job\_conf.xml (Galaxy config), set up a **SLURM runner** so Galaxy submits jobs via sbatch.
   * Galaxy will handle job tracking, resource requests, and dependencies.
3. **Convert your pipeline to a Galaxy workflow**
   * In Galaxy, connect Module 1 (preprocessing) → Module 2–6 (array jobs).
   * Use Galaxy’s dataset collections for per-chromosome parallelization instead of a SLURM array (or keep your array if needed).
4. **Parameterize**
   * Instead of hardcoding paths (INPUTDIR, GENOMEINDEXFILE, etc.), expose them as Galaxy tool parameters (with defaults).

**two-step Galaxy workflow**

**—> Replace SLURM Array**

Instead of:

#SBATCH --array=0-50

CHROM=$(sed -n "$((SLURM\_ARRAY\_TASK\_ID + 1))p" "$GENOME\_CSV" | cut -d',' -f1)

You do:

* A preprocessing step to **extract chromosome names** from genome\_allchr.csv.
* Use that as a **Galaxy dataset collection** (one chromosome per item).
* Galaxy automatically dispatches one job per chromosome with --chromosome CHROM.

**Concrete Deliverables**:

1. A small script/tool to split genome\_allchr.csv into a Galaxy collection of chromosome names.
2. XML wrappers for: run\_pipeline\_module1.sh run\_pipeline\_module2to6.sh (taking --chromosome as parameter).
3. A Galaxy workflow diagram showing how these link together.
4. How to use it:

python genome\_parse\_toJson.py genome\_allchr.csv chromosomes\_collection.json

* Input: your CSV file
* Output: JSON file usable in Galaxy as a dataset collection descriptor